

CHAPTER 48

SQL Server 2014 Reporting Services

This chapter introduces SQL Server Reporting Services 2014 (SSRS). It provides an overview of the product's features and architecture, covers the most important functional areas, and gives examples of how to get started developing on the platform. Although this chapter is by no means a comprehensive description of everything in SSRS, its purpose is to get you excited about the capabilities of this potent technology.

SSRS is an enterprise-class reporting platform that provides all the tools and services needed to support data-driven reporting. SSRS empowers you to design reports using a variety of advanced data visualization controls; populate them with data culled from a variety of sources; deploy, secure access to, and schedule execution of reports; export them to a variety of formats; and deliver them to the Web, SharePoint, email recipients, file shares, and more, on a scheduled or on-demand basis.

What's New in SSRS 2014

SQL Server 2014 includes very few updates to the SSRS platform, although there were many in SQL Server 2012 SP1. The enhancements in SSRS for version 2014 are:

- ▶ New versions of SQL Server Data Tools (SSDT) for Visual Studio 2012 and 2013, available for download at: <http://www.microsoft.com/en-us/download/details.aspx?id=36843> and <http://www.microsoft.com/en-us/download/details.aspx?id=42313>
- ▶ Google Chrome browser support for Report Manager and Report Viewer

IN THIS CHAPTER

- ▶ What's New in SSRS 2014
- ▶ Reporting Services Architecture
- ▶ Installing and Configuring SSRS
- ▶ Developing Reports
- ▶ Management and Security
- ▶ Performance and Monitoring

Notable enhancements that came with SQL Server 2012 include:

- ▶ An updated rendering extension for Microsoft Excel 2007–2010, supported both by Report Builder and the SSRS web rendering controls (both covered in this chapter). The maximum number of rows you can export from an SSRS report to an Excel worksheet is now 1,048,576, the maximum number of columns you can export is now 16,384, and the maximum number of colors supported for generated content is 16 million. The export format is Open Office XML (OOXML), and files are generated as zip-compressed *.xlsx archives.
- ▶ An updated rendering extension for Microsoft Word 2007–2010 (and 2003). It also supports the OOXML standard, generating zip-compressed *.docx files. Note that in order to open *.docx and *.xlsx files, Microsoft Office clients must be at version 2007 or later, or you must install the Office Compatibility Pack (available free from Microsoft's download site).
- ▶ Although the Visual Studio (VS) shell is still the primary integrated development environment (IDE) for SSRS projects, it was renamed to SQL Server Data Tools (SSDT). When you open older SQL Server projects in SSDT, they will be upgraded to the latest project file format. SSRS functionality in SSDT has been updated to support SQL Server 2014 data sources and deployments. The standalone Report Builder (RB3) IDE has also been updated to support to SQL Server 2014. No other RB3 features have been introduced (or removed).
- ▶ The ability to view web-based reports on touchscreen Apple iOS devices (using Safari, though only through direct URLs, not via Report Manager) as well as on Microsoft Surface-based tablet devices. This includes support for tablet-specific gestures, such as swiping to navigate a report, pinching or stretching to control zoom level, tapping to move along pages, and more. (Unfortunately, report export formats on iOS are limited to PDF and TIFF.)

The larger-scale enhancements that came with SSRS 2012 are only available in SharePoint-integrated mode (not covered in this chapter). They include the following:

- ▶ An updated version of the report server components required for integrating SSRS with SharePoint (SP) version 2013.
- ▶ Power View, new to the SSRS/SP family, enables advanced users to build ad hoc reports based on existing Excel PowerPivot data sources or Semantic BI models deployed to SQL Server Analysis Services (SSAS) 2014 (covered in Chapter 46, “SQL Server 2014 Analysis Services”). Once generated and published, you can view these reports using specialized SP pages that include a rich set of UI manipulation settings for on-the-fly modifications.
- ▶ Data Alerts, which provide a way for SSRS to notify interested parties when user-defined conditions about report data are satisfied. You design alerts using a SP UI tool (Data Alert Designer), save your conditions in reusable data alert definitions, schedule when these definitions run, and control the recipient list.

Discontinued Functionality and Breaking Changes

With SSRS 2014, a number of features have been discontinued or deprecated, including the following:

- ▶ SSDT does not include the capability to create or edit Report Models or even to open Report Model projects. You can still create reports (using SSDT or Report Builder) that rely on existing models, but this is deprecated. Microsoft recommends updating existing model-dependent reports to use other data sources because support for models will likely be discontinued in the next SSRS release.
- ▶ Reports built with the Report Definition Language (RDL) version included in SQL Server 2005 (and earlier) are deprecated, but you can add such reports to an SSDT project, and they will be automatically upgraded.
- ▶ Snapshots, custom report items, and some outdated HTML rendering extension settings (for example, those related to ActiveX and Flash) formerly available in SQL Server 2005 (and earlier) are deprecated.
- ▶ In the ReportServer2010 Web service endpoint, the methods `GetProperties(String, Property[])` and `IsSSLRequired()` are deprecated.
- ▶ The SharePoint Report Explorer and Report Viewer web parts are deprecated.
- ▶ SP-integrated mode is no longer configurable using Reporting Services Configuration Manager (RSCM) or the SSRS Windows Management Instrumentation (WMI) provider (via scripts you may have written for the `rs.exe` tool [covered later in this chapter]). You must use SP Central Administration and the SP PowerShell console. In addition, to view SP reports, you must navigate to the full SP site URL rather than the Report Manager base URL.

Reporting Services Architecture

When referring to SSRS as a platform, we are actually talking about a cohesive set of development tools, configuration tools, web services, applications, and utilities, all working together to deliver enterprise-grade reporting.

In a nutshell, the platform includes the following components:

- ▶ A single Windows service, listed in the Windows Service Control applet as SQL Server Reporting Services, which acts as a host for and provides centralized control of SSRS's background processing engine, web services, and Report Manager web application. It also handles encryption and decryption of stored credentials and connection information.
- ▶ Two databases, known as the Report Server catalogs (note that the following are their default names; you can name them whatever you want using the Reporting Services Configuration Manager, or RSCM):

- ▶ **ReportServer**—Stores all reporting objects, including reports, security settings, schedules, subscriptions, snapshots, users, configuration settings, and encryption keys.
- ▶ **ReportServerTempDB**—Stores ephemeral report data (sometimes called *intermediate processing products*), such as cached reports, session and execution data.
- ▶ Four .NET web services, which serve as SSRS's programmatic APIs:
 - ▶ **ReportService2005.asmx**—Provides methods for managing all aspects of an SSRS instance configured in native mode (deprecated).
 - ▶ **ReportService2006.asmx**—Provides methods for managing all aspects of an SSRS instance configured in SharePoint-integrated mode (deprecated).
 - ▶ **ReportService2010.asmx**—Subsumes functionality of ReportService2005.asmx and ReportService2006.asmx.
 - ▶ **ReportExecution2005.asmx**—Provides methods for custom report rendering and execution.
- ▶ Three command-line applications, all located in either %PROGRAMFILES%\Microsoft SQL Server\120\Tools\Binn or %PROGRAMFILES(86)%\Microsoft SQL Server\120\Tools\Binn:
 - ▶ **RSKeyMgmt.exe**—Provides encryption management for securing database-stored Report Server content, such as credentials, connection strings, accounts, and the encryption key itself. This tool is also used to join servers in an SSRS farm configuration (via the -j option).
 - ▶ **RS.exe**—Enables developers to write scripts in VB.NET that leverage the web service APIs.
 - ▶ **RSConfig.exe**—Enables you to programmatically change SSRS configuration values in RSReportServer.config (the configuration file for the web service APIs), either on a single or multiple machines.
- ▶ Report Manager, an administrative website that provides web-based control over SSRS, including the ability to
 - ▶ Add or remove, organize, configure, and run all kinds of SSRS objects, including the following:
 - ▶ Reports, report resources, data sources, shared datasets, report parts, and folders.
 - ▶ Report models and data source views (used with Report Builder).
 - ▶ Administer the SSRS security model, including the following:
 - ▶ Users and roles.
 - ▶ Role assignments (remember to keep these simple).

- ▶ Manage the following:
 - ▶ Report snapshot, history, and caching configuration.
 - ▶ Schedules, subscriptions, and related settings (Note: SQL Agent must be enabled for automated report execution).
 - ▶ Report execution timeout duration.
- ▶ Reporting Services Configuration Manager (RSCM), a configuration GUI application (covered in detail in the following section).
- ▶ A suite of SharePoint Web parts, pages, and documentation.
- ▶ Report Builder, a ClickOnce application for designing and executing ad hoc reports.
- ▶ SSDT, which includes Report Designer; specialized tool windows; and other capabilities for report development, testing, and deployment.
- ▶ A Report Customization Extension (RCE), enabling developers to alter the RDL stream on-the-fly. This feature is implemented as a processing event into which you can wire your custom RDC code to change the layout, language, and so on. (This topic is not covered in this chapter.)
- ▶ Two Microsoft .NET Report Viewer controls (one for ASP.NET, one for Windows Forms), for integrating reporting in custom applications. Report Viewer offers a rich programming interface for controlling report execution and interactivity and is available for C#, VB.NET, and the other .NET languages.
 - ▶ The Report Viewer control is capable of processing SSRS reports using two modes:
 - ▶ **Local**—Using this mode, report processing happens in your application, meaning that SSRS is not required to run your application's reports.
 - ▶ **Remote**—Using this mode, report processing happens via the Report Server web services.
 - ▶ A Windows Management Instrumentation (WMI) provider, which exposes a set of WMI interfaces that programmers can use to configure the Report Server or build other configuration utilities.

Figure 48.1 provides a tiered view of the SSRS architecture, illustrating each platform component.

HTTP Architecture

Although SSRS does a lot of web-related tasks on Windows Server, keep in mind that it doesn't rely on Internet Information Services (IIS) at all. There are no IIS-based websites or applications to configure. Any ISAPI extensions, virtual directory settings, or other advanced web customizations you may have built are probably not compatible with the SSRS HTTP architecture.

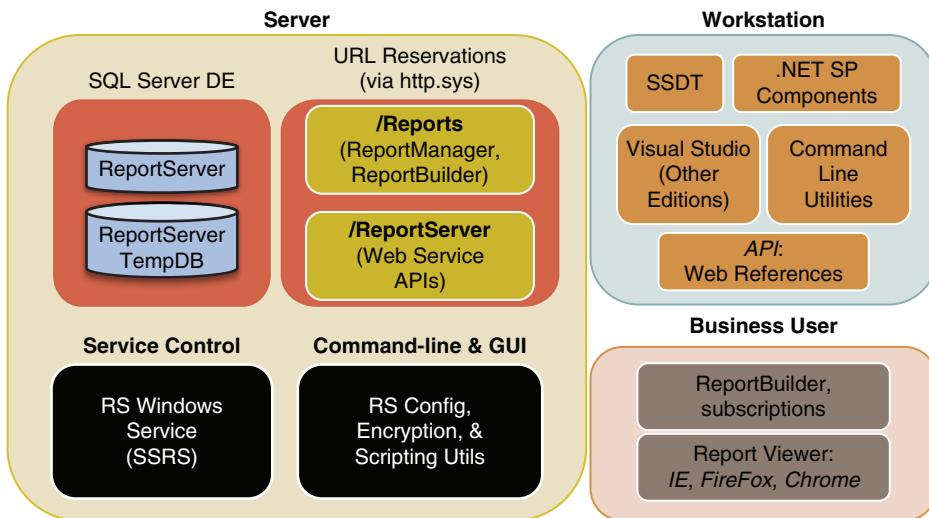


FIGURE 48.1 SSRS tiered architecture diagram.

To process HTTP requests for SSRS web services, Report Manager ASPX pages, and other calls, SSRS 2014 natively hosts the .NET Framework and ASP.NET. It ties directly in with operating system level (or kernel mode) HTTP, listening for requests by way of the Windows HTTP API (sometimes referred to as http.sys). This means that under the covers, SSRS registers its virtual paths (also known as URIs; for example, www.myserver.com/Reports) with http.sys in the same way that IIS would register a virtual directory. The operating system redirects incoming HTTP requests to IIS or SSRS, based on the path specified in the request.

The Report Server Service stops and starts all aspects of SSRS, including Report Manager, the web services, and the background processing engine (a.k.a., the *scheduling and delivery processor*). SSRS also includes a native authentication layer (as opposed to using IIS for authentication), as well as native HTTP logging and server memory usage configuration.

This HTTP architecture brings with it one HTTP port-related gotcha:

- When you install SSRS, port 80 is used in the standard native configuration. This means that when configuring your SSRS URIs using RSCM you must be sure to use paths and ports that are not already in use in any IIS site or virtual directory (to avoid unintended collisions).

TIP

If you try to use a port already reserved by a website or other network service, RSCM usually warns you. However, if that website or server is configured but in the stopped state, RSCM may allow you to use its port. This could cause a resource conflict when that website or service is started again.

In the next section, you learn how to install SSRS, where to find each installed component on your file system, and how to use RSCM to optimally configure your installation of SSRS. After configuring SSRS, we move on to report development with SSDT and Report Builder.

Installing and Configuring SSRS

When you launch the SQL Server installer, the setup application checks the prerequisites on your system to determine whether SSRS can be installed without issue. If your system meets all the requirements, you may proceed, following the steps described in the following sections.

The Installation Sequence

To install SSRS, you need to run the SQL Server installer and be sure to check the Reporting Services - Native check box on the Feature Selection installation step, as shown in Figure 48.2. (Note that the figures in this section show the installation screens as they appear when creating a new SQL Server instance.)

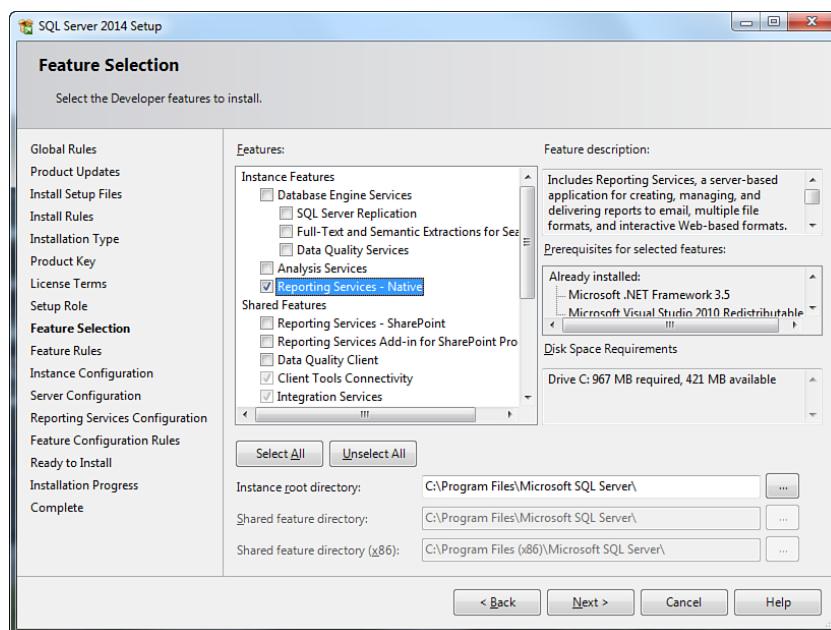


FIGURE 48.2 Selecting SSRS on the Feature Selection installation step.

Next, on the Server Configuration step (illustrated in Figure 48.3), select an account to use for the SSRS Windows service. You can use the suggested built-in account that SSRS will create (`NT Service\ReportServer$ [InstanceName]`) or create and use a dedicated account instead (recommended).

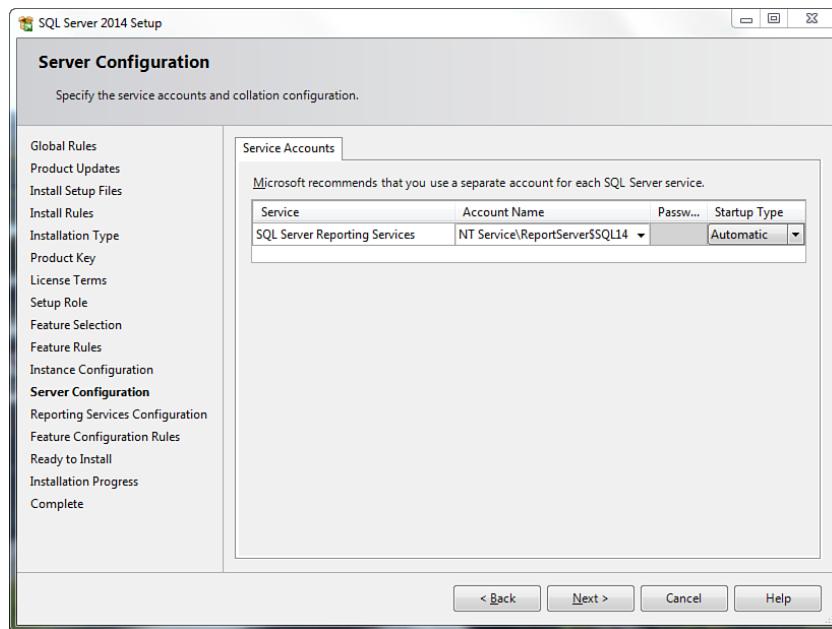


FIGURE 48.3 Configuring the SSRS Windows service account on the Server Configuration installation step.

When you reach the Reporting Services Configuration mode selection step (illustrated in Figure 48.4), you have up to three configuration options, depending on your server's configuration:

- ▶ The installer detects if SharePoint is running on the target server. If it is, you have the option of installing SSRS in SharePoint-integrated mode (in the previous step, otherwise this option is grayed out) and having it automatically configured with the default settings.

TIP

Accepting the default installation settings means that setup creates and sets up the database catalogs, configures the web service URLs, installs all needed files and Registry settings, and sets up all necessary security settings and permissions. The only configuration option you have in this scenario is selecting the Windows service account.

-
- ▶ You can install SSRS in native mode and have it configured with the default settings
 - ▶ You can install SSRS but not configure it. For this example, you should make this selection because the next learning task is how to use RSCM to configure SSRS.

