CHAPTER 2

Obtaining and Installing SQL Server

As you saw in Chapter 1, there are plenty of versions of SQL Server available. We're going to use SQL Server Express, which is completely free, as the basis of this book. Although much more limited than other versions, Express is still very powerful and includes all of the features we are going to look at during the course of the book.

Once we've downloaded Express, we'll walk through installing it. You'll see that installing SQL Server is a bit more involved than the normal installation process you might be used to!

Downloading SQL Server Express

Just as there are multiple editions of SQL Server, so are there multiple editions of SQL Server 2014 Express. Use this URL to access the download page.

```
www.microsoft.com/en-gb/download/details.aspx?id=42299 (http://www.microsoft.com/en-gb/download/details.aspx?id=42
```

If this link doesn't work (Microsoft is always changing them around), use your favorite search engine to look for *sql server express 2014 download*. The first result returned should be the one you are after.

When you are taken to the download page shown in Figure 2-1, you'll see a list of download options. Go for Express Advanced ("ExpressAdv"), as this contains everything we need to work through the examples in this book.

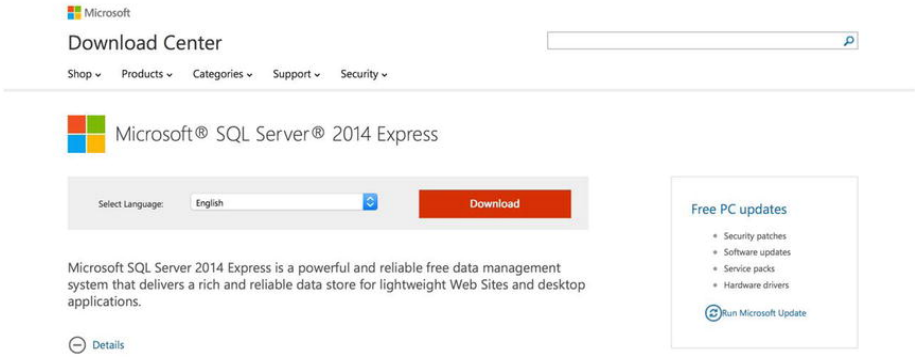

Figure 2-1. *The SQL Server 2014 Express download page*

Download the 32- or 64-bit version as appropriate. Once it has downloaded, double-click it to begin installation. The download is about 1.1GB, so it may take a while, depending on the speed of your Internet connection.

Now, you are probably used to installing programs where you click a **Next** button a few times and you're done. Things are even easier these days, as installing an application from an app store just involves clicking on an app and entering your password. Unfortunately, things are not so simple with SQL Server. There are a lot of things to configure before you can begin tinkering around with databases, so prepare to answer some questions.

We're installing the basic version, Express, but if you can install Express you'll be able to install any version. Let's take a quick look at what we are going to do:

- Allow the installer to extract files to our hard disk
- Choose to install a new stand-alone installation
- Follow the steps as guided by the installer

Before starting installation, you need to make sure SQL Server can be installed on your version of Windows. Versions of Windows supported are:

- Windows Server 2012
- Windows Server 2008
- Windows 8.1
- Windows 8
- Windows 7

Note that SQL Server Enterprise, Business Intelligence, and Web editions may only be installed on Windows Server 2012 or 2008.

Happy with your version of Windows? Cool, let's go!

Beginning Installation

Double-click the EXE file you downloaded and allow the installer to extract the files to the location requested (feel free to change this if you feel the need). After the extraction completes, you'll see the SQL Server Installation Center (Figure 2-2).

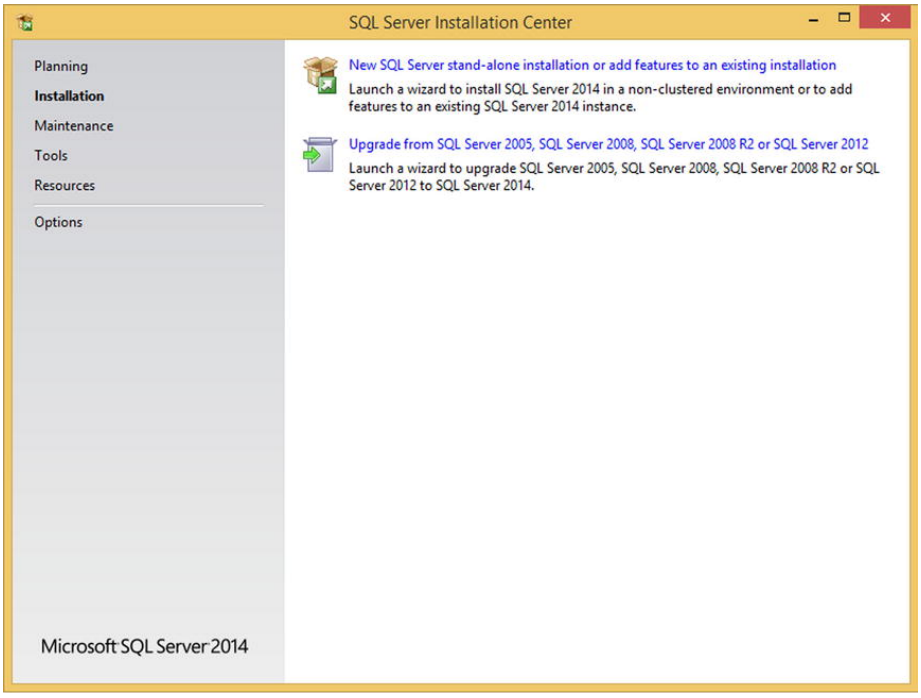

Figure 2-2. *SQL Server Installation Center*

This presents us with a few options. If you look to the left, you'll see we've actually started off at the second option, **Installation** (this is the default). Click **Planning**, which is worth a look. It gives you access to documentation telling you what SQL Server requires, and can also run a system check to ensure your machine can run SQL Server. There are plenty of other things there, too, so feel free to take a few minutes and have a play around.

All done? Hope you enjoyed yourself! Return to **Installation**, and choose **New SQL Server stand-alone installation or add features to an existing installation**. We are going to begin a clean installation of SQL Server. We are not upgrading from a previous version of SQL Server, so we can ignore the other option, but remember it is there should you ever need it. Likewise the other options presented on the left (**Maintenance**, **Tools**, **Resources**, **Options**)—you may never need them, but take a quick look so you know what they have to offer you.

Once you've clicked the **New SQL Server** option, the real installation wizard appears.

Installation Wizard Steps

You should now see the wizard in Figure 2-3, which probably looks pretty similar to other installation wizards you have seen.

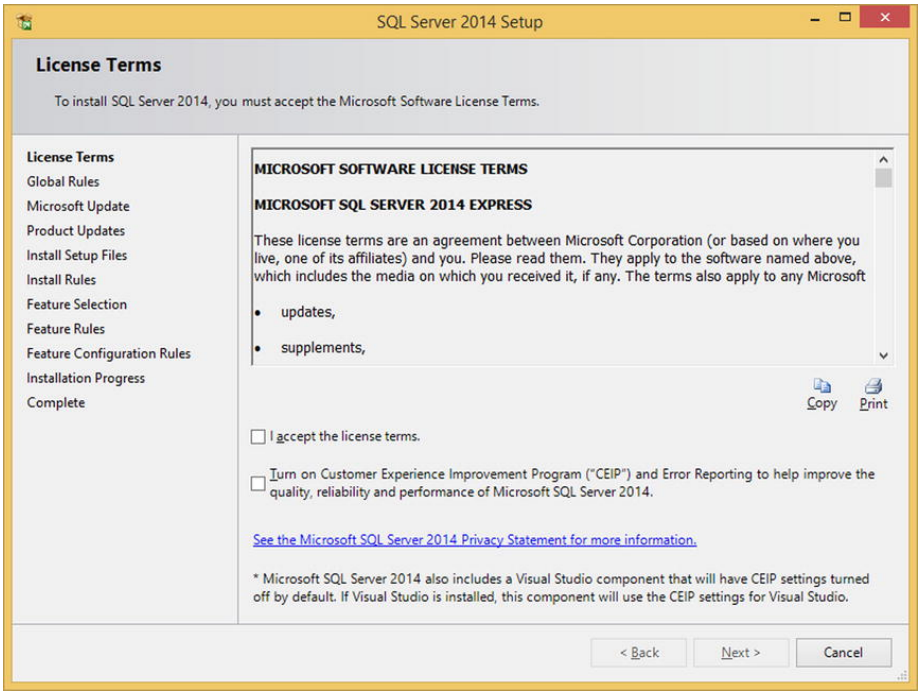

Figure 2-3. *The license acceptance dialog*

Take a look at all of the steps on the left! We need to configure a few things before SQL Server will be ready for use. Here is what we are going to do:

- The wizard will perform a couple of update checks to make sure we have the latest software. It may download some updates if required.
- We'll choose the features of SQL Server we want to install.
- We'll configure those features.
- SQL Server will install them.
- We'll be happy.