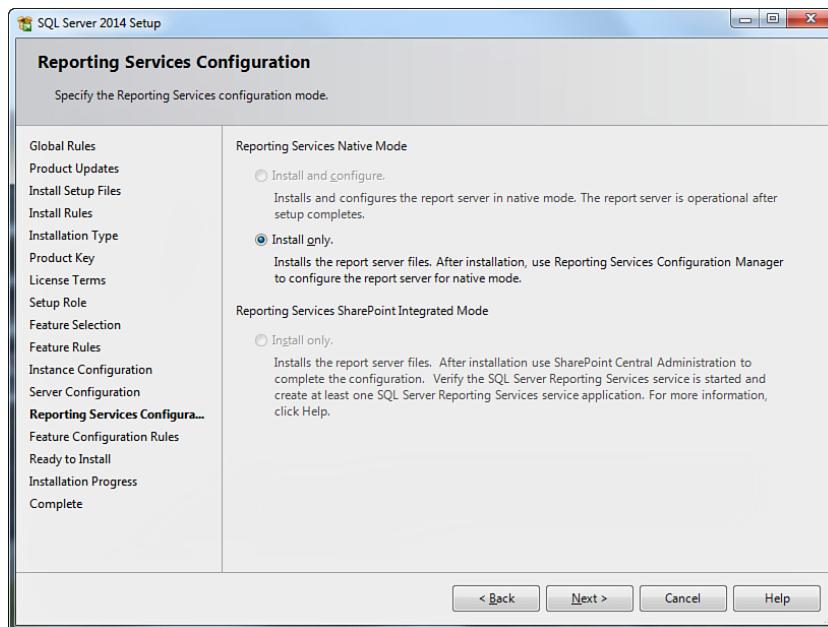


FIGURE 48.4 Choosing an SSRS configuration mode during installation.

In this scenario, Setup copies all necessary files, creates the appropriate Registry settings, and sets up the Windows service, but otherwise leaves the Report Server unconfigured.

In the final step, Ready to Install, the installer shows the name of the new SSRS instance (you need this later to locate the installed files) and the configuration mode selected (illustrated in Figure 48.5). In this example, the installer reports that the SSRS install is being performed in `FilesOnlyMode`, meaning that you must use RSCM to completely configure SSRS after installation completes (or you won't have a working platform).

Assuming all went well, your next task is to open Windows Explorer and navigate to your install location to examine what's on the file system. This might seem like a trivial exercise, but in times of immediate need, it's critical to know where things live.

File Locations

The root folder you should care about most is `%PROGRAMFILES%\Microsoft SQL Server\MSRS12.[InstanceName]\Reporting Services`.

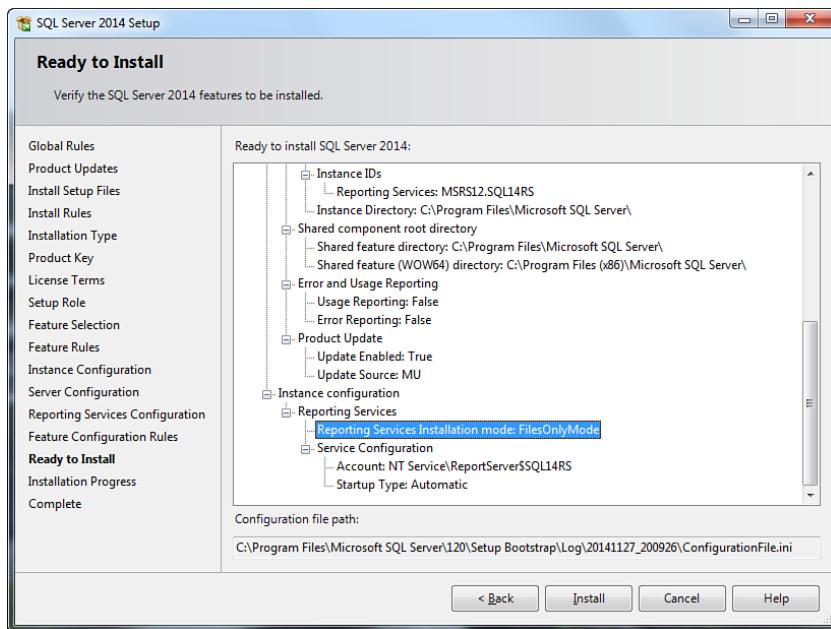


FIGURE 48.5 SSRS installation, final step.

Below this folder, you find all the items listed in Table 48.1 in their respective locations.

TABLE 48.1 SSRS Folder Content

SSRS Item	Installation Subfolder
Log files (ReportServerService_[timestamp].log, the primary log file)	LogFiles
Report Manager (SSRS's administrative website)	ReportManager
Location of cascading style sheets (use them to tweak the look of the Report Manager website)	ReportManager\Styles
Web service APIs (and associated configuration files)	ReportServer
Windows service (and associated configuration file)	ReportServer\Bin
Report Builder 3.0	ReportServer\Report Builder
Command-line utilities	Found in %PROGRAMFILES(x86)%\Microsoft SQL Server\120\Tools\Binn
SharePoint web parts	Found in %PROGRAMFILES(x86)%\Microsoft SQL Server\120\Tools\Reporting Services\SharePoint

SSRS Configuration Using RSCM

RSCM is a comprehensive configuration tool that enables you to perform the following platform tasks:

- ▶ Set up `http.sys` URL reservations (for Report Manager and the reporting web services).
- ▶ Create the SSRS databases (`ReportServer` and `ReportServerTempDB`).
- ▶ Generate and back up SSRS's symmetric encryption keys (used for encrypting sensitive data stored in the SSRS databases).
- ▶ Configure the SMTP account settings used for scheduled report delivery.
- ▶ Configure the unattended execution account used by report data sources that don't require authentication (such as images, XML files, and so on). Note that images are retrieved from the server only when the report page containing them is rendered on demand (except in the case of snapshot creation, when they are all retrieved up front).
- ▶ Configure multiserver scale-out (for building SSRS web farms that share a common SSRS catalog).
- ▶ Start and stop the SSRS Windows service.
- ▶ Change the SSRS Windows service account.
- ▶ View SSRS version information for your instances.

You can use RSCM at the end of a custom installation or at any time to change the platform settings. When working with RSCM, you navigate the tree displayed on the left of the GUI from top to bottom, from task to task, configuring all appropriate settings, as shown in Figure 48.6.

Next, let's walk through a typical configuration scenario using RSCM. This step is necessary because in the installation example you ask the installer not to configure SSRS.

The first step is to launch the program, located either on the Start screen (when using Windows Server 2012) or in your Programs menu (on Windows Server 2008 R2 and earlier) under Microsoft SQL Server 2014\Configuration Tools. When RSCM starts, it prompts you for an SSRS instance to which to connect.

Windows Service Configuration with RSCM

After you connect to your instance, notice the configuration choices available on the left side of the main window. Click your SSRS instance name (at top left) and ensure that your SSRS Windows service is running. Keep in mind that the Report Server Windows service is an essential Report Server component. It needs to be running for reports to be executed either on demand or offline. You can change its service account and/or password by clicking the `Service Account` node.

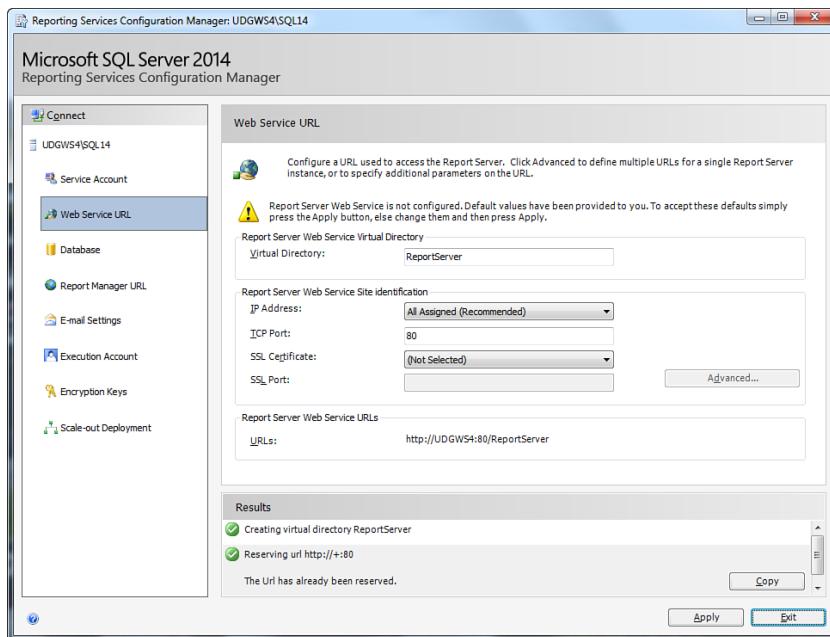


FIGURE 48.6 Configuring the SSRS web service URL with RSCM.

Web Service Configuration with RSCM

The Report Server web service exposes the Simple Object Access Protocol (SOAP) interfaces clients use to interact with the platform.

Using the tree on the left of the screen, click on Web Service URL. On the detail pane (located on the right side of the main window), under Report Server Web Service Virtual Directory, enter ReportServer (or similar) in the Virtual Directory text box (if it is not already present). This is the directory name users have come to expect.

In the next group box, select an IP address, port, and optional SSL certificate and SSL port. Save your configuration changes by clicking the Apply button (at bottom right). When the settings are applied successfully, your window should look something like the one in Figure 48.6.

After completing the remaining RSCM steps, click the link located under Report Server Web Service URLs to test your new virtual path. Keep in mind that later, when you begin developing reports with SSDT, you need to enter this service URL on your Report Server project's properties to be used as your deployment path (covered later in the section, "Deploying Reports").

The setting changes you make in this area of RSCM are saved to the following file:

```
%PROGRAMFILES%\Reporting Services\ReportServer\rsreportserver.config
```

These settings are saved to an XML node that you can locate in the configuration file via the following XPath:

```
\Configuration\UrlReservations\Application[Name='ReportServerWebService']
```

Database Configuration with RSCM

As mentioned previously, SSRS relies on two databases: the main store for metadata (named `ReportServer` by default) and a temporary store for user sessions (named `ReportServerTempDB`). `ReportServerTempDB` is created in simple recovery mode and doesn't need to be backed up periodically because it contains only transient data—data about the in-flight sessions being actively served by SSRS. `ReportServer` is created in full recovery mode; it houses all the critical components of your reporting system. SSRS cannot run without it.

Click on the Database node in the tree on the left. On the detail pane, click the Change Database button. On the ensuing Change Database dialog, select the radio button labeled Create a New Report Server Database. Next, select your target SQL Server instance on the Database Server dialog. On the next step (Database), enter `ReportServer` as your database name and leave the Native Mode radio button selected. Click Next.

On the Credentials screen, select the SQL Server instance (local or remote) and login credentials you want you use for the database account. This account is granted the SQL Server role `RSEexecRole` on both SSRS databases. This role is critical because it contains all the permissions necessary for report administration. For convenience, you can make this the same account you selected for the Report Server service (illustrated in Figure 48.7).

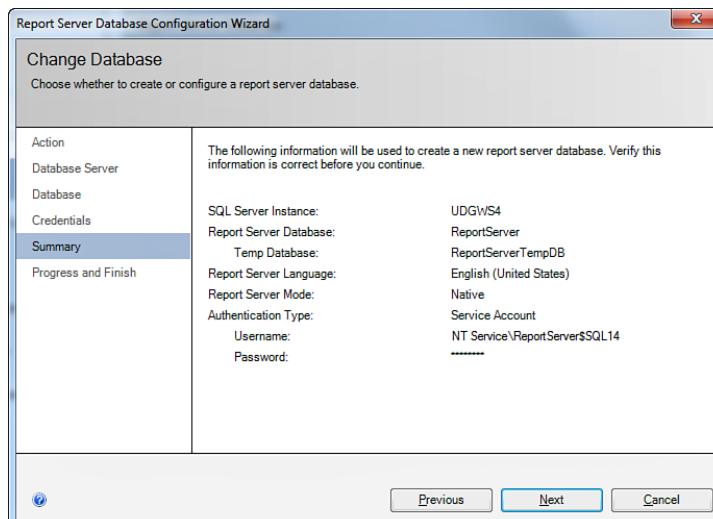


FIGURE 48.7 Configuring the SSRS databases using RSCM.

When this process completes, open SSMS and verify that your new databases are present on your target instance of SQL Server. Next, in Object Explorer navigate to the Security\Users node for each database. Check the properties of the user you specified in your database credentials and ensure that the user is a member of the RSEexecRole.

Report Manager Configuration with RSCM

Click the Report Manager URL node in the tree on the left. In the Virtual Directory text box, enter Reports or something similar (Reports is the default name for the Report Manager virtual path). Click Apply and then click the URL link to test that Report Manager is working properly. Your browser should open to Report Manager, looking something like the window shown in Figure 48.8 (depending on what works for your particular system configuration). If you receive an access denied or related security-related error message, be sure to run Internet Explorer as Administrator. You may also need to temporarily disable User Account Control (see this Microsoft Connect article: <https://connect.microsoft.com/SQLServer/feedback/details/622737/user-does-not-have-required-permissions-verify-that-sufficient-permissions-have-been-granted-and-windows-user-account-control-uac-restrictions-have-been-addressed>).

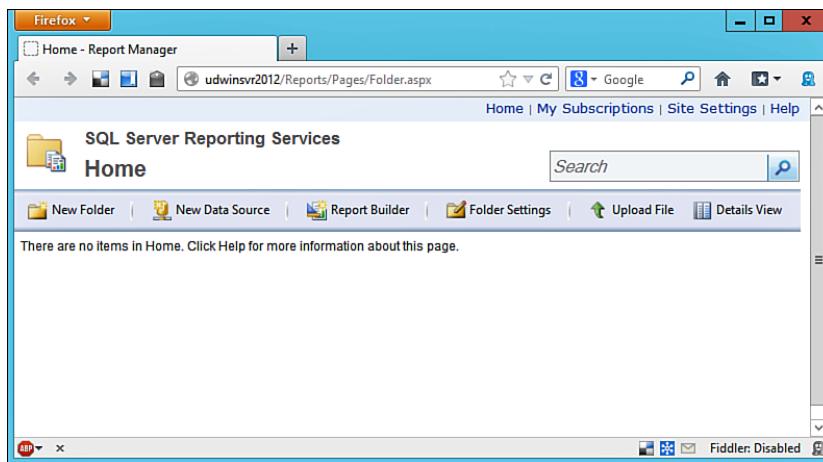


FIGURE 48.8 Viewing Report Manager in Firefox

Your web settings are saved to `rsreportserver.config` in an XML node you can locate via the following XPath:

```
\Configuration\UrlReservations\Application[Name='ReportManager']
```

Email and Execution Account Configuration with RSCM

If you plan to deliver reports over SMTP (something your users will really appreciate), click the `Email Settings` node in the RSCM tree and enter your mail server account settings. If you plan to use externally stored images or XML data sources in your reports, click on

Execution Account in the RSCM tree and enter credentials for an account that has access to those file systems.

Encryption Configuration with RSCM

As mentioned earlier, SSRS is capable of securely storing sensitive information (for example, connection strings to data sources for reports, subscription information) in the SSRS catalogs. To be able to do so, it uses the Windows Cryptographic APIs, which are based on the account under which the service is configured to run.

When the service is first started and it connects to an empty Report Server database, it creates a symmetric key used for encryption. It then encrypts this symmetric key with the public key of the account used to run Report Server Windows services. It then stores the encrypted keys in the SSRS catalog and uses the keys to encrypt and decrypt data. You can also use this RSCM screen to manually delete encrypted content, change the encryption key itself, and restore an existing key.

It's a good idea to make a password-protected backup file of this encryption key. To do this, click on the `Encryption Keys` tree node and then click `Backup`. Select a file location and enter your password. Now, if anything should go wrong with your SSRS installation, you can still decrypt your encrypted data. This capability is quite important because if you lose your key, there is no way to retrieve it again, and all your encrypted data is rendered inaccessible.

You should always change the service account under which the Report Server service runs via RSCM because, when you do this, the system needs to back up and restore the encryption keys as well as make sure the new account has access to the Report Server database. This explains why you are prompted to save the encryption key when you perform this operation.

Scale-Out Architecture Configuration with RSCM

The final RSCM screen to discuss is the Scale-out Deployment screen. If you plan to deploy the SSRS runtime components to a number of servers but use only a single SQL Server instance for SSRS data storage, you're ready to scale out (although this does require SQL Server Enterprise Edition). You use this screen to join or unjoin servers to and from your web farm (you can also use the `RSKeyMgmt.exe` command-line utility for this purpose).

For a list of the available features in each SSRS edition, see “Compare Edition Features” at <http://www.microsoft.com/en-us/server-cloud/products/sql-server-editions/>.

Developing Reports

Now that you understand the SSRS architecture and your instance of SSRS is properly installed and configured, you're ready to dive into report development.

Tools of the Trade

SSRS 2014 provides two primary design tools for building reports and related objects:

- ▶ SSDT, a powerful development tool integrated with VS 2013
- ▶ Report Builder, a simpler-to-use yet no-less-powerful application for designing ad hoc reports

Both tools provide rich graphical design surfaces and allow for a (mostly) WYSIWYG experience, and you can achieve almost all the same results with either. SSDT, however, is marketed heavily toward developers, whereas Report Builder is marketed more at advanced business users.

In practice, Report Builder users usually depend on having at least one hard-core SSDT developer to lean on, not only for questions on how to work with the platform, but also to prepare reports, report data sources, and report datasets, and to maintain the management system, in order to succeed. As a developer, you really need to master both tools because your end users will almost certainly bring their Report Builder questions to you.

Report Basics

What is a report? A report is a means of visualizing data derived from one or more sources. Typically, these sources have always been datasets coming from T-SQL statements, stored procedures, or web services. But with the advent of the mapping features in SSRS (covered in this chapter in the section, “Working with Maps”), reports can also rely on information stored in ESRI (geospatial) data files.

A report may

- ▶ Display data-bound and non-data-bound controls, offering static and interactive views on the data
- ▶ Include a header, footer, table of contents (called a *document map*), links, images, and linear art (lines and rectangles)
- ▶ Make use of various layouts, styles, and file formats
- ▶ Include sorting, filtering, and grouping functionality (on row data)
- ▶ Accommodate input parameters, whose values are passed from user or programmatic input to your report’s queries
- ▶ Reference embedded or externally stored credentials and data sources

All reports are internally described by RDL, an Extensible Markup Language (XML)-based dialect understood by a variety of design tools available from Microsoft as well as a select few third parties. (The report file extension is .rdl.)

RDL is a content model describing the report layout, formatting, and instructions on how to fetch the data. It may also contain custom code written in VB.NET that is executed during report rendering. You can write such code in two ways:

- ▶ Using the built-in functionality provided with SSRS's VB.NET-style expressions
- ▶ Referencing and calling methods against custom or core .NET Framework assembly classes

(Expressions are covered in this chapter in the section, “Understanding Expressions.”)

Keep in mind that you do not need to learn the RDL dialect to develop reports. It becomes important only when you need to generate or manipulate the markup directly—for example, when generating your own RDL files from an XML source using XSL for Transformations (XSLT) or when developing a custom rendering extension.

RDL supports all the controls in the SSDT Toolbox window. It also enables you to control page naming, page breaks, pagination (which can be restarted within a page range), margins, header and footer visibility, and null value handling.

Overview of the Report Development Process

Generally speaking, report development follows four or five phases (illustrated in Figure 48.9):

1. Preparing your data sources and datasets for use with SSDT and/or Report Builder
2. Designing your report using SSDT or Report Builder—that is, laying out your visual controls and wiring up the datasets
3. Deploying your reports, report parts, data sources, and shared datasets to the ReportServer catalog, where they are stored (SSDT and Report Builder both provide this function)
4. Testing your reports using a supported web browser (Firefox, Internet Explorer, Chrome, or Safari) via Report Manager or your development tool of choice
5. (Optionally) Securing your reports and setting up data caching rules and a delivery schedule (both covered later in this chapter)

Data Planning and Preparation

The first step in report development is to prepare your data for use with SSRS. Generally speaking, when you are working with non-file data, this means creating the T-SQL tables, views, procedures, and functions from which your data sources will retrieve rows. Any complex logic required to get to report data should happen within the Database Engine, not within SSRS.

The rule of thumb is to keep complex logic and intricate calculations out of your reports and to prepare them ahead of time in your sources. A good reason to put this policy into practice is that, from a maintenance perspective, it will be much easier for you and your colleagues to modify a report if all that is necessary is to change the underlying T-SQL. You don't want to bury your business logic in extensive RDL expressions and embedded code that will be difficult to find.

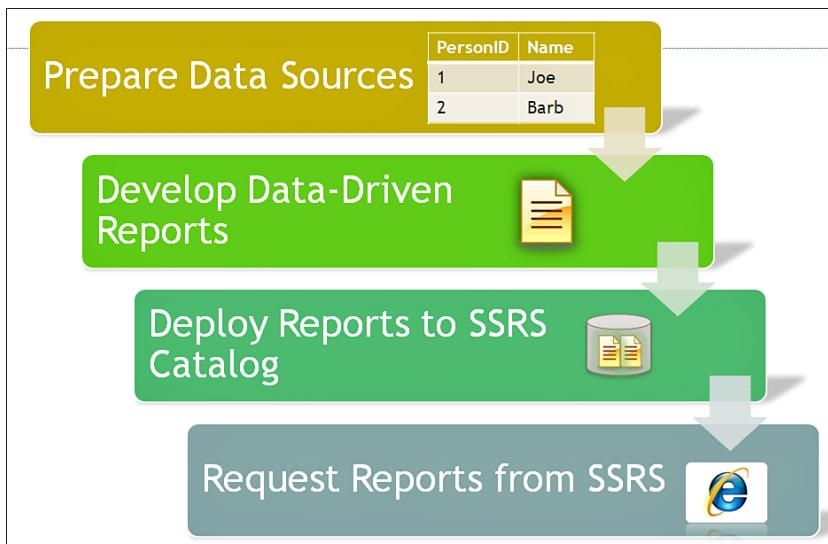


FIGURE 48.9 Phases of report development.

Another thought to keep in mind during the data planning phase is to ensure that your database server is capable of handling the increased data storage and execution loads that SSRS will bring. Plan and discuss this issue with your database and network administrators.

If your increased loads warrant it, consider the idea of dedicating a separate instance of SQL Server to SSRS storage and execution, or even a separate machine.

In addition, you might want to build or make use of content available in a data mart or data warehouse; doing so prevents your reports' execution from impacting the transactional performance of your online databases.

Using Shared Data Sources

Unless you use shared data sources, the data source for your report is embedded into its RDL. This means that when you want to change that data source, you must modify the report or, at the very least, change its data source properties using Report Manager. If you don't use shared data sources, each of your reports requires its own data source, and these data sources cannot be used by any other report.

It's truly a best practice to use shared data sources. When the time comes to make a connection change, you have to look in only one place. Think of how useful (and time-saving) this will be when you need to test your reports in a development environment and then run them against production. Save yourself the headache and start using shared data sources from the onset of your report development.

Using Datasets

Every meaningful report relies on at least one dataset. In SSRS terms, a *dataset* is simply an abstraction of some set of source data generated by a query and used within a report. Datasets remember the data structure of the queries whose output they contain (the fields, field names, field data types, collation, case sensitivity, and so on). Datasets also store the underlying query used to derive report data from a data source and are aware of any parameters needed to obtain that data—for example, in cases where the underlying data source is a stored procedure.

Every data-bound control on your report needs a dataset from which it will be populated at report (and query) execution time. In SSDT, datasets are listed in the Report Data window's tree listing, and, after you create a dataset, you can simply drag its fields from the tree onto the appropriate drop zones of your data-bound controls to create a link between the two. In simple terms, this means that for every row your query returns, an instance of that field is repeated in the data-bound control. This description is, of course, an oversimplification; you can slice and dice your data in many other ways, as you'll soon see.

Using Shared Datasets

Shared datasets further improve the decoupling of report data from reports. They also encourage reuse and accelerate report execution. A *shared dataset* is simply a dataset you define at design time that can then be reused by any number of reports. You may modify or delete a shared dataset independently of any reports or report parts that depend on it and vice versa. Using shared datasets prevents the need for re-creating the same dataset in multiple reports. Redundancy is the enemy of maintainable code, and using shared datasets prevents you from letting small differences in your underlying queries produce inconsistent results (something end users tend to intensely dislike).

Like regular datasets, shared datasets may make use of input parameters. At execution time, shared dataset output is cached according to unique combinations of parameter input (much like SQL Server stored procedures). This leads to efficiency gains at report execution time.

Like reports and other SSRS objects, shared datasets are deployed to the SSRS catalog during report project deployment. When they are published, you can manage your shared datasets using Report Manager. You can also modify or delete them using SSDT or Report Builder. Shared datasets are XML files stored with the .rds extension. You can take the .rds file created by SSDT and deploy it to other SSRS catalogs by uploading it with Report Manager.

Developing Reports Using SSDT

The first step is to create a Report Server Project. This SSRS-specific project type enables the development and organization of most report objects. Launch SSDT, and, using its main menu, click File, New, Project. In the New Project dialog, click the Business Intelligence Projects node in the tree at the left of the screen; then click Report Server Project under Visual Studio Installed Templates (shown in Figure 48.10).

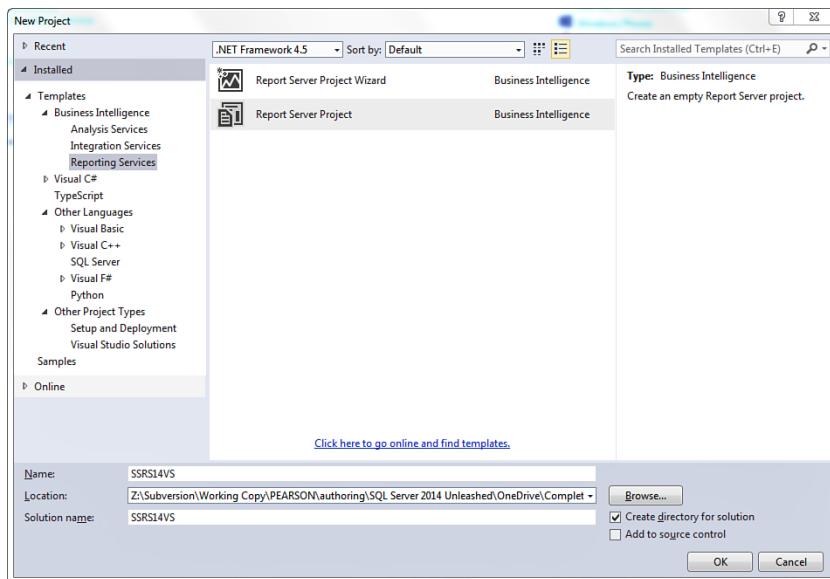


FIGURE 48.10 Creating a Report Server Project using SSDT.

Creating a Shared Dataset

After you've created your project, open your Solution Explorer window and create a shared data source by right-clicking the Shared Data Sources folder and then clicking Add New Data Source. This is your report's connection to the database from which it will retrieve report data. Configure the data source to point to your local instance of the AdventureWorks2012 database.

Next, right-click the Shared Datasets folder, and then click Add New Dataset. The Dataset Properties window appears. Select your new shared data source, and then, using Query Designer (or the Text window), design or type in a T-SQL query or EXEC statement (for running stored procedures) that will return at least a few rows.

Click the Fields tab on the left; here, you can create calculated fields (using expressions), or add, remove, rename, or provide a data type for the fields in your shared dataset. Using the left navigation tabs, you can also set various dataset options (such as case sensitivity) and add any necessary parameters or filters. When your dataset is set up as you like it, click OK.

The following sample query creates a simple shared dataset that takes one input parameter:

```
SELECT
    BusinessEntityID,
    PersonType,
    Title,
    FirstName,
```

```

LastName
FROM Person.Person
WHERE FirstName LIKE @FirstLetter + N'%'
ORDER BY FirstName

```

In the sample project (in the sample files and code listings folder for this book on the Web), you can find this shared dataset file named `sdsPeopleByLetter.rsd`.

The next step is to configure the deployment URL for your project. Right-click your project in Solution Explorer and then click Properties. In the General section, under the Deployment header, enter the path to your SSRS web service URL in the Target Server URL field and then click OK. In most cases, the URL you use will be your server's name followed by *ReportServer* (for example: `http://MyServerName/ReportServer`). You can refer back to RSCM to find your report server's URL under the Web Service URL section.

Next, right-click your shared dataset in Solution Explorer and click Deploy to publish it to the SSRS catalog. Once deployed, you can use your new shared dataset in any report. To use a shared dataset (when building a report with SSDT), you right-click the `Datasets` folder in the Report Data tool window and then click Add Dataset. On the ensuing Dataset Properties window, select the Use a Shared Dataset radio button, click the icon representing your shared dataset, and then click OK.

You can use Report Manager (covered later in this chapter in the section, "Using Report Manager") to manage your shared dataset: Move it to another folder or delete it, change its caching rules, alter its inherited permissions, switch its underlying data source, and, most importantly, view a list of all reports that depend on it. Figure 48.11 illustrates how to accomplish these tasks.

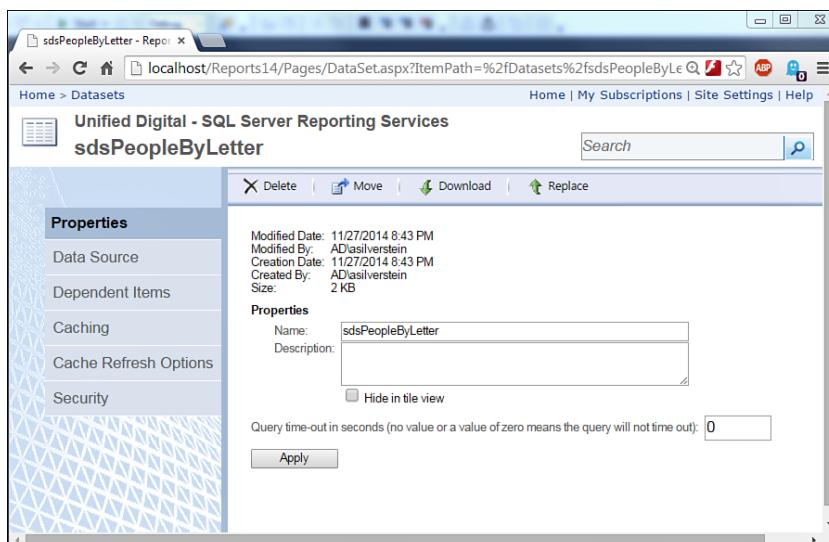


FIGURE 48.11 Managing a shared dataset using Report Manager.

Using the Report Wizard

To quickly get into report development, you will create a report using the Report Wizard. Right-click the `Reports` folder in Solution Explorer, then click Add New Report. This launches the Report Wizard, which leads you through all essential report-creation steps for building a simple report. If you want to skip the wizard and get directly to the design surface, click Add New Item and then select Report instead of selecting Add New Report.

The first step in the Report Wizard is to create a data source for your reports. Connect to `AdventureWorks2012`, the sample database for all the work in this chapter.

If you check the check box labeled Make This a Shared Data Source (on the Select the Data Source screen), the data source is deployed to the server and can be used by other reports. When they are deployed to the Report Server, the connection string and credentials are encrypted using the Report Server encryption keys. Keep in mind that a report can use zero, one, or several data sources, and a data source can be referenced by one or more datasets.

In the next wizard step (Design the Query), you can either paste a T-SQL statement (including statements such as `EXEC stored_procedure_name`) directly into the Query string window, or you can use Query Builder. Query Builder enables you to select tables and columns, build relationships, and apply filters to your input data. By either means, you end up with a T-SQL statement that will be created as a report dataset.

When building reports without using the Report Wizard, you always have the same options of either typing your T-SQL directly or using Query Designer (illustrated in Figure 48.12). This functionality is accessible via the Report Data tool window; you right-click your data source name and then click Add Dataset to create a new one or click Dataset Properties to modify an existing one.

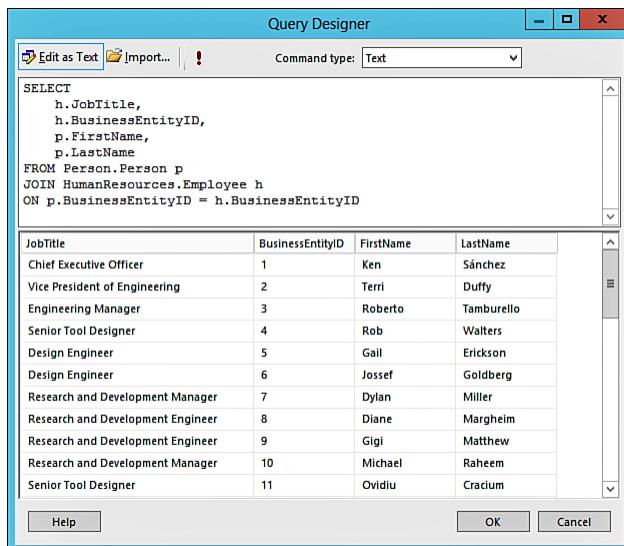


FIGURE 48.12 Creating a new report dataset using the Query Designer.

The Query Designer supports out-of-the-box queries against SQL Server databases, Analysis Services cubes, Oracle databases, and any generic OLE DB and ODBC drivers. If your queries contain parameters, the Query Designer prompts you to provide the necessary values when you execute the report.

Type or paste the code in Listing 48.1 into the Query string window.

LISTING 48.1 T-SQL Code for a Simple Wizard-Generated Report

```
SELECT
    h.JobTitle,
    h.BusinessEntityID,
    p.FirstName,
    p.LastName
FROM Person.Person p
JOIN HumanResources.Employee h
ON p.BusinessEntityID = h.BusinessEntityID;
```

When the wizard finishes, it executes your T-SQL and saves the result in a dataset, the storage container for your report data. You will see this new dataset displayed on the Data Sources Toolbox window after the wizard is complete.

In the next step (Select the Report Type), you can set up your report in either a tabular or matrix format. For this example, select Tabular and click Next. In the Design the Table step, you choose which fields to display on the report. The three sections displayed on this screen are implemented as follows:

- ▶ **PageFields**—Fields added here end up on the top of the report.
- ▶ **GroupFields**—Fields added here create the groupings for your report data (including a summary row).
- ▶ **DetailsFields**—Fields added here make up the detail rows for your report.

For this example, add `JobTitle` to the page area, skip the Group area, add all the other fields to the Details area (your window should now look something like the one in Figure 48.13), and then click Next. Choose a color theme for your report, click Next, and on the Completing the Wizard step, name your report `EmployeesByJobTitle`, check the Preview Report check box, and finally click Finish. Your completed report opens in Report Designer (RD) with its Preview pane (or the Output tool window) in focus.

Switch to the Preview pane (if not already there) and examine the final report. Notice the toolbar across the top which enables pagination, skipping to a particular report page, refreshing the report (rerunning the report query), printing, page layout, page setup and export features. If you're wondering why the page numbers are listed as `1 of 2?`, the reason is that each report page is rendered *on demand*; therefore, the total page quantity is not known unless you move through all pages or skip to the last page (using the toolbar or keyboard shortcuts).

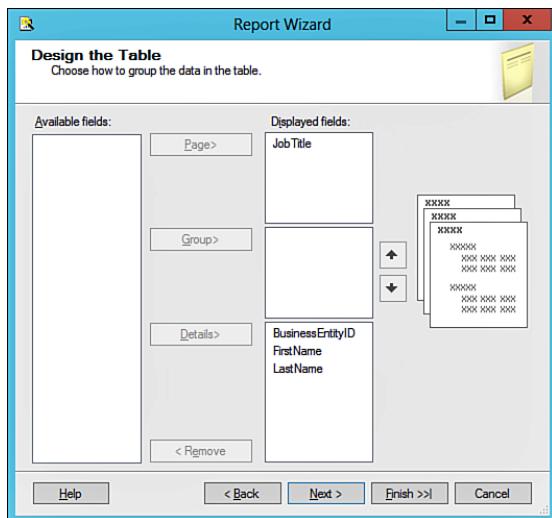


FIGURE 48.13 Field selections using the SSDT Report Wizard.