The first step on the road to happiness is to accept the license terms—you might even want to read them if you have a spare year. (Seriously, has *anybody* ever read a licensing agreement?!) You can also choose whether you want to send data to Microsoft to help them improve SQL Server.

Tick the license terms box, tick the Customer Experience Improvement Program box if you wish, and click **Next** to move on.

Updates and Features

Global Rules will install automatically, and then you'll be taken to the Microsoft Update page (Figure 2-4). The updates box is not checked by default, but it's a good idea to check for them, so I recommend checking the box before clicking **Next**.

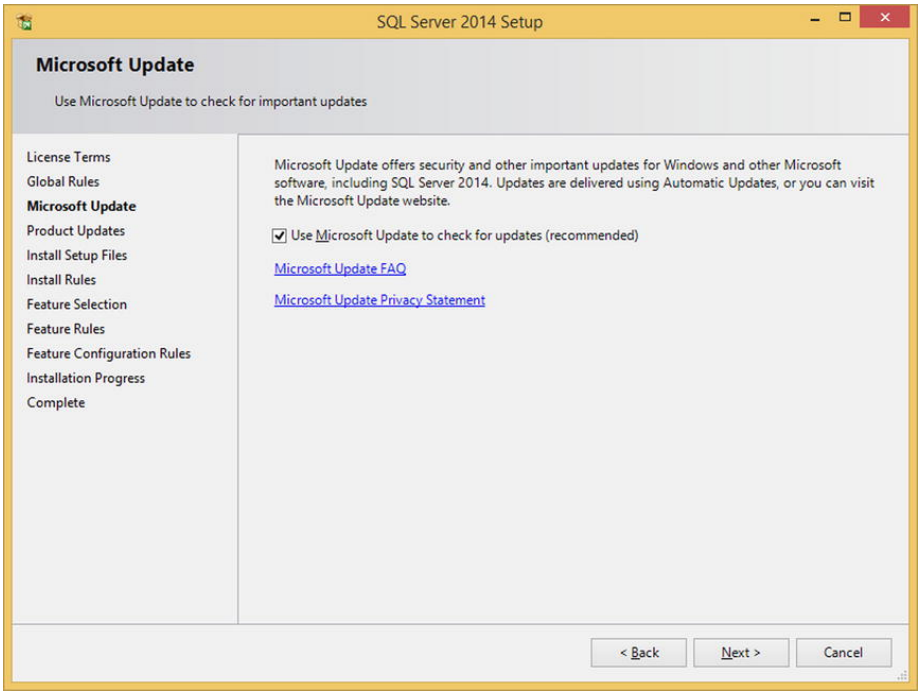

Figure 2-4. *The Updates and Features dialog*

Once you've done this, you'll probably be taken straight to the **Feature Selection** page. If you are not, let the update search complete/apply and then carry on.

Feature Selection is the most important page of the wizard. It is where you choose which pieces of SQL Server you want to install (you can see the checklist of options in Figure 2-5). I may as well tell you now that we are going to install everything (hence the reason for downloading the Express Advanced version), but here's a quick list of just exactly what those features are.

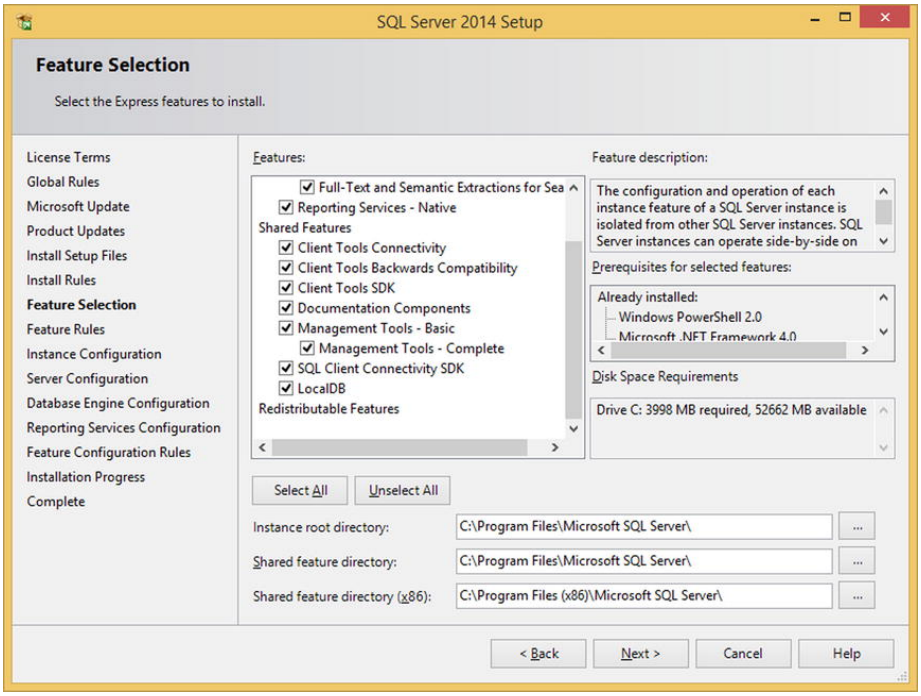

Figure 2-5. *The Feature Selection dialog*

Instance Features

- **Database Engine Services**: The core database engine. This is what we are going to be interacting with through SQL Server Management Studio.
- **SQL Server Replication**: You can configure SQL Server so it copies data from one server to others. This is an advanced administrative topic and is out of scope for our purposes.
- **Full-Text and Semantic Extractions for Search**: This is the full-text search feature I mentioned in Chapter 1. Again, this is an advanced topic so we won't look at it any further.
- **Reporting Services—Native**: These are the SQL Server Reporting Services (SSRS). SQL Server Express does not support SQL Server Integration Services (SSIS) or SQL Server Analysis Services (SSAS).

Shared Features

Most of these tools are required to allow you to communicate with your SQL Server. They can be used to manage other instances of SQL Server too (you'll find out what an instance is *very* soon).

- **Client Tools Connectivity**: Allows client machines to talk to SQL Server by installing a set of network libraries.
- **Client Tools Backward Compatibility**: Support for older client tools.
- **Client Tools SDK (Software Development Kit)**: Programming tools, which allow developers to write code to manage aspects of SQL Server.
- **Documentation Components**: Tools that allow you to view SQL Server help (formerly Books Online) via the Internet. You can also choose to download help to your computer (a good idea if you ask me!). Books Online can be accessed on the Web at `https://technet.microsoft.com/en-us/library/ms130214.aspx`.
- **Management Tools—Basic (and Complete)**: This is pretty much SQL Server Management Studio, although some other utilities like SQLCMD are also installed. The Complete option installs tools like SQL Server Profiler, and also adds support to SSMS for SSRS, SSAS, and SSIS.
- **SQL Client Connectivity SDK**: You need this if you intend to develop database applications.
- **Local DB**: A lightweight version of SQL Server, aimed at lower-powered devices like mobile phones and tablets.

Click the **Select All** button to add all items to the installation. Now note the three paths at the bottom of the wizard page:

- Instance root directory
- Shared feature directory
- Shared feature directory (x86)

I mentioned instances a moment ago. I'll explain properly in a moment, but for now, know that the **Instance root directory** is the directory in which all files for your database server instance will be placed.

A single server can host multiple instances of SQL Server, but they will share many features, such as SQL Server Management Studio. There is no point in installing a copy of SSMS per instance—this would just waste disk space—so the **Shared feature directory** stores files used across instances. Do you recall our brief conversation earlier about 32-bit and 64-bit versions of SQL Server? The presence of these two architectures is why we have two shared feature directories. The x86 version stores 32-bit files, and the other directory stores the 64-bit files. Some pieces of SQL Server are still 32-bit only, hence the need for the x86 version.

We now have to decide where these files should be placed. Most places I have worked keep the default `C:\Program Files` locations. Feel free to change this if you have a different disk structure. All I would say is ensure the databases you create are not put on the same drive as the SQL Server files—this will affect performance. Unfortunately, this is a situation I see on a regular basis.

I digress. If you have checked all the boxes and are happy with the installation directories, click **Next**. The installer will run a check, ensuring various prerequisites are present. I performed the installation on a clean server, so you can see from **Figure 2-6** that I hit a problem with the .NET Framework 3.5.