Flip through the pages of your report. Notice how each new `JobTitle` value generates a new page, with the employees who have that `JobTitle` listed on that page. To understand the report settings behind this implementation, click on the Design tab (at the top of the surface) to switch to Design mode. Notice the Row Groups and Column Groups panes docked below the Report Designer surface. Report Designer is your primary SSDT component for developing reports. It includes all of the user interface (UI) controls listed in Table 48.4, as well as the following features:

- ▶ Right-click menus for each control that allow easy navigation among the controls on the design surface. For example, when you are accessing the properties of a cell's text box nested in a Tablix data region (covered later in this chapter in the section, "Working with the Tablix"), one set of commands is shown for the text box itself and another for the enclosing Tablix, preventing you from having to click all over the report to access the context menu you need.
- ▶ Text and HTML formatting capabilities, allowing for font and style mixing, import and display of HTML stored in report field data, and paragraph styling.
- ▶ A Chart control that includes:
 - ▶ Multiple chart types, such as pie, doughnut, pyramid, candlestick, stock, radar, polar, funnel, range column or bar, smooth area, smooth and stepped line, box plot, bar and column cylinder.
 - ▶ Support for secondary and custom axes, custom rotation angles, scales, strip lines, multiple combined areas, legends, and titles; custom intervals; and enhanced interval labeling.

- ▶ Color, text, and other formatting capabilities; 3D effects (for some chart types); and the ability to edit labels directly on the chart at design-time.
- ▶ Built-in statistical and financial calculations.
- ▶ Two subpanes on the surface for controlling data grouping:
 - ▶ The Row Grouping pane, which you use to define and manipulate row groups in your reports.
 - ▶ The Column Grouping pane, which you use to define and manipulate column groups (when developing matrix-style reports).

Both subpanes include a number of visual cues indicating your current grouping settings, as well as drop-down boxes that allow you to add, remove, change, and nest data groups and also define your subtotal and grand total aggregates.

- ▶ The Report Data tool window, which displays all built-in fields, data sources, datasets (shared and nonshared), report parameters, and images, all in an easy-to-navigate tree. From this tree, most of these objects may be dragged and dropped onto the report design surface. The menu bar located at the top of the Report Data tool window offers commands for creating and editing data sources, datasets, parameters, and images.
- ▶ Context-sensitive rulers, a Report Properties dialog (for changing report-wide settings), and a simple context menu.
- ▶ The ability (using the Report menu) to publish report parts to the SSRS catalog (covered in this chapter in the section, “Using Report Parts”).
- ▶ The ability to create, modify, use, and deploy shared datasets (covered in this chapter in the section “Using Shared Datasets”).
- ▶ The ability to rotate text boxes, vertically or horizontally, up to 270 degrees.
- ▶ Support for developing reports with all the visualization controls in the Toolbox window, including the following:
 - ▶ **Tablix**—At first glance, the Table, Matrix, and List controls still appear to be the same as in previous editions of SSRS. However, they have actually been changed to become data region templates, a kind of layout device for the Tablix, a super-control that embodies all the functionality of the Table, Matrix, and List combined.

The Tablix provides functionality for all manner of data grouping; hierarchical nesting of groups; header, footer, and detail row display. Each Tablix cell may contain any other type of Tablix data region or control, enabling an endless variety of layouts.
 - ▶ **Gauge**—Gauges enable the graphical representation of a single data field or aggregate value (sometimes referred to as a key performance indicator or KPI). Each gauge has its own gauge panel, whereupon you may drag and drop

additional gauges to display multiple values. Gauges are displayed in either a radial or linear fashion.

- ▶ **Indicator**—Actually, a type of small gauge, an indicator enables fast visual comprehension of a single data field or aggregate value. You may add additional indicators or gauges to an indicator's gauge panel. When selecting the icons used by your indicator, you choose from a small set of predefined images, usually corresponding to commonly understood symbols, such as traffic signals, directional arrows, rating stars, and so on.
- ▶ **Data Bar**—Actually a type of small chart, the data bar allows for the graphical representation of one or more data series, just as you would find in a chart. (A data bar may also be promoted to a full chart control with a click of the context menu.) These bars can be drawn as horizontal or vertical bars and can express multiple groups (categories) of values within a series.
- ▶ **Sparkline**—Also a type of small chart (and also promotable to a full chart), a Sparkline is much like a data bar in that it enables the visual expression of one or more data series (grouped or ungrouped). Sparklines come in a range of styles, including chart types such as column, line, area, shape, and range. Sparklines and data bars are meant to be small and quickly comprehended; as such, they lack legends, tick marks, labels, and axis lines.
- ▶ **Map**—The map control enables the visualization of geospatial (mapping) data that you may combine with related analytical data. Every map begins with a map layer, which you can source from one of three places:
 - ▶ A predefined U.S. country or state map (these come with SSRS) that you may select from the map gallery.
 - ▶ An Environmental Systems Research Institute, Inc. (ESRI) shapefile (a vector format that contains geographical data and other attributes).
 - ▶ A query against spatial data stored in SQL Server using the new geography or geometry data types (discussed in Chapter 21, “Creating and Managing Tables.”).

As for your related analytical data, you can source it from any related dataset, or it may be embedded in the spatial data source itself. For example, if the data you want to visualize includes product orders by region, you can relate that regional data to fields in the spatial dataset using match fields. This topic is covered later in this chapter in the section, “Working with Maps.”

Finally, and perhaps most exciting, the map control supports the overlay of Bing map tiles (to Internet-connected users) on your map, providing a professional look and feel.

Returning to our sample report, notice that it has a single grouping on the `JobTitle` column. To see how this is set up, under Row Groups, click the black drop-down arrow at the right of the item named `list1_JobTitle` (this is the autogenerated name given to the group). Take note of the menu actions you can perform:

- ▶ **Add Group**—Allows creation of nested and adjacent groups
- ▶ **Add Total**—Creates a summary total row based on the selected group
- ▶ **Delete Group**—Deletes the selected group
- ▶ **Group Properties**—Shows the Group Properties window, from which you can configure formatting, rendering, sorting, filtering, and other advanced options related to the selected group

Click the Group Properties menu item; then click the General tab at the left of the screen. Notice the group expression, `[JobTitle]`, which indicates the field being grouped. Notice how your report's page breaks (which are forced on a per-`JobTitle` value basis) are controlled via the Page Breaks tab. Sorting (by `JobTitle`) is controlled on the Sorting tab. You can also change a number of other options using the remaining tabs.

On the left side of SSDT, notice the Report Data Toolbox window. It provides a hierarchical view of everything related to your report, including data sources, datasets, report dataset fields, built-in fields, report parameters, and images. Expand the `Built-In Fields` node. These fields provide essential data frequently used in reports. You can drag any field to your report, where it will be instantiated as a text box control whose content is expressed by the simple expression pertaining to the field name (that is, `[FieldName]`). Simple expression syntax is covered in the section, “Using Expressions.”

Working with the Tablix

Returning to the report area of the designer, click anywhere on the report itself near the table. Notice how the UI changes to a raised appearance? This indicates that your report is using the Tablix control. The Tablix includes the Table, Matrix, and List controls, providing all their functionality in one. It offers three *data region templates* (Table, Matrix, and List) that you drag from the Toolbox tool window onto the report. Take a moment to open the Toolbox tool window to view these and the other standard controls.

The Tablix offers several important visual clues as to how your report data is organized with the control. Within its inner border, the innermost grouping for your report is always indicated by a dark orange bracket. On its outside (gray) border, groupings are indicated by dark gray brackets, which may be nested depending on your report. Because the Tablix is so important to report development, let's examine it a bit further.

Using the Solution Explorer, right-click your project's Reports folder, select Add, New Item, and then select Report. Drag the Table data region template from the Toolbox onto your new report. Create a dataset (use the query in Listing 48.1) and click OK. Your new table-styled Tablix is bound to your new dataset. Stretch out the Tablix to fit the report; then mouse over its right-most bottom cell. (If you have any difficulty in selecting the Tablix itself [to move or resize it], simply click the upper-left corner of its border; the Tablix changes its state to reveal its grab handles.) Notice the tiny table icon that appears in its upper-right corner. If you click it, you can select the field you want to display in that cell.

A simpler method for setting up a tabular report is to drag each field you want to display from your dataset in the Report Data window to the Header area of each column in the report. To add additional columns, simply right-click any column and select Insert Column, then choose Left or Right. Click on any cell in the bottom row of your Tablix. Notice how the (gray) outer border contains three horizontal lines? This indicates that the data bound to that row represents your detail group, meaning that the row data will repeat once per row (see Figure 48.14; notice the black arrow in the bottom-left corner).

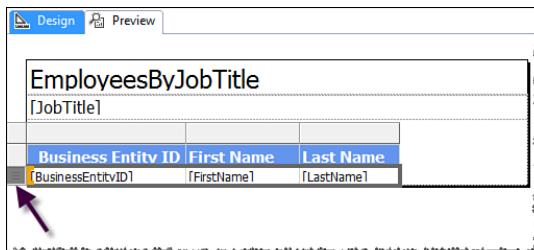


FIGURE 48.14 Detail group row on the table data region of a Tablix.

Right-click any column border's header and then click Tablix Properties. Here, you are presented with a range of options for how to format your Tablix: you can control its name, ToolTip, source dataset, dataset filtering and sorting, page breaks, row and column header repetition rules, and visibility.

Understanding Expressions

A powerful feature of report development is the use of script expressions to dynamically populate report fields and other property values. Report expression syntax is organized into two logical groups: *simple* and *complex*. Complex expressions are simply the Visual Basic .NET (VB.NET) expressions you've used for years. Simple expressions are a newer kind of syntax, allowing for shorthand expression of simple values. For example, instead of expressing a `ProductID` field value as `Fields!ProductID.Value`, you can express it simply as `[ProductID]`. Simple expression syntax is summarized with examples in Table 48.2. (Complex expressions are summarized in Table 48.3.)

TABLE 48.2 Simple Expression Syntax

Report Item	Sample Expression	Equivalent Complex Expression
Built-in field	{BuiltInFieldName}	=Globals!
	BuiltInFieldName.Value	
Data source field	[FieldName]	=Fields!
	FieldName.Value	
Parameter	[@ParameterName]	=Parameters!
	ParameterName.Value	

Report Item	Sample Expression	Equivalent Complex Expression
Aggregate on data source field FieldName.Value)	[StDev(FieldName)]	=StDev(Fields!
Literal text (with escaped brackets)	\[LiteralText\]	[LiteralText]

The Expression Engine

You can create expressions that perform aggregation of an aggregate—for example, =StDev(Sum(Fields!Stocks.Value)).

The expression engine includes the following global variables:

- ▶ **RenderFormat.Name**—Returns the name of the current rendering format. You use it in expressions to produce different output behavior depending on the rendering format.
- ▶ **PageName**—Returns the name of the current page.
- ▶ **OverallTotalPages**—Returns the total number of pages in the entire report.
- ▶ **OverallPageNumber**—Returns the current absolute page number (not impacted by page number resetting).

The expression engine includes the following functions that operate on datasets structured as rows of name-value pairs:

- ▶ **Lookup**—Given two datasets whose rows hold a one-to-one relationship, looks up a single value from a sibling record (by field name and matching value), just as you would when doing a T-SQL JOIN. This function returns a single value.
- ▶ **LookupSet**—Works similarly to Lookup, except that you use this function when your datasets hold a one-to-many relationship. It returns the matching set of corresponding values.
- ▶ **MultiLookup**—Works just like Lookup (on datasets that hold a one-to-one relationship), except that it returns a matching set of corresponding values.

You've seen some simple expressions, so now it's time to delve a bit deeper into complex expressions. Almost every property of every reporting control can have its value determined at runtime as the result of an expression (either simple or complex). You write expressions using VB.NET code. This means you can derive the value of almost any writable report property from contextual report data, built-in or custom function output, .NET assembly method output, or static content. This is no small statement (no pun intended). Table 48.3 shows some examples of complex expressions.

TABLE 48.3 SSRS Complex Expression Examples

Complex Expression	Explanation
=Avg(CInt(Fields!FieldName.Value))	Converts runtime dataset values from FieldName to integer and then averages those values.
="Page " & CStr(Globals!PageNumber) & " of " & CStr(Globals!TotalPages)	Displays a string such as "Page N of N", using global values. (Use this in a header or footer row, or outside a data region.)
=IIf(IsDate(Fields!FieldName.Value), "Yes", "No")	If the context value of FieldName is a valid date, returns the string "Yes"; otherwise returns "No".
=CLng(First(Fields!FieldName.Value, "DataSetName")) << 3	Casts the first row's value of FieldName in integer and then left-bit-shifts that value by 3.

You don't even have to remember these examples to get started; the Report Designer's Expression Editor makes it easy to build complex expressions on your own. You can launch the Expression Editor, shown in Figure 48.15, in two ways:

- ▶ Right-clicking any cell (or another single control) within your report's Tablix (or any other control) and then selecting Expression.
- ▶ Selecting the value column for any writable property in the Properties tool window and then clicking its drop-down box and selecting <Expression...>.

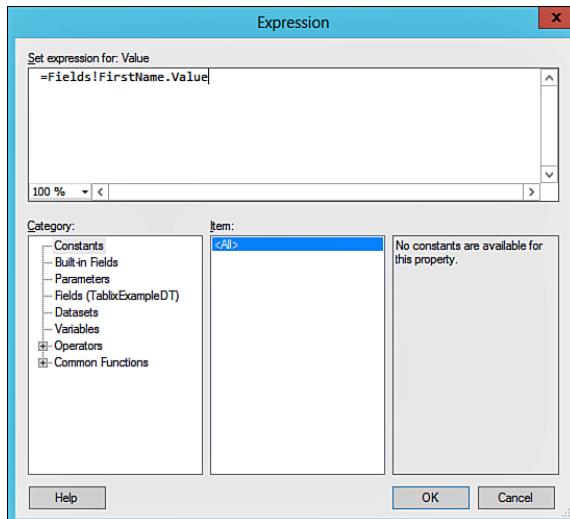


FIGURE 48.15 Using the SSDT Expression Editor.

The top half of the Expression window contains the evaluation area where you type your expressions. It offers IntelliSense (with limited autocompletion) and instant syntax checking. The bottom-left side of the Expression window (labeled Category) offers a complete list of various expression building blocks, including constants, built-in fields, parameters, dataset fields, variables, operators, and built-in functions, grouped by type.

Most complex expressions (those that contain something more than a static value) begin with the equal sign and are built up from there. To use any of these items in your expression, simply click on a Category on the left and then double-click the item you want to add to your expression listed under Item, or, in the case of dataset fields, click on the field in a third list box that appears (named Values). What's even nicer is that as you single-click through each item under Item, the Expression window provides a description and usage example on the right.

Report Design Fundamentals

Every report has three main parts: a body, header, and footer (you can view the header and footer and control their visibility settings by right-clicking an outer edge of the report on the design surface). A report body can be a collection of static controls, such as text boxes and lines, but most useful reports contain at least one data-bound control, meaning that the control is wired up to a dataset; its contents usually repeat in some fashion relative to the number of rows in the dataset. Notice that the header and footer cannot contain data-bound controls. Data-bound controls themselves may contain either data-bound or non-data-bound controls.

Table 48.4 summarizes all the controls in the Toolbox, with their data-binding requirements and some typical uses.

TABLE 48.4 Reporting Services Controls Summary

Control	Data-Bound	Purpose	Notes
Tablix	Yes	Displays tables, matrixes, and lists via data hierarchical groupings and header, footer.	Subsumes Table, Matrix, and List controls (not displayed in the Toolbox).
Table	Yes	Displays tabular data, allowing grouping of rows.	Tablix data region template.
Matrix	Yes	Displays multidimensional data, allowing grouping of both rows and columns (useful for cross-tab data (that is, data having a variable number of columns)).	Tablix data region template.
List	Yes	Displays report content in a simple repeating fashion (once per row); by default, repeats all contained items (tweakable per control).	Tablix data region template.

Control	Data-Bound	Purpose	Notes
Chart	Yes	Provides enhanced graphical display of source data in a wide variety of formats; is great for visualizing results; supports financial reporting, accounting, asset tracking, and so on; supports multiple series of values; provides 2D or 3D display (with or without perspective); internally uses Dundas brand charts.	Provides several styles and rendering options (including 3D).
Gauge	Yes	Provides graphical display of KPI or other single data value; is great for data dashboards.	
Indicator	Yes	Illustrates that a data value falls within a finite set of conditions, values, or thresholds.	
Data Bar	Yes	Displays a small, single-bar chart within a cell.	
Sparkline	Yes	Displays a small chart that quickly illustrates a trend in the data.	
Map	Yes	Renders geospatial and related analytical data; includes support for Bing map tiles; uses <code>SQL geometry</code> or <code>geography</code> data types; can also use ESRI spatial vector data files.	
Rectangle	No	Enables you to lay out reports or other controls; is good for static grouping.	Useful when displaying adjacent controls.
Line	No	Has primarily visual uses (layouts).	Useful for styling.
Image	No	Displays images, either embedded within the report, culled from field data, URLs, or deployed as resources stored within the SSRS catalog. Supported formats: PNG, GIF, JPG, X-PNG.	Gives a report a professional look.
Subreport	No	Displays another report on a report; parameters passed to a subreport enable drill-through (that is, from one report to the next); uses subreports to chunk reports into reusable blocks.	Allows your current dataset to supply values to a subreport as parameters.
TextBox	No	Displays textual data; supports internationalization.	Each Tablix data region cell contains a single text box by default.

Using the Data Visualization Controls: Sparkline, Indicator, and Data Bar

As mentioned earlier, a Sparkline is a small, easy-to-understand chart that you usually place inside the text box cell of a grouping row in a Tablix. Sparklines make it easy to see trends in data at a glance. Most types of Sparklines can be converted to full chart controls when necessary (except for 3D charts). In the following example, we add a Sparkline to a new report.

As you've done before, right-click your `Reports` folder in Solution Explorer and then click Add New Report. As you follow the steps of the Report Wizard, select your shared data source (pointing to the `AdventureWorks2012` database) and then enter the T-SQL shown in Listing 48.2 for your report query.

LISTING 48.2 T-SQL Code for a Sparkline Report

```

SELECT
    h.BusinessEntityID,
    e.Gender,
    p.LastName,
    h.Rate,
    h.RateChangeDate
FROM HumanResources.EmployeePayHistory h
JOIN HumanResources.Employee e
ON e.BusinessEntityID = h.BusinessEntityID
JOIN Person.Person p
ON e.BusinessEntityID = p.BusinessEntityID
ORDER BY h.BusinessEntityID, h.RateChangeDate;

```

This query illustrates changes in hourly pay rates for all employees over their entire term with the company.