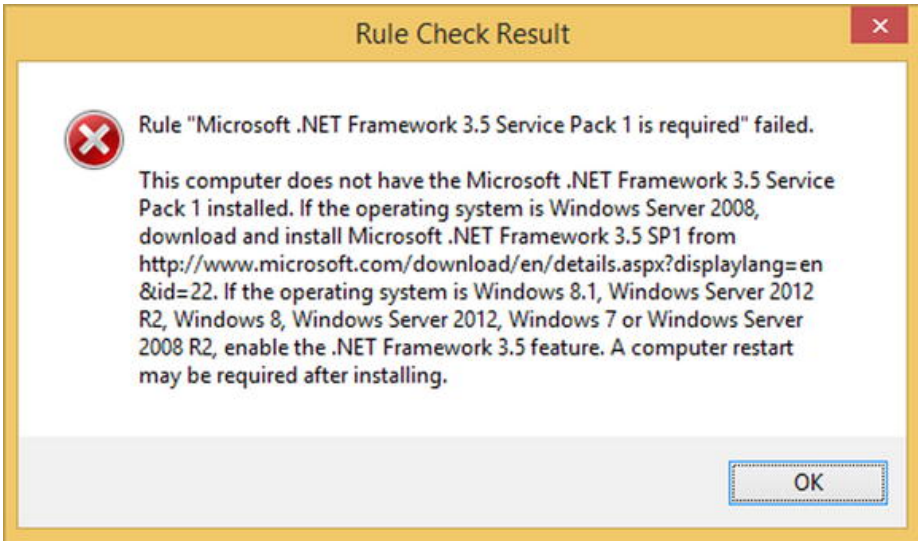

Figure 2-6. *The "Microsoft .NET Framework 3.5 Service Pack 1 is required" error*

The dialog shown in Figure 2-6 is displayed when you click the **Failed** message next to the appropriate item in the rule check list (not pictured). The dialog displayed after clicking a **Failed** message will explain how you can resolve the issues with that particular rule. Figure 2-6 shows that Microsoft has explained how to obtain and install the Microsoft .NET Framework 3.5 Service Pack 1 software, which will resolve the rule check failure.

I happen to be running Windows 8.1, so I followed the instructions for enabling .NET 3.5 (these instructions also apply to Windows 7 and 8). Open **Control Panel**, click **Programs**, and click **Turn Windows features on or off**. When the list of features appears (Figure 2-7), tick **.NET Framework 3.5 (includes .NET 2.0 and 3.0)**, then click **OK**. The selected features will install (you may be asked to download files from Windows Update).

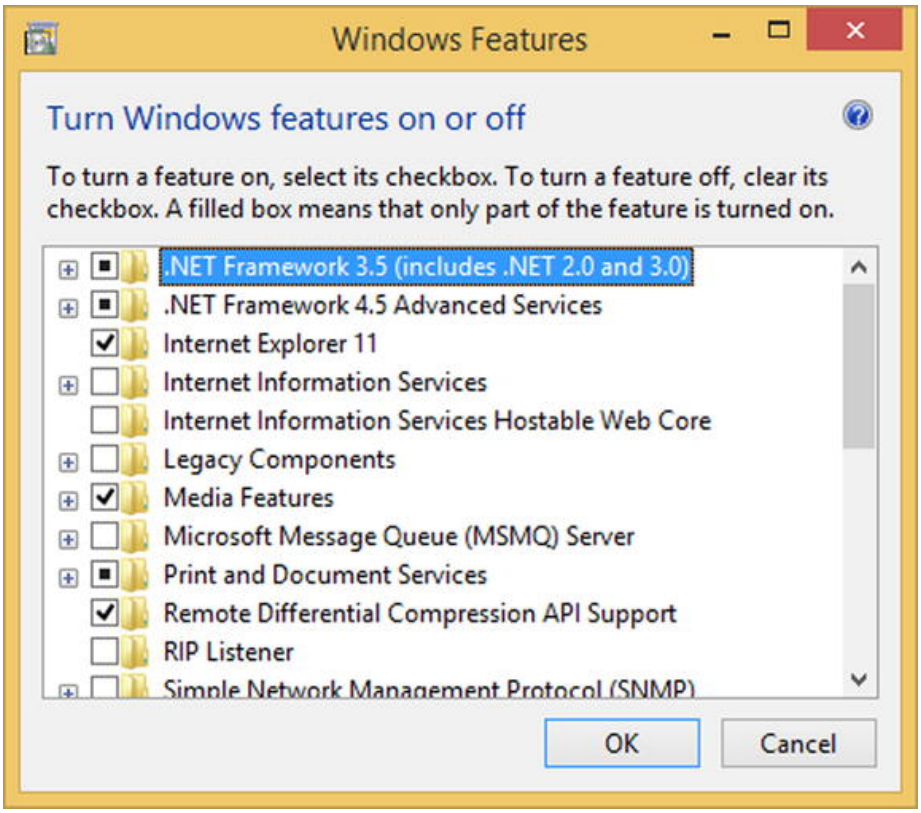

Figure 2-7. *Installing the .NET Framework 3.5*

Kick back for a moment if you are waiting for the file download and installation!

I didn't need to reboot when I did this, so hopefully you won't, either (Windows has really improved in this area). Return to the SQL Server installer, and click the button to **Re-run** the feature rule checks. This time all rules should pass and you'll be taken to the Instance Configuration page. Time to learn about instances. . . .

Database Instances

SQL Server now wants to know what type of instance you would like to create. The screen should look something like Figure 2-8.

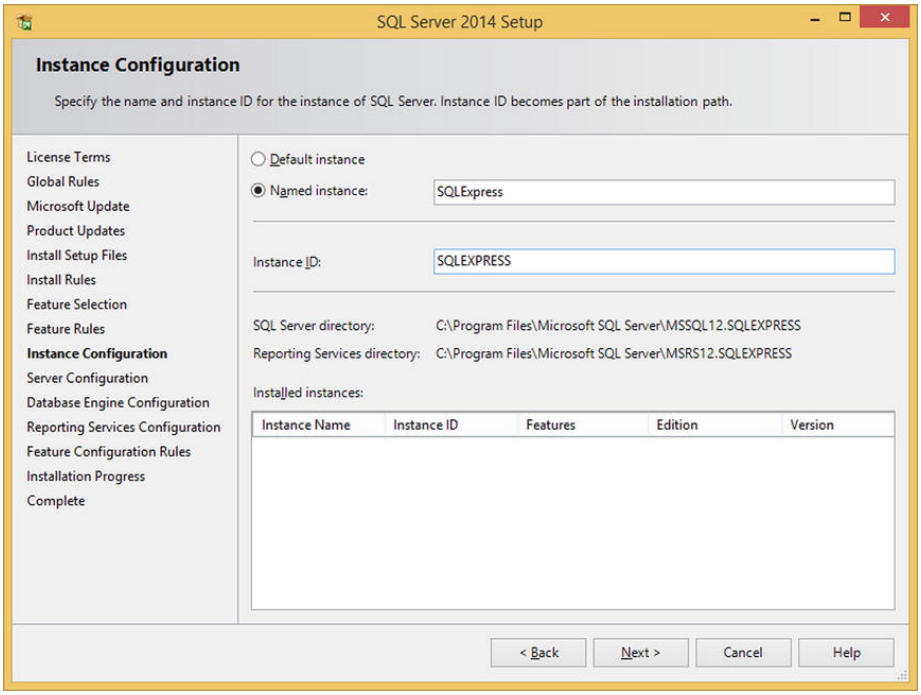

Figure 2-8. *The Instance Configuration dialog*

You can see there are two types of instances: Default and Named. But just what *is* an instance? An instance is a SQL Server installation that can house databases, SSAS, SSIS, and SSRS, as well as all of the other SQL Server instance components. Each instance has its own security. Assume we have two instances:

- Development
- Testing

Both of these instances can exist on the same computer. You may have full administration rights on Development, meaning you can do anything there—create databases, add users, and so on. Yet you may not have any access at all to Testing, meaning you cannot create databases on that instance—or even run a `SELECT` statement there.

You might be thinking, "Why do I need multiple instances? I'll just put all my databases on the same instance." Unfortunately, this doesn't give you any logical separation of concerns—it can often make sense to create an instance for a particular department or function. Instances also allow you to limit the number of databases each instance has to handle, as this can affect performance.

Say you have an instance with five databases on it. One particular database is used a lot, but the other four are used moderately. Users may complain that the applications accessing the moderately used databases are performing poorly; this is because the busy database is starving them of resources. At this point, it would be a good idea to move the busy database to its own instance, to reduce contention with the other databases.