Continuing with the Report Wizard, choose the Tabular report style and then add all the columns generated by the query to the detail area of the report (at the Design the Table Wizard step). Select any color theme, name your report `RateChangesByEmployee`, and preview it in SSDT. Switch back to Design view and take the following actions using the Report Designer surface:

1. Click the left edge of your table to select its last row.
2. Using the Row Groups pane (below your report), click the black arrow for your grouping (named `table1_Details_Group` by default), then click Group Properties.
3. On the General tab, under Group Expressions, click the Add button.
4. In the Group On box, select `[BusinessEntityID]`.
5. On the Filters tab, click the Add button.
6. In the Expression box, type the following: `[Count(BusinessEntityID)]`.

7. Change the Data Type drop-down to `Integer`, change the Operator value to `>`, enter the number `1` in the Value combo box, and then click OK.
8. Click the top edge of your table to select the Gender column; then right-click and select Insert Column, Right.
9. Show the Toolbox tool window. Then click and drag a Sparkline control into the text box of the empty cell in the column you just created.
10. Click your new Sparkline control. On the Select Sparkline Type dialog, select the second Area chart from the left and click OK.
11. Click your Sparkline control again; then click the green plus sign above the Values box, select the `Rate` field, and then click anywhere on the design surface.
12. Returning to the table, change the expressions for the Rate and Rate Change Data columns to `[First(Rate)]` and `[Last(Rate)]`, respectively.
13. Change the header text for the Rate and Rate Change Data columns to `Starting Rate` and `Final Rate`, respectively.
14. Preview the report; the result should look something like that shown in Figure 48.16.

ID	Gender		Last Name	Starting Rate	Final Rate
4	M		Walters	8.6200	29.8462
16	M		Bradley	24.0000	37.5000
167	M		Johnson	6.5000	9.5000
170	M		Li	6.5000	9.5000
172	M		Ingle	6.5000	9.5000
174	M		Martin	6.7500	9.5000
175	M		Koch	6.5000	9.5000
176	M		Lawrence	6.5000	9.5000
177	M		King	6.5000	9.5000
178	M		Frum	6.5000	9.5000
224	M		Vong	8.6200	16.0000
234	F		Norman	39.0600	60.0962
250	F		Word	9.8600	30.0000

FIGURE 48.16 Sparkline report in Preview mode.

In the next example, we examine the Indicator control. Listing 48.3 illustrates the T-SQL for this report. In this report, you use the indicator as an icon representing the shift worked by a distribution of employees.

LISTING 48.3 T-SQL Code for an Indicator Report

```

SELECT
    e.BusinessEntityID,
    p.FirstName,
    p.LastName,
    s.ShiftID
FROM HumanResources.Shift s
JOIN HumanResources.EmployeeDepartmentHistory h
ON s.ShiftID = h.ShiftID
JOIN HumanResources.Employee e
ON e.BusinessEntityID = h.BusinessEntityID
JOIN Person.Person p
ON p.BusinessEntityID = e.BusinessEntityID
WHERE e.OrganizationLevel = 3
AND e.BusinessEntityID BETWEEN 40 AND 108
ORDER BY p.LastName;

```

Add a new report by following the Report Wizard steps for a tabular report (just as in the Sparkline report example), using the T-SQL in Listing 48.3 for your query. Name your report `EmployeeShifts`, open it on the Report Designer surface, and take the following actions:

- 1.** Click the cell that reads `[ShiftID]`. Then right-click it and choose Delete to delete the text box in that cell.
- 2.** Show your Toolbox tool window; then click and drag an Indicator control into the cell where you just deleted the text box.
- 3.** On the ensuing Select Indicator Type dialog, choose a set of icons that appeals to you and then click OK.
- 4.** Click your new Indicator control. Then, under the Values box, where it reads `(Unspecified)`, click the black drop-down arrow and then select `ShiftID`.
- 5.** Preview your report; the result should look something like that in Figure 48.17.

48

For our next example, we utilize the Data Bar control. In this simple report, we illustrate quantities sold of bicycle parts. Add a new report by following the Report Wizard steps for a tabular report (just as in the Sparkline example), using the T-SQL in Listing 48.4 for your query. Name your report `QuantitiesSold`.

The screenshot shows a report preview window with a toolbar at the top. The main content is a table titled "EmployeeShifts". The table has four columns: "Business Entity ID", "First Name", "Last Name", and "Shift ID". The "Shift ID" column contains small colored diamonds (yellow, green, red) in each cell. The data rows are as follows:

Business Entity ID	First Name	Last Name	Shift ID
62	John	Campbell	Yellow Diamond
40	JoLynn	Dobney	Green Diamond
78	Reuben	D'sa	Yellow Diamond
47	Andrew	Hill	Red Diamond
108	Jinghao	Liu	Red Diamond
93	Kok-Ho	Loh	Yellow Diamond
55	Taylor	Maxwell	Green Diamond
102	Zheng	Mu	Yellow Diamond
87	Cristian	Petculescu	Green Diamond
71	Michael	Ray	Green Diamond

FIGURE 48.17 Indicator report in Preview mode.

LISTING 48.4 T-SQL Code for a Data Bar Report

```

SELECT TOP 15
    p.ProductID as ID,
    p.Name,
    p.ListPrice,
    SUM(d.OrderQty) as TotalSold
FROM Production.Product p
JOIN Sales.SalesOrderDetail d
ON d.ProductID = p.ProductID
GROUP BY p.Name, p.ListPrice, p.ProductID
ORDER BY SUM(d.OrderQty) DESC;

```

Follow these steps to add a data bar visualization to your report:

1. Click the cell that reads `[TotalSold]`. Then right-click it and choose Delete to delete the text box in that cell.
2. Show your Toolbox tool window; then click and drag a Data Bar control into the cell where you just deleted the text box.
3. On the ensuing Select Data Bar Type dialog, choose a data bar style that appeals to you and then click OK.
4. Click your new Data Bar control. Then, under the Values box, where it reads `(Unspecified)`, click the black drop-down arrow and then select `TotalSold`.
5. Preview your report; the result should look something like that shown in Figure 48.18.

ID	Name	List Price	Total Sold
712	AWC Logo Cap	8.9900	<div style="width: 100%;"> </div>
870	Water Bottle - 30 oz.	4.9900	<div style="width: 50%;"> </div>
711	Sport-100 Helmet, Blue	34.9900	<div style="width: 75%;"> </div>
715	Long-Sleeve Logo Jersey, L	49.9900	<div style="width: 80%;"> </div>
708	Sport-100 Helmet, Black	34.9900	<div style="width: 60%;"> </div>
707	Sport-100 Helmet, Red	34.9900	<div style="width: 65%;"> </div>
864	Classic Vest, S	63.5000	<div style="width: 25%;"> </div>
873	Patch Kit/8 Patches	2.2900	<div style="width: 30%;"> </div>
884	Short-Sleeve	53.9900	<div style="width: 40%;"> </div>

FIGURE 48.18 Data Bar report in Preview mode.

Deploying Reports

When you are happy with your reports' content and layout, to make them available to others, you deploy (also known as publish) them to the SSRS catalog using SSDT or Report Manager, where they are stored. When you publish a report, its content is validated and compiled to an internal format in which it is saved. (If the report contains code, it is compiled into .NET assemblies.)

Once in the catalog, reports can be managed, secured, and delivered to end users in a variety of formats, including HTML, Excel, PDF, TIFF, Word, CSV, and XML. When they are deployed, various delivery, caching, and execution options are made available, as are scheduling and historical archiving, covered later in this chapter.

To deploy a report, you must first specify a Report Server URL in the properties of your SSDT project (for example, [http://\[servername\]:\[port\]/ReportServer](http://[servername]:[port]/ReportServer)). Note that this URL must point to the web service virtual directory, not the main [http://\[servername\]:\[port\]/Reports](http://[servername]:[port]/Reports) directory, which is only for Report Manager. On the Properties window, you can also specify the catalog folder for your shared data sources, shared datasets, and report parts, and whether they are to be overwritten upon redeployment, as well as your target catalog folder for reports and the startup report for your project. Figure 48.19 illustrates the project deployment settings for our example.

When your reports are successfully deployed, authorized users can execute them using Report Manager or any other SSRS-integrated tools. To deploy from SSDT, right-click your Report Server project name in the Solution Explorer and then select Deploy. You might need to first authenticate (once per user session) to the Report Server because it is secured.

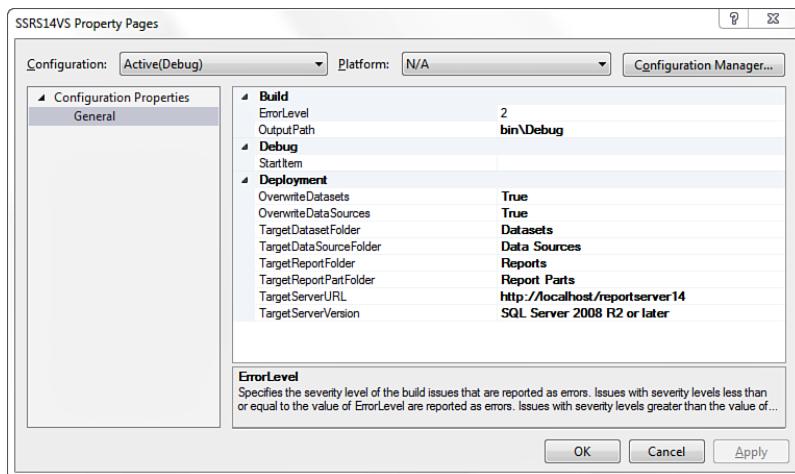


FIGURE 48.19 Deployment settings for a Report Server project in SSDT.

The execution of reports includes a key process known as *rendering*. This step meshes the retrieved report data with the RDL markup and the transformation of that result to a specific output format.

Report Rendering

The latest rendering capabilities in SSRS offer fine-grained control over report output. Highlights include the following:

- ▶ The comma-separated value (CSV) supports two styles of rendering: *default* mode, which is optimized for Excel; and *compliant* mode, which is the standard, format-free CSV you would expect to see rendered.
- ▶ Overlapping data regions, repeating items (such as headers and footers), page breaks, and aggregate visibility may be handled differently according to the rendering extension in use. SSDT warns you upon compilation of your reports if such overlapping occurs.
- ▶ Page rendering happens *on demand*, meaning that as a user scrolls or pages through a report, the next page of the report is dynamically rendered.
- ▶ Report page counts are not always known at runtime (until the last page is rendered). This is reflected in the Report Viewer toolbar. (Page counts are now displayed as 1 of ?.)
- ▶ The Microsoft Excel rendering extension supports rendering data regions and subreports that are nested inside Tablix cells. Also, Excel worksheet tabs can be named. (You achieve this at design time by setting the value of the `InitialPageName` property of your report to an expression that returns a name.)

- Extra whitespace in the body of your reports is preserved during rendering (although you may change this behavior using the `ConsumeContainerWhitespace` property of the report).

Using Report Manager

Report Manager's Web user interface has a SharePoint-like color scheme and makes liberal use of Asynchronous JavaScript and XML (AJAX) technology for a fast overall experience (less navigation per click).

The latest Report Manager also includes a SharePoint-style context menu for each folder item. This works in a manner similar to what happens when you click a list item in a SharePoint site. These context menus offer actions that you can take without navigation, depending on the type of item you select.

Report Manager is organized hierarchically as a folder tree. Using it, you can create, delete, rename, secure, and organize all your deployed reporting objects, as well as the folders that contain them. The root Report Manager folder is simply known as `Home`, from which all other report objects in the tree descend. Permissions (known as *role assignments*) are inherited from parent item to child, and they may be overridden at any level.

Take some time now to explore Report Manager. To do this, navigate to the URL you configured earlier in the chapter (it should be something like `http://[servername]:[port]/Reports`). The layout of Report Manager is fairly straightforward: site configuration links are found at top right, where you can modify settings, get help, and (if you have personal folders enabled) view your subscriptions. The main site navigation enables you to create folders and data sources, launch Report Builder, upload a file to the SSRS catalog, and change the permissions for the current folder (in this case, `Home`).

If you've created all the sample reports so far, below the main navigation you find icons representing the folders to which you deployed your report items from SSDT. These folders correspond to the deployment folders you configured using the Properties page of your report project. If you hover your cursor over any folder, the outline of a drop-down box appears. If you click the drop-down, you get the following menu choices (provided you have adequate permissions): Move, enabling you to move the item; Delete, to delete it; Security, to set its permissions; and Manage, to view or change its properties. If you click the name of the item itself, the default action for that type of item is executed. Every deployed item in Report Manager has a hover menu and primary link. The options offered by each menu are item specific. Figure 48.20 illustrates how to work with Report Manager.

Navigate around your folders and view the properties for various items, such as your shared data source, shared dataset, and reports. Next, click the Details View link on the right side of the toolbar. Notice the new main navigation buttons that appear, allowing you to move and delete deployed items easily. If your SSRS content is still sparse, a great way to get started with Report Manager content is to download and install the sample reports for AdventureWorks2012 from www.codeplex.com.

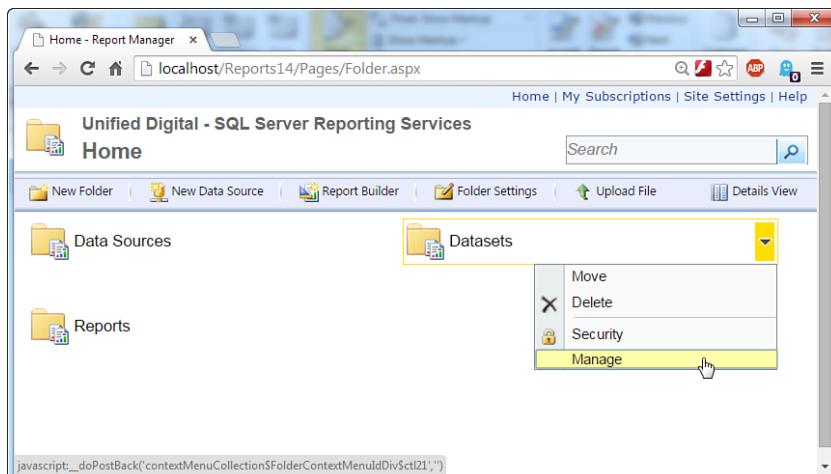


FIGURE 48.20 Working with Report Manager.

Continuing with Report Manager, click the Site Settings link and then click on the General tab and configure a different name for your site (notice the company name displayed at the top of Figure 48.20). On this page you can also control the execution timeout limit for all reports, the number of report snapshots that SSRS stores by default, and the ClickOnce URL to launch Report Builder (you can use this to launch a different version of Report Builder than the default, version 3.0). Snapshots are covered in more detail in the section, “Execution Snapshots.”

The Security tab in the settings area provides a straightforward means of adding and removing site users and changing role assignments. The Schedules tab enables you to set up schedules that you can later apply to report execution and delivery jobs. This chapter covers a few of these management tasks in greater depth in the section, “Management and Security.”

Using Tables and Hierarchies in Reports

Now that you’ve had a quick tour of Report Manager, it’s time to go a little deeper into our report development examples.

Let’s say you want to represent an organizational hierarchy in a report (a fairly common task). Using the SSDT Solution Explorer, create a new report by selecting Add New Item (don’t use the Report Wizard this time) and name it `EmployeeHierarchy.rdl`. Drag a Table data region from the Toolbox onto the report; then use the query in Listing 48.5 for its dataset (which is named `DataSet1` by default). Note that the `OrganizationNode` column in this query is of type `hierarchy_id`.

LISTING 48.5 T-SQL Code for a Hierarchical Data Report

```

SELECT
    e.JobTitle,
    e.BusinessEntityID,
    e.OrganizationNode.ToString() as OrganizationNode,
    e.OrganizationNode.GetAncestor(1).ToString() as ParentOrganizationNode,
    p.FirstName,
    p.LastName
FROM Person.Person p
JOIN HumanResources.Employee e
ON p.BusinessEntityID = e.BusinessEntityID
ORDER BY p.BusinessEntityID;

```

Add three new columns (for a total of six) to your table control; then, using the Report data tree, drag each dataset field onto the report. Next, using the Row Groups pane, click the Details drop-down box and then select Add Group, Parent Group. In the ensuing Tablix Group dialog, select `ParentOrganizationNode` in the Group By drop-down box, check the Add Group Header check box, and then click OK.

Next, using the Row Groups pane again, click its `ParentOrganizationNode` drop-down box; then select Add Group, Child Group. In the ensuing Tablix Group dialog, select `OrganizationNode` in the Group By drop-down box, check the Add Group Header check box, and then click OK. In Design mode, your report should look something like the one shown in Figure 48.21.

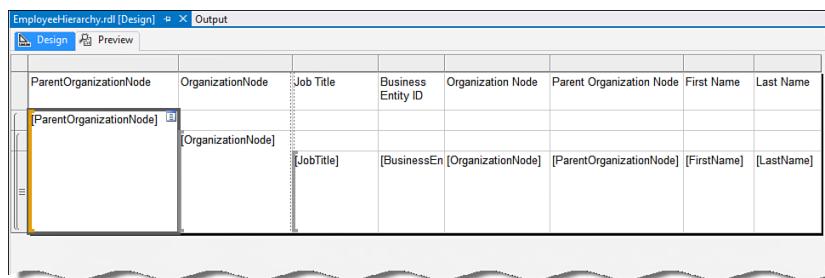


FIGURE 48.21 A hierarchically grouped report in Design mode.

Adding Interactivity

The next step is to add drill-down interactivity. To do this, select the final row in your Tablix by clicking on its gray border at the left edge of the table. Next, right-click the same border and then select Row Visibility. On the ensuing Row Visibility dialog, select Hide using the When the Report Is Initially Run radio button. Check the Display Can Be Toggled by This Report Item check box; then select `ParentOrganizationNode` in the drop-down box (see Figure 48.22).