I did mention a single computer can run multiple instances of SQL Server. I've rarely seen this implemented in production environments successfully. Note that I am *not* saying it cannot be done; I'm just saying that most organizations don't plan these things out. My general rule of thumb is if you need a new instance, put it on a new server (be it physical or virtualized). It will reduce contention and confusion. If there is a time when multiple instances of SQL Server need to run on the same physical computer or server, I strongly recommend a virtualization agent such as VMWare to utilize the resources of the virtualized instance as if it were a separate

database server. However, virtualization is an advanced concept more closely related to Windows Administration, and is out of the breadth of this book.

Default Instances

A default instance uses the computer name. Let's say your computer is called HAL9000. The SQL Server instance will use this name to identify itself. As a result, you can only install one default instance on a computer. You should install a default instance if the computer you are installing to is only going to host one SQL Server instance.

I prefer using default instances to named instances because it means people accessing the database server only need to remember one name: the server name. Named instances can be useful but they do require people to remember two things: the server name and the instance name. They do have their place, however, as we'll see in the very next section.

Named Instances

As I just said, named instances need an extra name to identify them. Say you wanted to install an extra instance on the HAL9000 computer, called Aries. You would access this using the name:

```
HAL9000\ARIES
```

If you just typed in `HAL9000` you would be connected to the default instance.

Named instances come into their own on development machines. You may be building software that needs to run on SQL Server 2005, 2008, 2012, and 2014. You can install a named instance of each of these on your development machine, giving each instance a unique name. Assuming my development machine is called HOLMES, I could have:

- `HOLMES\SQL2005`
- `HOLMES\SQL2008`
- `HOLMES\SQL2012`
- `HOLMES\SQL2014`

This kind of setup really aids development.

One last point: Each server can host up to 50 instances. I really hope I never come across a server with that many instances on it!

Clear on instances now? Cool! We'll continue with the installation.

The Instance Configuration Page

You'll note in Figure 2-8 that the SQL Server installer has selected a named instance by default. This is because we are installing SQL Server Express. Any other version of SQL Server will have a default instance selected when you hit this page.

It is conventional to install SQL Server Express as a named instance, called SQLEXPRESS. This is what the install defaults to and it is what most organizations use—in fact, I don't think I've ever seen SQL Express installed as a default instance. (This just goes to show that most people keep clicking **Next** during installation!)

We'll side with the majority and install as a named instance. Before we do, note the **Instance ID** text box in Figure 2-8. This is the internal ID SQL Server uses to identify the instance. It is good practice to keep it the same as the instance name—I can't think of a situation where you would want to change it. The Instance ID affects the name of the directory to which the instance files are installed.

If you change the value of the **Instance ID** text box, you'll see the end of the SQL Server directory and the Reporting Services directory values change to match it. We cannot modify these paths directly. The two paths at the middle of the screen show where the instance files will be installed. You'll see **MSSQL12** and **MSRS12** in these paths; this shows that we are using version 12 of SQL Server (this is Microsoft's internal version number; the public or marketing version is **2014**).

Make sure you have **Named instance** selected, and that **SQLEXPRESS** is present in both the **Named instance** and **Instance ID** textboxes. Then click **Next** to begin entering service account information.

Service Accounts

To be able to run correctly, SQL Server needs appropriate permissions. These permissions are obtained via a user account that is configured to run the SQL Server services. A service is a program like any other, except it has no user interface and is constantly running in the background. As an example, the SQL Server Database Engine runs as a service, and to work correctly it requires users to assign an account with appropriate permissions to it (Figure 2-9).

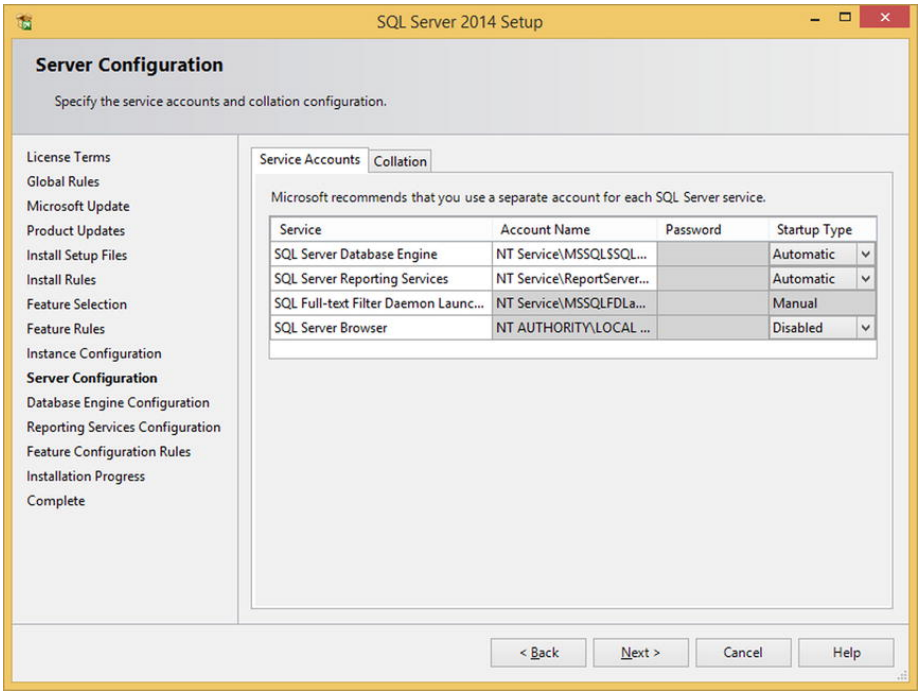

Figure 2-9. *Configuring service accounts dialog*

We can see four services in Figure 2-9:

- SQL Server Database Engine
- SQL Server Reporting Services
- SQL Full-text Filter Daemon Launcher
- SQL Server Browser

We were introduced to these services in Chapter 1 (the SQL Full-text service has a longer name on this screen, though). Strictly speaking, each service should be assigned a distinct user account. There are two reasons behind this:

- Each service has different permission requirements.
- Using the same account for all services will give an attacker access to all services should the account become compromised.

These are issues to consider if you are installing a nondevelopment instance. If you are installing a local instance on your own development machine, I wouldn't worry too much about configuring multiple accounts, unless you want some practice. If you are installing a production instance you should definitely use different accounts.

In the table listing the service accounts on this screen are four columns:

- Service
- Account Name
- Password

- Startup Type

Looking at the Startup Type first, you can see three values in the dropdown list: **Automatic**, **Manual**, and **Disabled**. **Automatic**means the service will start when Windows starts. **Manual** ensures the service won't start automatically but you can start it yourself should you wish, via the Services applet in the Control Panel. Finally, selecting **Disabled** makes the service unavailable for use; if you wish to use it you will have to enable it in the **Services** applet.

The Account Name and Password columns are self-explanatory; these are the account details the services will use. If this were an administration book I'd go into a lot more detail about these, but I'm mainly concentrating on development, so leave the default options selected.

Collation

On the same page is a Collation tab. Click this and you'll see a text box containing a value. Your Windows configuration will determine which value you see here. Figure 2-10 shows my Windows 8.1 machine's collation, `Latin1_General_CI_AS`. Windows 7 installations may see the value `SQL_Latin1_General_CP1_CI_AS` instead.

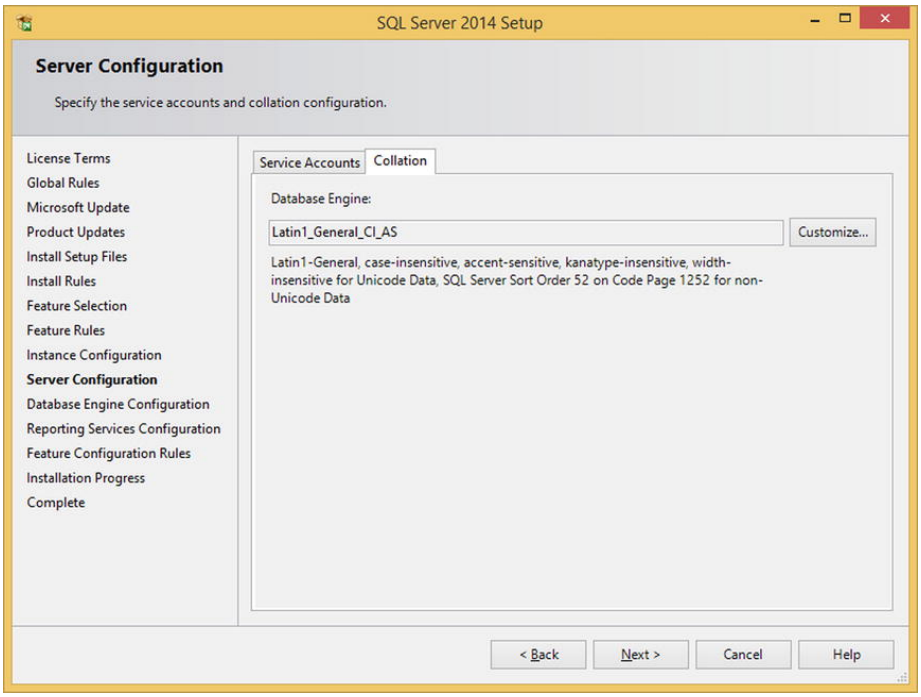

Figure 2-10. *Choosing a collation*

A collation tells SQL Server how it should consider the values used in the database. `Latin1_General` represents languages using common alphabets and rules for sorting and comparing characters, so immediately we are saying the language chosen for our database will follow certain well-established rules. Our language of choice in this instance will be US English, which complies with this definition of `Latin1_General`. *CI* stands for Case Insensitive, and this clause will ensure our queries will be treated the same no matter how values are specified. If CI is chosen, both of these queries will return the same result:

```
SELECT PupId, PupName, DateOfBirth FROM Pups WHERE PupName = 'Bertie';

SELECT PupId, PupName, DateOfBirth FROM Pups WHERE PupName = 'bertie';
```

If *CS* (Case Sensitive) was used, then only one of the above queries would return a result (assuming a record with the name Bertie exists).

The *AS* part is for Accent Sensitive, and it causes SQL Server to treat characters with accents as distinct characters—so *c* will not be the same as *ç*, for example. Usually you will keep the collation that the installer selects for you, unless you know you have specific requirements (e.g., you are going to be dealing with Japanese characters, in which case you'll need to specify Kana sensitivity).

Click the **Customize** button to view the collation options available to you (shown in Figure 2-11). If you feel the need to change your collation, go ahead. Just remember that while you can change a collation at the server level, database level, and even the column level, doing so is not a straightforward task for the uninitiated. This is one of those things it is better to do right first time.

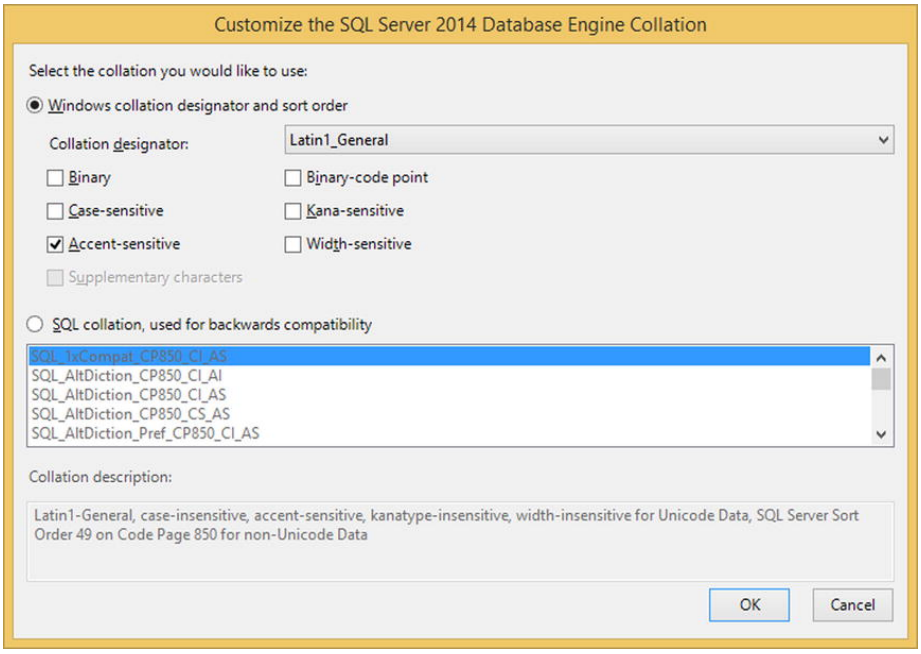

Figure 2-11. *Configuring the collation*

Also, if you have multiple servers and/or databases, make sure they all use the same collation, or you may find yourself having to use the `COLLATE` keyword in your queries. This isn't pleasant—you have been warned!

KEEPING COLLATIONS CONSISTENT

Unless you have specific needs regarding collation configuration, it is advisable to apply the required collation at the server level. If possible, use the same collation on all your servers. This will make any cross-server queries you need to write easier, and will also ensure the way SQL Server handles your data will be consistent from a collation point of view.

When you are happy with your selections, click **Next**. Still a little way to go yet. . . .

Database Engine Configuration

You can tell we are in the homestretch, as we now configure the individual pieces of SQL Server. First up is the main Database Engine, which is what we'll be engaging with once installation is complete. The following sections describe the four tabs on this screen.

Server Configuration

You'll do two things here: specify the security model your instance will use, and specify administrator accounts. As Figure 2-12 shows, you should see **Windows authentication mode** selected by default. This is the more secure option, as it integrates with other Windows servers on your network, using Active Directory (see **Appendix B**) to map user accounts to SQL Server logins. Even if you don't use Active Directory, SQL Server will use the User Groups and Accounts on your own computer to obtain the user account details. I prefer this method because it allows me to centralize all of my accounts and my account management. It also has the full weight of the Windows security model behind it.

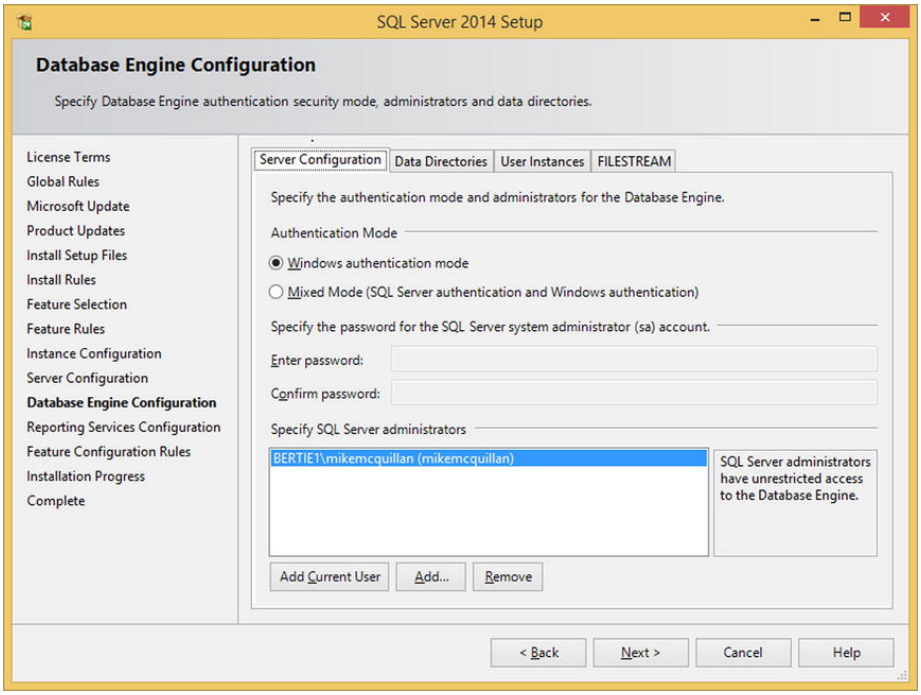

Figure 2-12. *The Server Configuration tab*

If you choose **Mixed Mode** you can still use Windows authentication, but you also will enable SQL Server authentication. This is SQL Server's own internal user account engine. There is a default account, **sa**, which originally came configured without a password (the installer will force you to specify one now). Lots of administrators never used to turn this account off—immediate security hole! Since **sa** is a system administration account, full access to the server can be gained via this account.

My issue with SQL Server authentication comes from connection strings. Connection strings are used by .NET and other applications to connect to SQL Server databases (a brief description can be found in **Appendix B**). You must specify a user account to connect to the SQL Server in the connection string. With Windows authentication, you don't need to provide any account details, but you have to specify the username and password if SQL Server authentication is used, which means those details can be seen (and used) by anybody who can access the configuration file containing the connection string (this pretty much includes all of your developers).