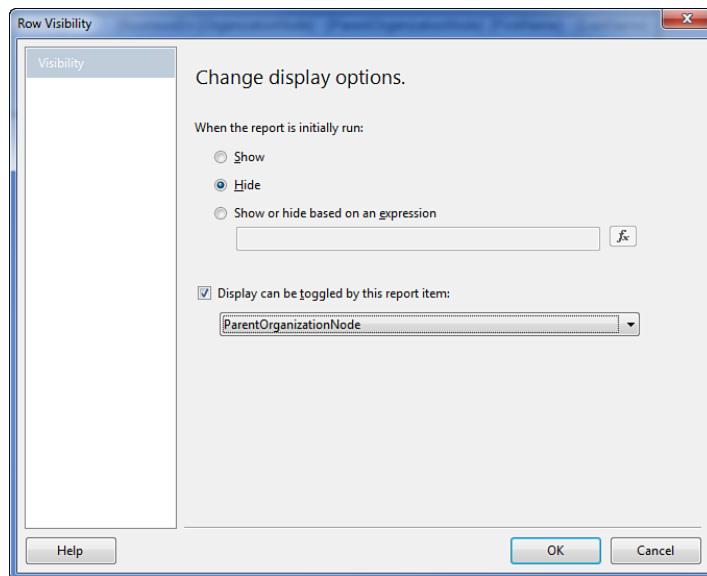


FIGURE 48.22 Changing detail row visibility in a hierarchically grouped report.

Save and preview the report. The result should resemble the report shown in Figure 48.23.

EmployeeHierarchy.rdl [Design] < X Output							
	Design	Preview	Find	Next			
ParentOrganizationNode	OrganizationNode	Job Title	Business Entity ID	Organization Node	Parent Organization Node	First Name	Last Name
□	/	Chief Executive Officer	1 /			Ken	Sánchez
□ /	/1/	Vice President of Engineering	2 /1/	/		Terri	Duffy
	/2/	Marketing Manager	16 /2/	/		David	Bradley
	/3/	Vice President of Production	25 /3/	/		James	Hamilton

FIGURE 48.23 Hierarchically grouped report with toggle in Preview mode.

By clicking the + and - signs, your users can now expand and collapse the hierarchy.

Now that you've built a number of reports using SSDT, it's time to learn how to work with the other primary reporting tool, Report Builder 3.0.

Designing Reports Using Report Builder

Report Builder (RB) is a report authoring application that enables advanced users to design and publish their own reports. Users can generate reports using data generated by any T-SQL query from any supported data source, data from report models (although these must be built using an older edition of SSDT), and mapping sources.

Report Builder

Report Builder simplifies both the data-retrieval and the layout-design phases of report building for non-developers, although it still helps to have a developer around for configuration and advanced tasks.

Its user interface (UI) has the look and feel of an Office 2010 application, including its Ribbon bar. It offers data visualization and formatting enhancements; on-demand rendering; and support for multiple data sources, shared datasets, and report parts. It is available as a ClickOnce application that you launch from Report Manager or SharePoint (if using SharePoint-integrated SSRS mode, a topic beyond the scope of this chapter).

In terms of overall usability, you can quickly preview reports in Report Builder, thanks to its use of *edit sessions* that enable reuse of cached dataset data.

RB Versions

SSRS 2014 comes with Report Builder 3.0 (RB3) bundled as a ClickOnce application. If you need Report Builder 2.0, you can still (as of this writing) download it from Microsoft. Using the main navigation button on Report Manager, you launch RB3 by clicking the Report Builder link on the main menu. You may also install RB3 on a user workstation (download the SQL Server Feature Pack) as a standalone application you launch via the Windows Start menu or screen.

Installing RB3

When installing the standalone version of RB3, you encounter two options screens:

- ▶ **Feature Selection**—Gives you a choice of file installation location.
- ▶ **Default Target Server**—Prompts you to enter the URL to your installation of the SSRS web services (shown in Figure 48.24). In some cases, you will experience an installation error if you enter a default server URL. A workaround for this is to leave this field empty, then enter the value post-install. When RB3 is installed, you can modify this option by clicking the Office button at the top left of the main UI and then click Options to open the screen shown in Figure 48.25.

Be aware that RB3 runs only in full trust mode. To launch it from Report Manager, use the following URL: [http://\[server-name\]/\[path-to-report-server\]/ReportBuilder/ReportBuilder\[version\].application](http://[server-name]/[path-to-report-server]/ReportBuilder/ReportBuilder[version].application).

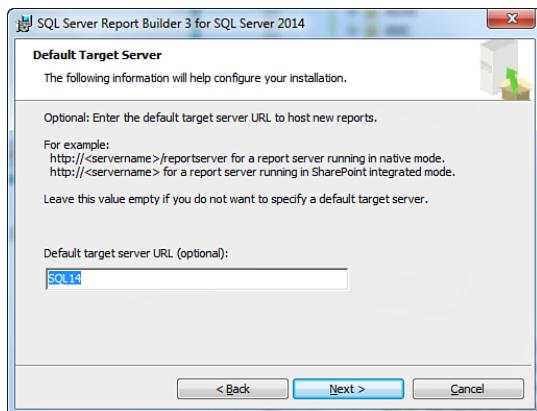


FIGURE 48.24 Report Builder 3.0 Standalone MSI Installation: Default Target Server screen.

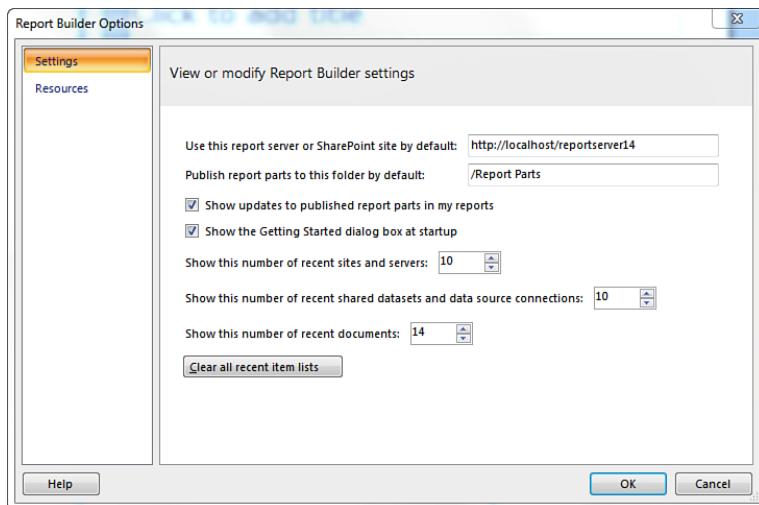


FIGURE 48.25 Report Builder 3.0 Options screen.

By default, the executable program (`MSReportBuilder.exe`) is located in the following folder:

```
%PROGRAMFILES (x86)%\Microsoft SQL Server\Report Builder 3.0
```

You can run this program using the Windows start screen or, on older Windows operating systems, via the menu shortcut located in the Windows Programs menu under

```
Programs\Microsoft SQL Server 2014 Report Builder 3.0
```

Getting Started with RB3

When you run the RB3 application the first time, the main window appears as shown in Figure 48.26.

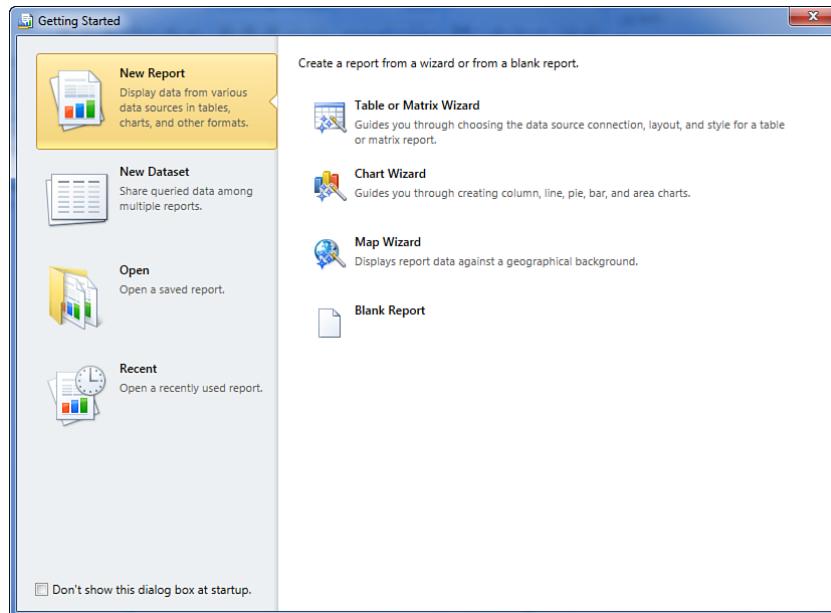


FIGURE 48.26 Report Builder 3.0 first run UI (new item screen).

As you can see, RB3 comes with a number of wizards that make it easy for you to create reports that use tables, matrices, charts, and maps, as well as create datasets and blank reports (you learn how to use the Map Wizard a bit later in the section, “Working with Maps”).

If you’ve been following along with all the examples in this chapter, you’ll notice that the RB3 UI includes many of the same features as the SSDT Report Designer, including the Report Data tool window (which you can use just as you would with SSDT to work with data sources, datasets, built-in and custom fields, images, and parameters), Row Groups and Column Groups designer panes, rulers, Properties window, and right-click menus. Of course, the big difference is the implementation of the Office Ribbon bar and associated wizards, which make it very easy to build your report. Plus, with RB3, you also get the new Report Part Gallery (report parts are covered in the section, “Using Report Parts”). Take a few minutes to explore the different sections of RB3’s Ribbon bar. Table 48.5 summarizes all its features.

TABLE 48.5 RB3 Ribbon Bar Features by Tab

Features	Uses	Tab	Tab Group
Run report	Execute your report against the target server	Home	Views
Cut, copy, paste	Move report items	Home	Clipboard
Styling	Format report text, paragraph, and border styling	Home	Font, Paragraph, Border
Numeric formatting	Style numbers displayed in reports	Home	Number
Control positioning, cell structure	Change z-order index and relative alignment of report controls; merge and split cells	Home	Layout
Report controls that use data regions	Insert Tablix data regions (Table, Matrix, and List) into reports; includes wizards for easy generation	Insert	Data Regions
Graphical data-bound report controls into reports; includes some wizards for easy generation	Insert Chart, Gauge, Map, Data Bar, Sparkline, and Indicator	Insert	Data Visualizations
Non-data-bound report controls	Insert Text Box, Image, Line, and Rectangle controls into reports	Insert	Report Items
Subreport control	Insert Subreport controls into reports	Insert	Subreports
Header & footer	Control visibility of report header and footer	Insert	Header & Footer
Visibility settings	Toggle visibility of all designer tool windows and panes, including the Report Part Gallery	View	Show/Hide
Design	Return to Design mode when running a report	Run (appears on the Ribbon whenever you run a report)	Views
Zoom	Zoom in or out	Run	Zoom
Page navigation	Move from page to page and back, refresh report, stop execution	Run	Navigation
Printing	Manage print setup and layout options	Run	Print

Features	Uses	Tab	Tab Group
Exporting	Export a report to a file in any of the supported formats	Run	Export
Parameters, Table of Contents (TOC)	View the document map (the report's table of contents) and any report parameters	Run	Options
Searching	Locate text within the report	Run	Find

Let's start using the RB3 designer surface. To begin, close the Getting Started dialog, and then click to highlight the text that reads Click to Add Title; then enter `Sales by Product`. Click the Save button at the top left. Notice how the ensuing Save As Report dialog actually shows you the folders as they are organized in your Report Server catalog (illustrated in Figure 48.27). This dialog provides a view of the server folder hierarchy, not your local file system as you might expect. Double-click an appropriate folder, enter a report name, and then click Save.

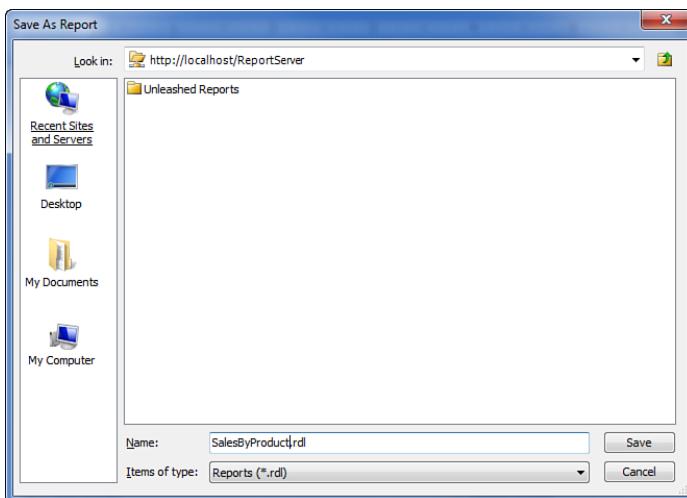


FIGURE 48.27 Viewing SSRS folders using Report Builder 3.0's Save As Report dialog.

Returning to the designer, click the Insert Ribbon bar menu item, then click the Chart button and select Chart Wizard. On the first step of the Chart Wizard, select the radio button to create a dataset. On the next step, click the Browse button and navigate to your shared data sources folder. Then select the shared data source that you created and deployed in a previous example. Then click Next.

The UI in the next step (Design a Query) is built to visually represent in a logical structure the tables, views, and stored procedures in your selected database. On the left, database objects are grouped by schema name. On the right, the fields you select from these objects

are added to your query, as well as the appropriate relationships and any filters you want to apply to your data.

Using the Database View panel, expand the `Sales` schema node; then expand the `Tables` node and make the following selections: from `SalesOrderHeader`, check the boxes next to the `SalesOrderId` and `OrderDate` columns. From `SalesOrderDetail`, check `SalesOrderId`, `ProductID`, and `OrderQty`. Now, scroll up to the root of the tree and expand the `Production` schema node. From the `Product` table, check the `ProductID` and `Name` columns. Click the Run Query button. Your Design a Query window should now look something like the one in Figure 48.28.

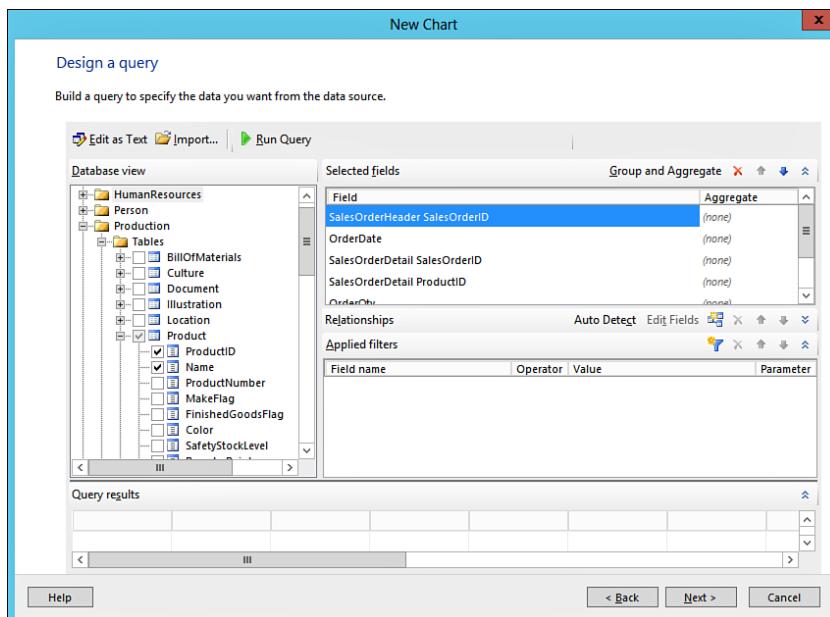


FIGURE 48.28 Report Builder 3.0's Design a Query window in graphical mode.

When you click Next, a dialog pops up, telling you that the relationship between `SalesOrderDetail.ProductID` and `Product.ProductID` could not be detected. This makes sense because the relationship is only implied in the physical model. Accept the dialog or cancel it (either choice does the same thing); then use the red x button to remove the `ProductID` and `Name` columns from the Selected fields list; then click the Edit as Text button to view the T-SQL being generated. Replace the entire contents with the T-SQL from Listing 48.6. When finished, click the Run Query button again to make sure your query is correct (as shown in Figure 48.29); then click Next.

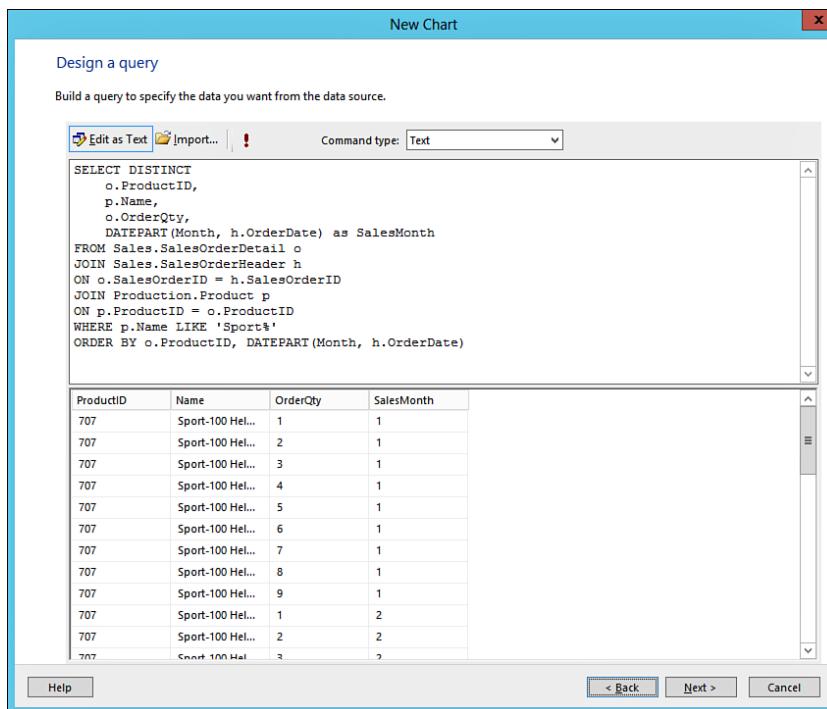


FIGURE 48.29 Report Builder 3.0's Design a Query window in T-SQL edit mode.

LISTING 48.6 T-SQL Code for a Report Builder 3.0 Chart Report

```

SELECT DISTINCT
    o.ProductID,
    p.Name,
    o.OrderQty,
    DATEPART(Month, h.OrderDate) as SalesMonth
FROM Sales.SalesOrderDetail o
JOIN Sales.SalesOrderHeader h
ON o.SalesOrderID = h.SalesOrderID
JOIN Production.Product p
ON p.ProductID = o.ProductID
WHERE p.Name LIKE 'Sport%'
ORDER BY o.ProductID, DATEPART(Month, h.OrderDate);

```

On the ensuing Choose a Chart Type screen, select Column and then click Next. On the Arrange Chart Fields screen, drag and drop SalesMonth from the Available Fields list box to the Categories panel; then drag and drop Name to the Series panel and OrderQty to the

Values panel and click Next. Select a chart style; then click Finish. Stretch out your chart a bit on the designer and then double-click it to edit the following text items: chart title, categories axis text, and series axis text. Now, right-click the chart control on any of its chart bars, select 3D Effects, and then check the Enable 3D check box. Click OK, save, and then run the report. The result is shown in Figure 48.30.