This is a Windows Authentication connection string:

```
Server=HOLMES\SQL2014;Database=WatsonDB;Trusted_Connection=True;
```

And here is a SQL Server Authentication version:

```
Server=HOLMES\2014;Database=WatsonDB;User Id=John.Watson;Password=Sherlock;
```

As you can see, the username and password are easily visible in the SQL Server Authentication version. Do yourself a favor and steer clear, unless you have a specific need (e.g., backward compatibility).

At the bottom of this screen, you can specify the administrator accounts. Accounts you add here will be configured as system administrators in SQL Server (I discuss the different levels of security available in Chapter 20). Your own user account should already be in this list. If it is not, click **Add Current User** to add yourself. You can add and remove other users using the buttons provided.

Note System administrators (sysadmins) can perform any action on a SQL Server. The chapters in this book all presume you have sysadmin permission.

Data Directories

This is straightforward enough—it shows the paths where databases will be installed, as seen in Figure 2-13.

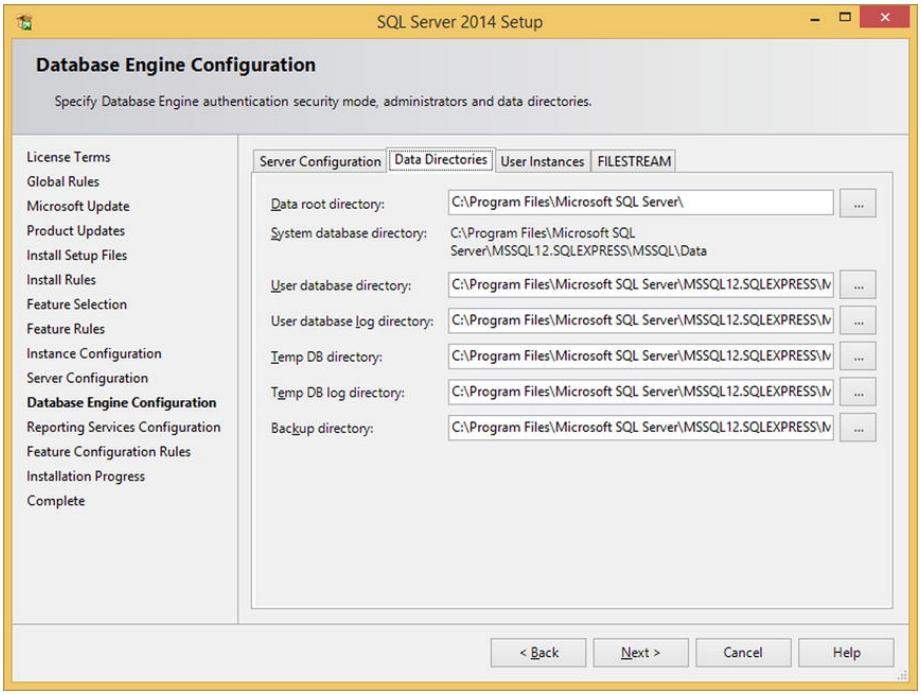

Figure 2-13. *The Data Directories tab*

Briefly, we have:

- **Data root directory**: The top-level directory, housing all other data folders.
- **System database directory**: For **master**, **msdb**, and so on (see Chapter 3). You cannot change this path.
- **User database directory**: The default directory where SQL Server will create your database files, if no alternative path is provided.
- **User database log directory**: As described, but for log files.
- **Temp DB directory**: The location where SQL Server will create the **TempDB** database.
- **Temp DB log directory**: As described, but for Temp DB's log file.
- **Backup directory**: The folder to which backups are saved, if no path is provided in the `BACKUP DATABASE` command.

We'll look at how databases are structured and what logs are used for in Chapter 3.

User Instances

As Figure 2-14 shows, this is turned on by default, and it allows users who do not have administrative rights to run their own separate instance of the SQL Server. This instance is generated from the parent instance. This feature should be used by developers who do not have local administrator permissions on their own machines (litmus test: if you can install apps on your machine, you are probably a local administrator). This is a surprisingly common configuration in many corporate environments, which is no doubt why Microsoft brought this feature into SQL Server Express.

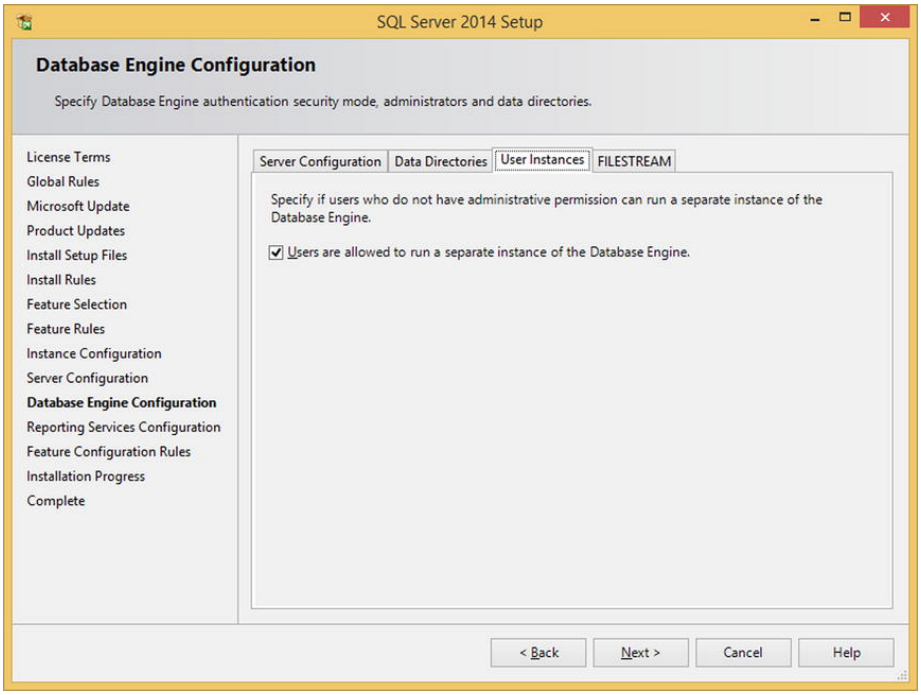

Figure 2-14. *Adding user instance support*

The user in question must have sysadmin permissions in SQL Server Express to use this feature.

FILESTREAM

This is way out of scope for this book, but I'll touch on it. Let's say you have a system that needs to store file attachments: documents, pictures, videos, and so forth. These cannot be easily stored in a typical SQL Server database. FILESTREAM creates a special folder on your computer or network to store these files. You then mark the appropriate columns for these files as FILESTREAM columns, and anything you insert or update into those columns will be saved in the special folder.

You can access content held in FILESTREAM just as you would any other SQL Server column. If you ever find yourself needing to store external files in a database, have a look at this feature—it may be just what you need. To turn this feature on, check the **Enable FILESTREAM** checkbox you can see in Figure 2-15.

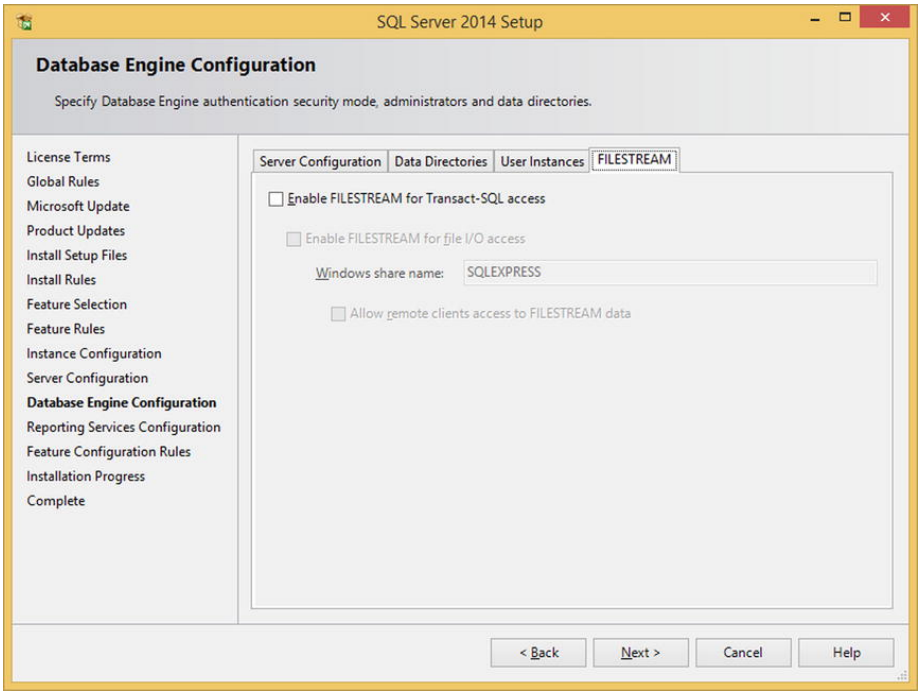

Figure 2-15. *The FILESTREAM tab*

We're all done with our Database Engine configuration, so we'll proceed to the Reporting Services Configuration. Click **Next**.