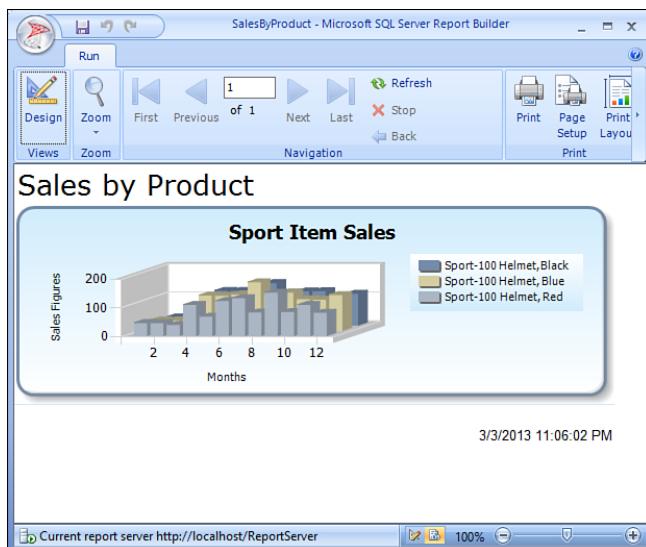


FIGURE 48.30 Rendered 3D chart report using Report Builder 3.0.

Using Report Parts

With editions of SSRS prior to 2012, when you wanted to reuse a portion of a report, such as a data-bound table, you had to make that section of the report into a subreport and then use that subreport on different reports. This functionality is, of course, still supported in SSRS 2014, but in addition, you can now reuse charts (including data bars and Sparklines), gauge panels (including gauges and indicators), images, lists, maps, matrixes, report parameters, rectangles, tables, or shared datasets. To do this, you simply publish them to the SSRS catalog, making them available for reuse using Report Builder or SSDT. This section shows you how.

Best practices dictate that you use shared data sources and shared datasets to achieve maximum reuse and centralized distribution of your report data. Report parts take this a step further, enabling you to deploy the charts, tables, gauges, and other data visualization controls you create on one report to be reused on any other report. You no longer have to rely on using subreports to create reusable chunks of report functionality, although report parts do not render subreports obsolete; they still have their place in the Toolbox.

You can start using this feature by publishing the report parts that live in the chart example you just completed. Using RB3 with a report open, click on the Office button

(top left of the application UI) and then select Publish Report Parts. On the ensuing Publish Report Parts dialog, click the first button to publish all report items (those that may become report parts) or click the second button to review your report parts before publication.

The report controls that may become report parts are charts, data bars, Sparklines, gauges, indicators, images, lists, maps, matrices, report parameters, rectangles, tables, and shared datasets. As illustrated in Figure 48.31, the Select Report Parts to Publish step lists the publishable items, and you may check any combination of those you would like to publish. Check both the dataset and chart. Checking the dataset's check box automatically turns the report's (formerly embedded) dataset into a shared dataset on deployment. Click the Publish button; then click Close.

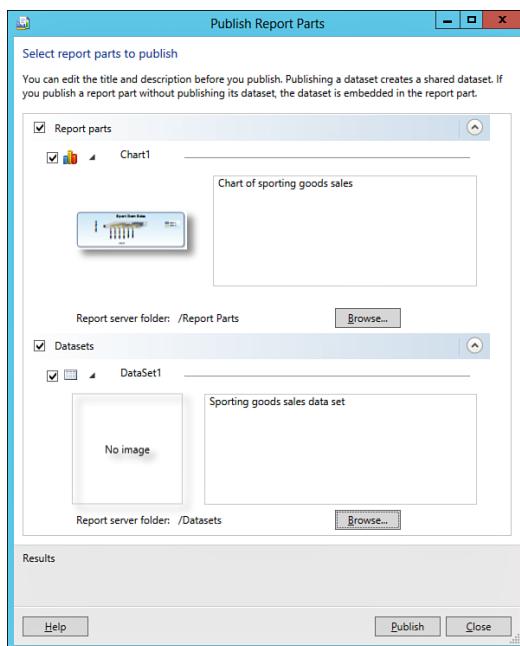


FIGURE 48.31 Selecting report parts to publish using RB3.

When publication is complete, return to the main designer and click on the Report Part Gallery tool window tab, go to the Insert tab in the Ribbon and click on the Report Parts in the Parts section, located at the far right of the Insert Tab (docked with the Properties window; go to the View tab in the Ribbon and check the checkbox to enable this). Click its magnifying glass icon to search for all published report parts (excluding datasets; they are not displayed in the gallery). Notice how the results grid displays a thumbnail of the chart you just created (shown in Figure 48.32). You (or any other privileged user) can now drag your chart from the Report Part Gallery onto the designer surface and reuse it. Keep

in mind that anyone can reuse report parts as-is or modify them at will (provided he or she has the appropriate permissions).

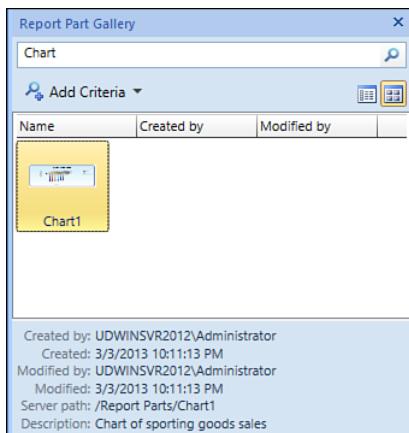


FIGURE 48.32 Using the Report Part Gallery in RB3.

After you modify a report part, you can then choose to republish it with your changes, overwriting the current version of the same. Note that this does not change the instances of the report part that live on other reports; those reports continue to use the version of the report part they dragged from the gallery when those reports were designed. In contrast, any new report that uses that report part for the first time gets your updated version from the gallery.

When the report's designer who used your report part (in its original state) opens the report at some future date in RB3, RB3 checks the SSRS catalog to see whether the report part has been updated since the report was last opened. Because you updated and overwrote the original version of the chart, the designer is presented with the option to either accept your latest changes or continue using the version of the chart he or she originally added to the report. The designer can also check at any time to see whether any updated versions of the report parts on his or her report are available by clicking the Office button and selecting Check for Updates. An interactive information bar appears just below the Ribbon, indicating whether any report parts used in the current report have available updates. It is then up to the designer to accept or ignore any new versions.

Returning to report building, let's keep the momentum going by quickly creating a map report.

Working with Maps

To create a map report, start by clicking the Office button and then selecting New. On the ensuing New Report or Dataset screen, click the Map Wizard button; then click Create at the bottom right. Click Next. Then, on the Choose a Source of Spatial Data screen, select the SQL Server Spatial Query radio button. Then click Next. On the following screen, select the Add a New Dataset radio button and then click Next. Select the shared

data source you created earlier and then click Next. On the Design a Query screen, click the Edit as Text button and then paste the following into the text area: EXEC Person.GetSampleSpatialData. Before proceeding to the next step, launch SSMS, connect to your data source, open a new query window, and execute the T-SQL shown in Listing 48.7. Now, back in RB3, you can click the exclamation point button and then click Next.

LISTING 48.7 T-SQL Code for a Report Builder 3.0 Map Report

```
use AdventureWorks2012
go
CREATE PROC Person.GetSampleSpatialData
AS
DECLARE @T TABLE
(
    Quantity int,
    StateProvinceName nvarchar(50),
    StateProvinceCode char(2),
    Location geography
);

INSERT @T
SELECT
    SUM(d.OrderQty) Quantity,
    s.Name,
    s.StateProvinceCode,
    NULL
FROM Sales.SalesOrderHeader h
JOIN Sales.SalesOrderDetail d
ON h.SalesOrderID = d.SalesOrderID
JOIN Sales.SalesTerritory t
ON t.TerritoryID = h.TerritoryID
JOIN Person.Address a
ON a.AddressID = h.ShipToAddressID
JOIN Person.StateProvince s
ON s.StateProvinceID = a.StateProvinceID
WHERE t.CountryRegionCode = 'US'
GROUP BY s.StateProvinceCode, s.Name;

UPDATE @T
    SET Location = a.SpatialLocation
FROM @T as t
JOIN Person.StateProvince s
ON s.StateProvinceCode = t.StateProvinceCode
JOIN Person.Address as a
ON a.StateProvinceID = s.StateProvinceID;

SELECT * FROM @T;
```

Execute your new stored procedure in your SSMS query window. Examining the dataset, you can see that it has four fields:

- ▶ **Quantity**—Contains the sum of all product orders for each state
- ▶ **StateProvinceName**—Contains the full English name of each state
- ▶ **StateProvinceCode**—Contains the two-letter code for each state
- ▶ **Location**—Contains SQL geography data-typed point data for each state

The Map Wizard is aware of which fields in your underlying query contain spatial data—data stored in a `geography` (in this case) or `geometry`-typed column. In our stored procedure, we pull spatial data from `Address.Location`, so we can use those locations as points on our map of the United States. This meets the spatial data requirement for the map control.

Back in RB3, at the top of the Choose Spatial Data and Map View Options screen, notice how the Location column has already been selected as the spatial field that the map control will use. It also recognizes that Location holds point rather than line or polygonal values. Check the check box near the bottom that reads Add a Bing Maps Layer; then select Road in the Tile Type drop-down. Integration with Bing Maps will surely give your reports a visual edge over the competition. Notice the reporting term used on this check box, layer. You can think of maps as collections of z-ordered layers of information. You can add layers to illustrate regions, points, place names, pictures, icons, roads, terrain, perspective, or just about anything else you can conceive of.

Click Next; then, on the Choose Map Visualization screen, click the Bubble Map button and then click Next. On the ensuing Choose the Analytical Dataset screen, select the Choose an Existing Dataset radio button and then click on your dataset (usually, it is named `DataSet1`). Our *analytical* data illustrates relative numbers of product sales by state; that is, we are relating relative integer values to points in space. Click Next and on the ensuing Choose Color Theme and Data Visualization screen, pick a color theme you like (Ocean looks nice). Then make sure the Use Bubble Sizes check box is checked and that the Data field shown in the drop-down contains the simple expression `[Sum(Quantity)]`. This field is taken directly from the sample dataset. Check the Use Bubble Colors check box, ensure that the data field expression is the same as before, (`[Sum(Quantity)]`), and then pick a color you like and click Finish.

When you return to the RB3 UI, change the title text for the report to something more meaningful and then run the report. Afterwards, save the report and then look at it using Internet Explorer in Report Manager. No doubt, your users will be very impressed with the result, illustrated in Figure 48.33.

Returning to RB3, switch back to Design mode and then click twice on your report's map control. The Map Layers dialog (appearing against the right edge of the map) functions much like the Layers palette you may have worked with in Adobe Photoshop. It enables you to perform the following tasks:

- ▶ Show or hide layers
- ▶ View layer properties
- ▶ Add new layers (either using a wizard or manually)
- ▶ Delete layers
- ▶ Reorder layers (around the z-index)
- ▶ Zoom in or out on the map (especially useful when using Bing Map layers)
- ▶ Nudge the map in any direction



FIGURE 48.33 Viewing a map report designed with RB3 in Report Manager.

Using the context menu for each layer in the Map Layers dialog, you can also do the following:

- ▶ Add or delete layers
- ▶ Add or remove a map legend or title
- ▶ Show or hide the distance or color scale
- ▶ Cut, copy, or refresh a layer
- ▶ Set up rules for how point colors or sizes are displayed

We mentioned earlier that maps can also consume nonrelational data sources, such as ESRI spatial data files. A few great public sources of a plethora of these are available:

- ▶ From the U.S. Centers for Disease Control and Prevention, at the following URL: <http://www.cdc.gov/epiinfo/>
- ▶ From the U.S. Census Bureau, at the following URL: <https://www.census.gov/geo-maps-data/data/tiger-line.html>

Let's take a quick look at a sample map report that uses an ESRI shapefile for a country. First, obtain an ESRI shapefile of your choice and launch RB3. Then, on the Start screen, click Map Wizard and click the Create button. On the Choose a Source screen, select the ESRI Shapefile radio button, click Browse to locate your shapefile (it will have a .shp extension), and then click Next. On the Choose Spatial Data screen, add a Bing Maps layer, zoom the map to your desired size, and then click Next. Choose Basic Map on the next screen and then click Next. Choose a color theme and then click Finish. Change the title text for your map using the designer surface and then run your report. It's that simple! The result is illustrated in Figure 48.34.



"if I forget you..."

FIGURE 48.34 A map report using an ESRI shapefile of Israel, designed with RB3.

Working with Report Models in RB3

Although you cannot create report models using SSDT, RB3 can still build reports that use existing report models. If you have deployed a report model built using AdventureWorks2012, you can follow along with the next example. Otherwise, you can safely skip to the next section.

Start RB3, and when it opens (on the New Report or Dataset window), click the Table or Matrix Wizard button and then click Create. On the following screen, select the Create a Dataset radio button and then click Next. On the Choose a Connection screen, click the Browse button; then navigate to a folder in your SSRS catalog that contains a report model. Click to select your model, click Open, and then click Next.

The Query Designer (QD) supports building a query from a report model. After you add a field, if you click the Edit as Text button, you see that the QD generates a Semantic Model Query Language (SMQL) XML document for Report Models rather than a T-SQL query. The RM itself is described by a Semantic Model Definition Language (SMDL) document, and this is the same SMDL used by the Entity Framework.

The QD's explorer pane includes the same Entities and Fields areas found on the SSDT Model Designer surface. The QD also offers a helpful search feature to locate RM objects, an advanced viewing mode, as well as the capability to create and add new fields to the RM on the fly.

Drag and drop the following fields into the Design area (top right): `Product`, `ProductNumber`, `ProductModel.Name`, `ProductReview.Rating` (a variation attribute of `ProductReview.Total Rating`), `ProductReview.Review Date`, and `ProductReview.ReviewerName`. Click the Run button (the exclamation point) to verify your results, then click Next. On the Arrange Fields screen, drag any one of the fields into the Values pane and then click Next. Accept the defaults on the ensuing screen, click Next, choose a color theme, and then click Finish.

For clarity, Listing 48.8 provides a T-SQL version of the query you just built using your report model.

LISTING 48.8 T-SQL Representation of a Sample Report Model SMQL Query

```
SELECT
    p.ProductNumber,
    m.Name,
    r.Rating,
    r.ReviewDate,
    r.ReviewerName
FROM Production.Product p
JOIN Production.ProductReview r
ON r.ProductID = p.ProductID
JOIN Production.ProductModel m
ON p.ProductModelID = m.ProductModelID;
```

Returning to RB3, notice your new dataset in the Report Data pane. Using the design surface, click once on the auto-created Tablix control, click its drag handle, and then press the Delete key to remove it. On the Insert tab of the Ribbon bar, click the List control button; then draw a list data region on the design surface where the table used to be.

Click in the middle of the list to reveal its Tablix borders, right-click its detail group row (this row has three horizontal lines in its middle), and then select Tablix Properties. On the General tab, under Dataset Name, use the drop-down to select `DataSet1` (the default name of the RM-based dataset you just created), and then click OK. Returning to the Insert Ribbon bar menu, click on the Textbox control button and draw a text box across the top third of your list control. Click once on the empty area of your list, right-click

your new text box, and select Expression. For the value of the expression, enter the following and then click OK:

```
= "Product: " & Fields!Product_Number.Value & " (" & Fields!Product_Model.Value &
")" & vbCrLf & "Reviewer: " & Fields!Reviewer_Name.Value & vbCrLf & "Review Date: "
& Fields!Review_Date.Value
```

Returning to the Insert Ribbon bar menu, click the Gauge control button; then draw a gauge that occupies the bottom two thirds of the list control. Select the 180 Degrees North radial style gauge and then click OK. Right-click the gauge and select Gauge Panel, Scale Properties. On the Radial Scale Properties dialog, set the Minimum drop-down value to 0 and the Maximum to 5; then click OK. Right-click the gauge again and select Gauge Panel, Pointer Properties. In the Value drop-down, enter the simple expression [Rating] and then click OK.

Right-click the designer surface (outside the boundary of the report); then click Remove Page Footer. Change the title for your report. Then, using the Line control from the Insert Ribbon menu, draw a line at the bottom of your list (just below your gauge) to separate list item repetitions during report execution. Save your report to the SSRS catalog and then click Run.

Report Builder and Report Model Security

Security is not limited to running reports in RB3. When you save a report to the catalog, users can access it through Report Manager. A user who has permissions to view a report but doesn't have permissions to one of its fields simply will not see that column when running the report. This powerful feature is called *column subsetting* and is specific to reports using RMs.

Several resources are independently secured when you run reports in the Report Server:

- ▶ You can secure the report itself, by setting permissions on the report or inheriting permissions from the parent folder or the parent's parent, and so on, all the way up to the Report Server root folder.
- ▶ You can secure the model in the Report Server; this is similar to the way reports are secured. If a certain user is not granted permissions to a model, he or she can't see the model when RB3 starts and can't build or run reports based on it.
- ▶ You can secure the items in the model—for example, entities, fields, and relationships—in addition to securing the model itself.

Keep in mind the following security override rules for models:

- ▶ If a certain user has permissions to manage the model, this overrides the permissions set for any model items in the report. For instance, if Bob is given content manager permissions on a model, Bob sees all entities and fields in that model, regardless of the security set for model items.

- Local administrators on the Report Server machine have special permissions in SSRS:
They can view and change security for any resource stored in the Report Server.

Enabling Report Builder

RB3 relies on having a Report Server available. It uses the Report Server to load data models, run reports, and save and load them from the server. On the other hand, like all the other SQL Server 2014 services, SSRS is locked down by default. The following sections describe the changes you need to make to enable RB3 functionality.