Reporting Services Configuration

Again, we won't spend too much time here as we won't be looking at Reporting Services in this book. There are two modes, of which the first, Native Mode, is commonly used (the other option integrates with Microsoft SharePoint and will only be enabled if you actually have SharePoint installed).

As Figure 2-16 shows, you can choose to **Install and configure**, which means the SSRS server will be ready to use once installation is complete (this is the default), or you can just **Install only**, which means you'll need to use the Reporting Services Configuration Manager to complete the installation.

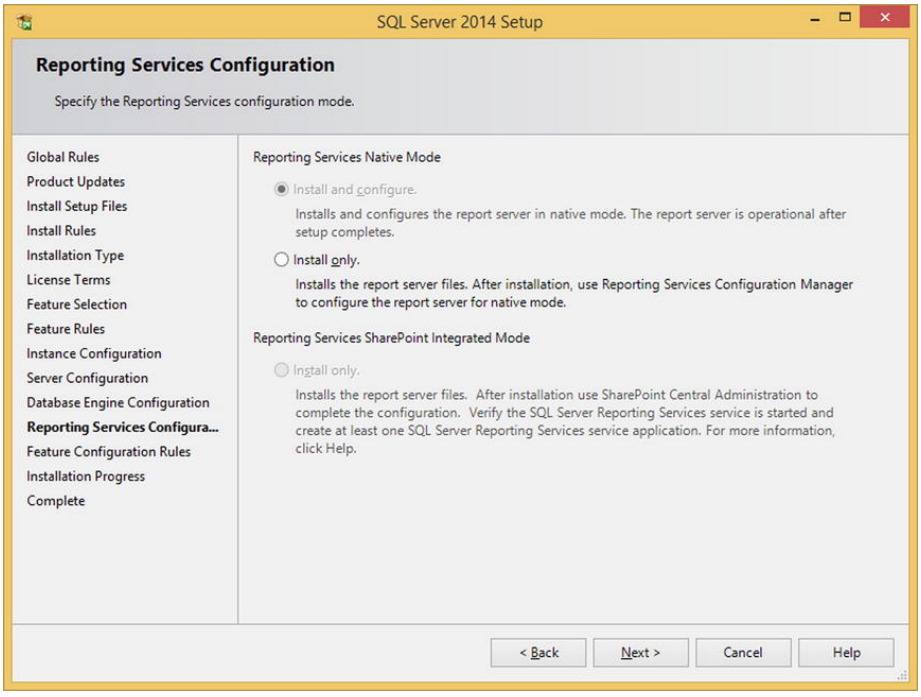

Figure 2-16. *Configuring Reporting Services (SSRS)*

Leave the defaults selected and click **Next** to move on. The end is in sight!

Installing

After clicking **Next,** SQL Server Express will finally(!) begin to install. Put your feet up or make a cup of tea while you wait. Be warned—this can take a while!

Once installation is finished, you'll see a nice confirmation message, just like the one in Figure 2-17.

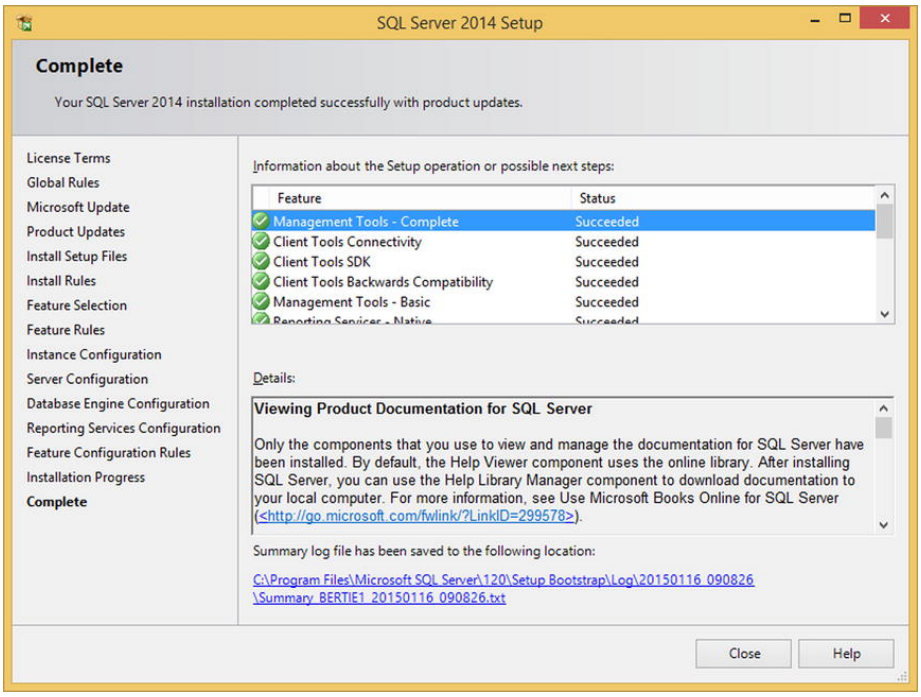

Figure 2-17. *Installation complete*

Click **Close** to close the installer. The SQL Server Installation Center will still be open, so you'll need to close that separately. All we need to do now is check that the SQL Server instance is ready to use.

Go to your Start Menu or Start Screen, and type *SQL Server 2014 Management Studio*. Once you've found it, click it to open. You'll be told user settings are being configured, and then you'll see something like Figure 2-18.

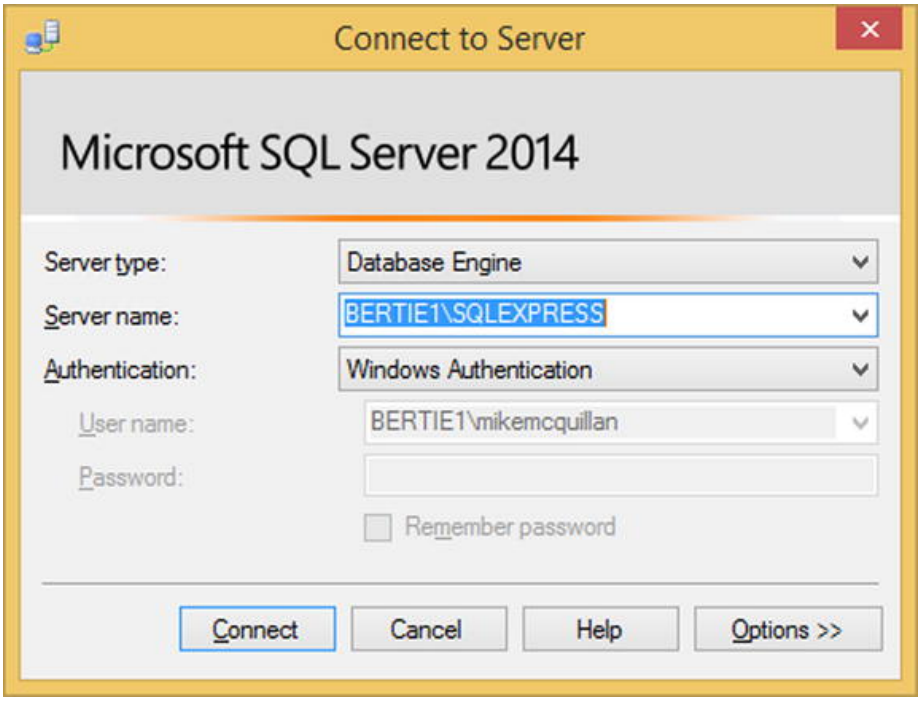

Figure 2-18. *The SQL Server login dialog*

This is the SQL Server Login screen. Click the **Connect** button and keep your eye on the Object Explorer on the left-hand side of the screen. You should see your server name appear with some items (such as Databases) below it, just like Figure 2-19.

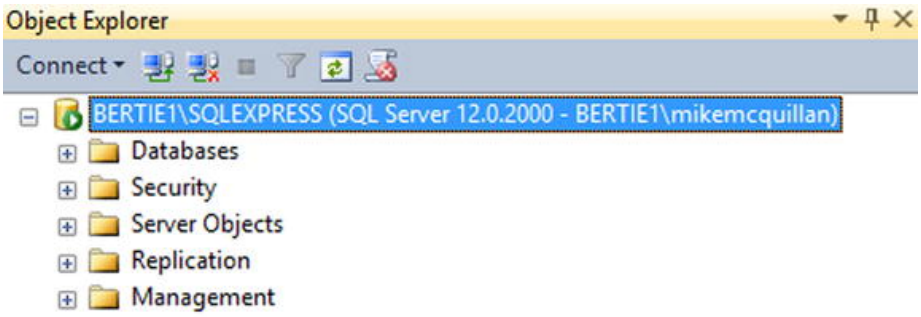

Figure 2-19. *The server, as displayed in the Object Explorer*

Congratulations, you now have a working SQL Server!

A Very Quick SSMS Overview

Before we start looking at SQL Server for real, it is worth a glance at some SSMS basics. We'll be spending pretty much the rest of the book in this tool.

Figure 2-20 shows that there are two sections to SSMS: the Object Explorer on the left, and a large unused area on the right. Object Explorer lists all of the server and database objects you can manage through SSMS. It uses a tree node system, where you have a parent and clicking the + sign shows you the children. Children of a parent node may have children themselves. You can drag the edge of the Object Explorer to make it larger or smaller as desired. I often change it so it slides in and out of view, by clicking the little pin next to the X in the Object Explorer's title bar.

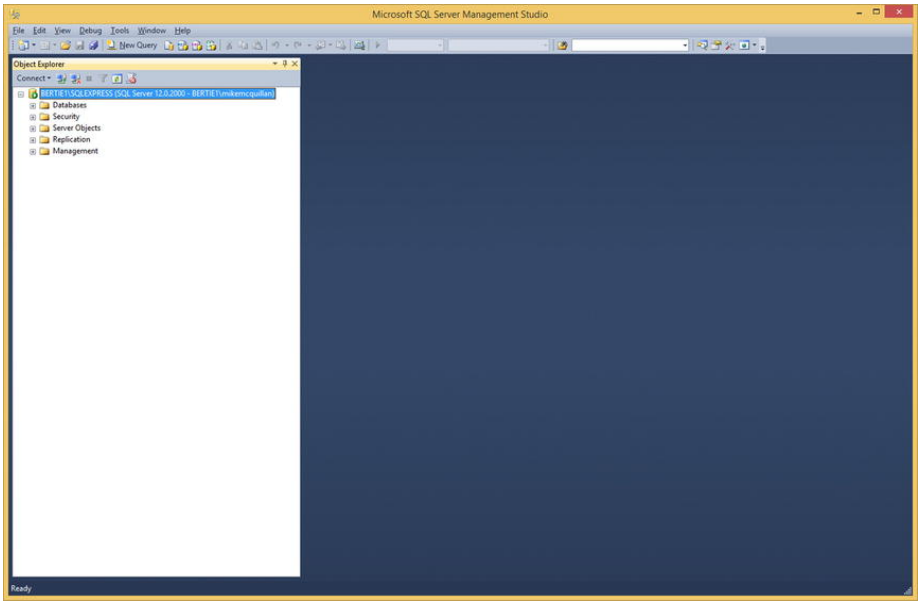

Figure 2-20. *A first view of SSMS*

There are five top-level items in Object Explorer:

- **Databases**: Lists all of your databases, including system databases. You can right-click this node to perform various database management tasks, such as restoring databases.
- **Security**: Supports management of user accounts and roles. Chapter 20 has more information about these.
- **Server Objects**: Shows server-wide objects, such as backup devices.
- **Replication**: Shows any replication jobs to which your SQL Server has subscribed (requested data from). We won't look at this; other versions of SQL Server have much stronger replication support.
- **Management**: Gives access to some management functions, like the ability to view your server logs.

The large empty area is used to display appropriate windows. You'll generally use it to write T-SQL queries. T-SQL, or Transact-SQL, is the dialect of the Structured Query Language developed by Microsoft and Sybase I originally referenced in Chapter 1. We'll see a lot more of T-SQL as the book progresses. Click **File ➤ New ➤ Query With Current Connection** (or just press Ctrl+N), and as Figure 2-21 demonstrates, a blank query window will appear, and most likely the Properties window will appear, too (press F4 on your keyboard if this doesn't appear on the right).

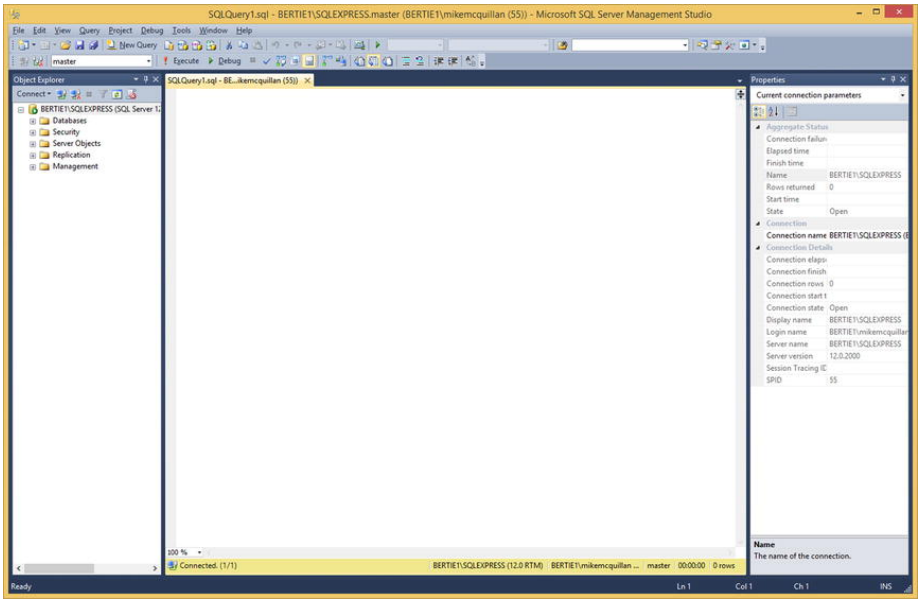

Figure 2-21. *SSMS showing query and properties windows*

I seldom use the Properties window, but that isn't to say it's no use. It lists various things about the connection you currently have to your SQL Server (you started a connection when you logged in). As you run queries, it will list the number of rows returned and the time it took to execute the query.

The main query window is where you will write your SQL code. If you type this statement in:

```
SELECT * FROM sys.databases;
```

you'll see a list of the databases available in the system. Figure 2-22 shows my list.

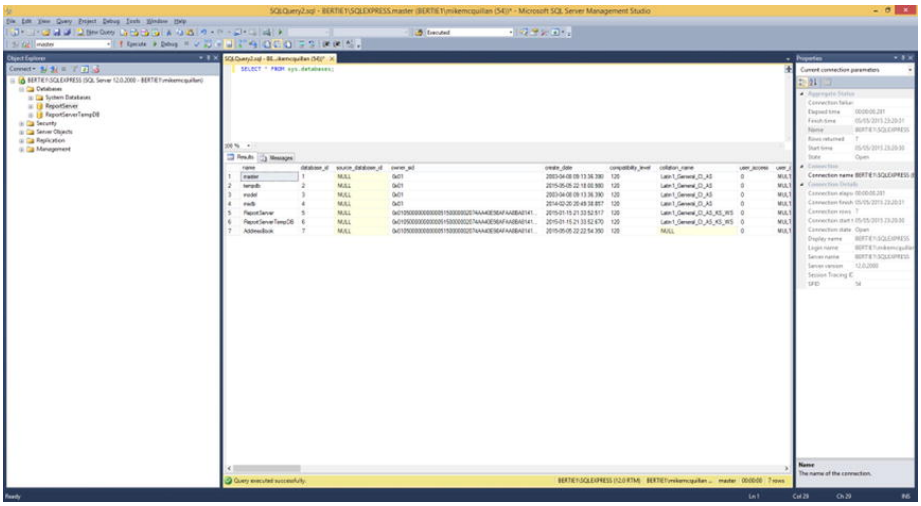

Figure 2-22. *A first query in SSMS*

Have a look at the menus and the toolbar options. SSMS is very customizable, so once you're a little more experienced with it you'll be able to configure it as you like it. One setting that makes development a little easier is to enable line numbers. This is done by navigating to **Tools ➤ Options ➤ Text Editor ➤ All Languages** and selecting **Line numbers**, then clicking **OK**. When an error occurs in a query, a line number is referenced. Enabling this feature makes it much easier to find the line causing the error.

Summary

We're done with the setup—now we're all ready to go and start developing. We'll kick off immediately in the next chapter by looking at how to create databases.

Let's go!

PREV
Chapter 1 : What Is SQL Server?

NEXT
Chapter 3 : Database Basics