Granting Permissions

To start, launch Report Manager, go to Site Settings, and click the Security link on the left of the page. The list of permissions that appears contains pairs of Windows users or groups and Report Server security roles. (A security role is a collection of permissions.) Click New Role Assignment and add your user or group to the `System User` role. This permission is required to run reports in RB3.

Setting Model Permissions

To run reports against a report model (a deprecated practice, although still possible), users need `Browser` permissions to that model. To set permissions, in Report Manager, locate your model, click the Security link on the hover menu, click Edit Item Security, click OK on the confirmation dialog, and add your server principal to the `Browser` role.

In addition, if you want your users to see the Report Manager home folder, you need to add their principals to the `Browser` role in the home folder. Because permissions are inherited, unless inheritance is specifically broken, the members of that group then have permission to the entire content of the Report Server.

To remove permissions on a specific folder, navigate to it using Report Manager, select Security on its hover menu, click Edit Item Security, and then remove the respective role assignments.

Management and Security

48

SSRS provides a set of useful tools for managing and securing all reporting objects. The following sections discuss tools and the overall security model in detail.

Securing Reports

As we've seen in previous examples, SSRS has a built-in, role-based security system. All operations done on the server are subject to permissions. Access to SSRS is controlled through security roles and role assignments.

A *security role* is a collection of permissions—for example, the permission to create reports in a folder, the permission to view a certain report or a folder.

A *role assignment* is a set of permissions represented by the role given to a principal on a certain report or folder in the Report Server catalog. For example, the permissions on a

folder called `Data Sources` contain the `local administrators` group with all permissions contained in the `Content Manager` role.

Permissions on folders are inherited to all items present in that folder, unless security inheritance is intentionally broken and the item is given its own permissions.

Built-in Roles and Permissions

SSRS comes with a set of built-in roles. Each role is a collection of permissions, normally used in tandem to enable a functional scenario. Following are some of the built-in roles:

- ▶ **Browser**—This role is a collection of read-only permissions that is useful for navigating the Reporting Services namespace and viewing reports and resources.
- ▶ **Content Manager**—This role is similar to an administrator on the part of the Report Server where it is granted. A person who has the `Content Manager` role can view and change any reports, folders, data sources, and resources and can read and set security settings in the folders where he or she has that permission.
- ▶ **Publisher**—This role is useful for report authors who need to create and change reports, folders, and data sources in a specified folder.
- ▶ **Report Builder**—This role can be used for granting permissions needed for editing Report Builder reports.
- ▶ **My Reports**—This security role is normally given to each user in his or her own `My Reports` folder. It gives each user of the Report Server his or her own place to publish documents on the server.

The security roles system is fully customizable. You can change or even delete existing roles, and you can also create new ones. To view and modify these roles, you use SSMS. Connect the Object Explorer to your SSRS instance and then navigate to the `Security, Roles` node. View the Properties page for any role definition to manage its permissions.

Local administrators on the Report Server machine are granted `Content Manager` permission at the root of the SSRS catalog, and no one else has any permissions. To make the Report Server accessible to users, you need to explicitly grant permissions on the folders you want to make available to them.

System Roles and System Permissions

So far we have talked about permissions only on items in the Report Server namespace: reports, folders, and data sources. The Report Server also contains a set of server-wide roles and permissions. They are called system roles, and you can access them using SSMS as described previously.

As mentioned earlier, system roles are collections of permissions, such as `View Shared Schedules` or `Execute Report Definitions`. These permissions are not specific to a certain folder or part of the namespace but are global to the entire Report Server installation. A site permission is a collection of these roles assigned to users or groups. Out of the box, local administrators on the Report Server box are, by default, given the permissions contained in the `System Administrator` role.

To open your system to users, you add Windows users and groups to the site security. As with normal roles, you can change or delete the built-in system roles, or you can create other system roles.

Subscriptions

An important advantage to having reports available on a Report Server is that you can use its subscription features to push reports to users.

You can create subscriptions using Report Manager. Start by publishing a report to the catalog. Start Report Manager, click the Details View link, hover over a report of your choice, and then click Subscribe.

The first step in the Subscription dialog is to choose a delivery mode. There are three built-in delivery methods: email, file share, and null. The null delivery, as its name suggests, doesn't deliver anywhere; however, you can use it as a way to periodically load reports into the Report Server cache. For details, see the section, "Report Execution Options," later in this chapter.

NOTE

The email delivery extension requires SMTP configuration via RSCM. Subscriptions and cache refresh plans rely on SQL Server Agent, so it must be running.

For this example, choose Windows File Share as the delivery method and fill in the file name, path, render format, server credentials, and overwrite options. The next step is to pick a schedule. You can set a schedule just for this report subscription, or you can use a shared schedule for several subscriptions. Five recurrence patterns are available for subscriptions: Hourly, Daily, Weekly, Monthly, and Once.

If your report has parameters, you also need to decide which parameter values to supply when the report is executed as part of the subscription.

One restriction in creating subscriptions is that report data sources must rely on stored credentials, whether Windows or database credentials. Integrated security and prompted credentials are not compatible with subscriptions because the actual user isn't around at runtime.

Data-Driven Subscriptions

Another way to customize report delivery is through data-driven subscriptions. A dynamic or data-driven subscription is very useful for delivering parameterized reports to a number of recipients whose email addresses are stored in a database, for instance. Unlike with a regular subscription, the recipient's settings—such as the To or CC address, the title of the email, and the parameters of the report being run—can all be based on a SQL Server query. For example, using the AdventureWorks2012 sample database, let's say you want to send quarterly human resources information reports to all managers at the company.

Suppose you have created a `HumanResources` report that takes a parameter, the login ID of an employee. If that employee is a manager, the report displays a list of all employees

reporting to him or her, together with the person's title, the base pay rate, vacation hours, and sick leave hours.

You start the Data-Driven Subscription Creation Wizard using Report Manager by clicking the New Data-driven Subscription button on the Subscription tab for your `HumanResources` report. Type a name (such as `Send quarterly HR reports to managers`) and select the email delivery extension for the subscription. Then click Next.

On the data source page, use the shared `AdventureWorks2012` data source you created earlier. Enter the query shown in Listing 48.9, which selects email addresses and login IDs for all managers.

LISTING 48.9 T-SQL for a Data-Driven Report Subscription

```
SELECT DISTINCT
    pe.EmailAddress,
    e2.LoginID
FROM HumanResources.Employee e
JOIN HumanResources.Employee e2
ON e2.OrganizationNode = e.OrganizationNode.GetAncestor(1)
JOIN Person.Person p
ON p.BusinessEntityID = e2.BusinessEntityID
JOIN Person.EmailAddress pe
ON pe.BusinessEntityID = e2.BusinessEntityID
ORDER BY pe.EmailAddress;
```

Next, bind `EmailAddress` to the `To` field in the email and `LoginID` to the `LoginID` parameter of the Human Resources report. The last step is to choose a schedule for this subscription.

Subscription and Delivery Architecture

Let's look at what happens under the covers when a subscription is created. The metadata for the subscription is validated and stored in the Report Server catalog database. A SQL Agent job is created, and the schedule specified in the subscription is saved in the SQL Agent job.

When the time comes, the SQL Agent job fires, inserting a so-called "event" row in the Report Server catalog. The Report Server matches the event with the information about the subscribers and sends notifications to the subscribers. The notifications are processed in the Report Server Windows service; the report is run and delivered to its target.

This architecture allows SSRS to scale very well with the number of subscribers and events. Because the events are recorded in the catalog database, it also allows for a scale-out configuration, so you can have a number of services process notifications in parallel, thus achieving greater scalability.

Report Execution Options

Other very useful features of SSRS are the capability to cache report content and dataset data, to display data as of a certain date, to display historical snapshots of those data, to refresh those snapshots, and to do so on a scheduled basis. The following sections describe these capabilities.

Live Reports and Sessions

By default, when a report is deployed on the server, it is configured to be run live or on demand. Every time a user clicks a report link in Report Manager, the report queries are executed. The report is processed, filters are applied, sorting is performed, and expressions are evaluated. Finally, the report is rendered to the desired format and returned to the user. The binary result of report execution is stored in a format-independent fashion in the Report Server's temporary database (`ReportServerTempDB`). This result is known as a *session snapshot*. When you page through a report or export it to a different format, the session snapshot is used to perform these operations; this way, report queries do not have to be rerun.

A session snapshot is tied to a specific user, is typically associated with a browser session, and is generally short lived (on the order of minutes).

Cached Reports

Let's assume (safely) that report queries take a relatively long time to run and the data to be displayed doesn't change very often. In that case, you can set execution options to cache the report. To do so, in Report Manager, view any report's properties (hover over it and select Manage); then click the Processing Options tab at the left of the page. Select one of the caching option radio buttons (other than the default Do Not Cache Temporary Copies of This Report). When the first user accesses the report, it is executed (as described earlier), but the resulting snapshot will, from that point on, be saved and then shared across user sessions. When a second user clicks the report link, instead of the report running again, the user gets the snapshot that was generated when the first user ran it.

There are two ways to remove a snapshot from the cache:

- ▶ **After a certain number of minutes of inactivity**—This is a “sliding” expiration, meaning that as long as users navigate to the report, it is kept in the cache. If no one has requested the report for more than the specified number of minutes, the cache is expired, and the next report request causes a live execution, which starts another cache session, and so on.
- ▶ **On a schedule**—This is useful if the data is not valid after a certain date (for example, if sales information changes every two weeks), and you don't want any cached report to show data older than this date.

Execution Snapshots

Snapshots are an extremely useful bit of functionality: They enable SSRS to fetch a report's data, pre-render the report ahead of its actual execution time, and then save that intermediate format for fast retrieval. If you configure your report to use snapshots, when the

time comes for a user to run it, it will render very quickly because the data has already been culled from the report's data sources and stored in `ReportServerTempDB`. The only downside is staleness of data, but you can adjust the snapshot creation interval to your customer's liking.

If you don't want your users to run reports against live data (for example, if the queries are prohibitively expensive or they make sense only for end-of-month sales and the report should be run only on the first day of each month for the data to be relevant), you can set the report settings to Execution Snapshot. The Report Server then runs the report on this schedule. Your users always get the data from this execution snapshot.

The main difference between execution snapshots and cached reports is that cached reports can run live if the cached data has expired; execution snapshots, on the other hand, are guaranteed to run only on the specified schedule.

Historical Snapshots

Historical snapshots are useful if you want to keep historical data for your reports. Say that you want to track all monthly sales reports from month to month. You can configure this on the Snapshot Options tab on the Report Properties window. You simply set the option Use the Following Schedule to Add Snapshots to Report History and then create a new or use an existing shared schedule.

To see historical snapshots, go to Report Manager, click the report, and then click the Report History tab. Notice the list of historical snapshots taken from the selected report (you can also create a new snapshot on demand on this screen). When you click a report on the list, you see the actual report data.

Cache Refresh Plans (CRPs)

As the name implies, a CRP is a plan that controls when cached shared dataset or report data is to be automatically refreshed. To create a CRP, view any report or shared dataset in Report Manager; then hover over it and click Manage. On the left, click Cache Refresh Options and then click the New Cache Refresh Plan button at the top of the page (if you already have a CRP, you can create a new CRP using the existing one as a baseline). You may get a confirmation dialog asking if it's okay to enable caching for the selected item. Answer in the affirmative. On the ensuing screen, enter a description for your new plan, create a new item-specific schedule, or use an existing shared schedule; then click OK (ensure that the SQL Agent service is running before doing so). If you are working with a shared dataset, your steps are complete. If you are working with a report that has one or more parameters, there is one more step to consider.

When caching a parameterized report, SSRS needs values supplied for all report parameters (at least, for those which do not except an empty value). A discrete set of cached report data is created for every unique combination of input parameter values you specify. Any errors in input render your CRP inactive.

User-Specific Data Limitations

Cached reports, execution snapshots, and history snapshots all have something in common: they each allow sharing of report data among users. This means that a given

SSRS instance does not let you use these features with reports that contain user-specific data. Per-user references in the report include the usage of `User!UserID` in the report definition, the use of integrated security in the report data sources, and the Impersonate After Connection Is Made option for SQL Server data sources.

Performance and Monitoring

SSRS includes a number of performance and monitoring tools: the trace log, execution log, event log entries for system errors, and a set of performance counters. Also, to improve the product's quality, SSRS can send error reports to Microsoft if you opt-in during Setup or by using the SQL Server Error and Usage Reporting Tool.

SSRS Trace Log

Similar to the SQL Server Database Engine, SSRS writes detailed trace and error information, useful for troubleshooting, to a log file, namely:

```
%PROGRAMFILES%\Microsoft SQL Server\MSRS12.[InstanceName]\Reporting  
Services\LogFiles\ReportServerService_[timestamp].log
```

The log file contains three types of events: errors, warnings, and informational messages. Each entry begins with a prefix that includes the current time stamp, the process and thread used, the type of event, and the message itself. Note that SSRS automatically deletes trace files older than a certain number of days.

Execution Log

If the level of detail provided in the `ReportServerService` log file is insufficient for your needs, you can take tracing to the next level by downloading the SCRUBS project (from scrubs.codeplex.com), which includes a T-SQL user database creation script, SQL Server Integration Services (SSIS) package, and a set of reports.

The SSIS package extracts execution data captured in `ReportServer.dbo.ExecutionLog` Storage to a new user database. To monitor execution, you can periodically run the package. Then you can use the provided reports to discover which reports your users most commonly run, which take the most time or are the biggest, or how many of them have succeeded or failed.

SSRS 2014 also includes three related views in the `ReportServer` database: `dbo.ExecutionLog`, `dbo.ExecutionLog2`, and `dbo.ExecutionLog3`. The first is essentially a copy of the table removed (from this SSRS edition) by the same name. The other two are similar, except that they include a new `AdditionalInfo xml` column joined from the `ExecutionLogStorage` table. `ExecutionLog3` has two columns renamed from `ExecutionLog2` (`ReportPath` is now `ItemPath`, `ReportAction` is now `ItemAction`, reflecting the fact that not just reports are accessed and/or executed).

As with trace log files, SSRS deletes execution log table entries older than two months. You can adjust the trace level for execution logging by changing settings in the `ReportingServicesService.exe.config` file.

For more complete information, see the MSDN article, “Querying and Reporting on Report Execution Log Data.”

Windows Event Log

SSRS writes configuration or internal server errors to the application event log. It also writes a number of informational messages (for example, when there are changes to the configuration files). The event log entries are marked with `Report Server*` as the source.

Performance Counters

SSRS defines a number of performance counters that you can use to measure the performance of your system. There are two performance counter categories: MSRS 2014 Web Service and MSRS 2014 Windows Service. You see a number of counters for each service and an instance for each instance of Reporting Services running on that machine. For more details, see the “Monitoring Report Server Performance” topic in Books Online.

Summary

This chapter describes the components included in SSRS as well as the product’s overall architecture. Specifically, it discusses report development, deployment, administration, security, and delivery.

If you use SSRS, you will no doubt come to appreciate what it has to offer: its power and simplicity, its open and extensible architecture, and its rich feature set. Today it is easier than ever to unlock the data from database systems and make it available to business users.

In Chapter 49, “Data Quality Services,” you learn how to develop knowledge bases to ensure the accuracy of growing data in your enterprise.

CHAPTER 49

Data Quality Services

IN THIS CHAPTER

- ▶ Data Quality Services
- ▶ What's New in DQS
- ▶ Master Data Management

As data expands exponentially, it becomes more and more critical to embed tools and logic that will help in making that data the highest quality possible. Microsoft continues to invest in the master data management and data quality areas. Master Data Services (MDS) and the Data Quality Services (DQS) capabilities with SQL Server 2014 have made great strides toward delivering on these capabilities. This chapter introduces you to general master data management and DQS concepts and introduces a small example to drive some of the points home. We usually talk of both MDS and DQS together because they form a data quality ecosystem that will enhance your data tier significantly.

Data Quality Services

Within the MDS family with SQL Server 2014 is SQL Server Data Quality Services (DQS) that complements MDS and is usable by other key data manipulation components within the SQL Server environment. This feature allows you to build a knowledge base of data rules and use them to perform a variety of critical data quality tasks, including correction, enrichment, standardization, and de-duplication of your data. Microsoft has also enhanced these data cleansing capabilities by also utilizing cloud-based reference data services that have been created by many industry data providers. This also includes the ability to do some basic data profiling to better understand the integrity and overall data quality state of your core data.

There are two primary components to DQS: a Data Quality Server and Data Quality Client. The Data Quality Server is a SQL Server instance feature that consists of three SQL Server catalogs with data quality features and functionality. The Data Quality Client is a SQL Server shared feature that is usable by anyone needing to manage, analyze, or visualize his or her data quality. These features are available as DQS Cleansing component in Integration Services and MDS data quality functionality.

What's New in DQS

Microsoft continues to stabilize DQS (and MDS). With SQL Server 2014, we see mostly just fixes and a re-deployment within the SQL Server 2014 environment. The core of MDS and DQS capabilities are pretty much still the same. With DQS you can now do matching of data before loading; in other words, before you load data, you can now confirm that you are not adding duplicate records using SQL Server DQS matching to compare two sources of data: the data from MDS and the data from another system or spreadsheet.

Master Data Management

In today's complex shared data environments, all too often data becomes the stumbling block because core data is not being managed effectively. You can distribute, dimensionize, and surface data in so many ways now, but if that data has poor data quality it is useless. Managing, or mastering, data is central to providing high-quality, high-integrity data to the information consumers of your organization and beyond. If you get this right, the whole company benefits. Mastering data starts with

- ▶ Identifying what data is core to your organization and assessing how it is currently being managed.
- ▶ Mastering data with data governance around it to enforce data policy and uniform consumption of that core shared data.
- ▶ And lastly, some type of mastering tools to aid each of the core data stakeholders; whether this is the producers of the data, the data stewards (who keep it in tip top shape), the platform owners (the data custodians; often an IT department), or the data consumers who will leverage the higher-quality data results from a proper data mastering effort.

Microsoft directly supports both the management of master data and the data quality capabilities needed to be successful. Now, let's introduce you to some of the major terms and concepts used with these capabilities. Chapter 50, "Master Data Services" explains most of the fundamental terms and concepts around the master data management framework created by Microsoft. You might want to take a few minutes in that section before proceeding too far into the DQS components.

Data Quality Services

Data Quality Services Installation

By now, you should have the DQS components installed and ready to use for this example. This is done at SQL Server instance installation time, as shown in Figure 49.1.

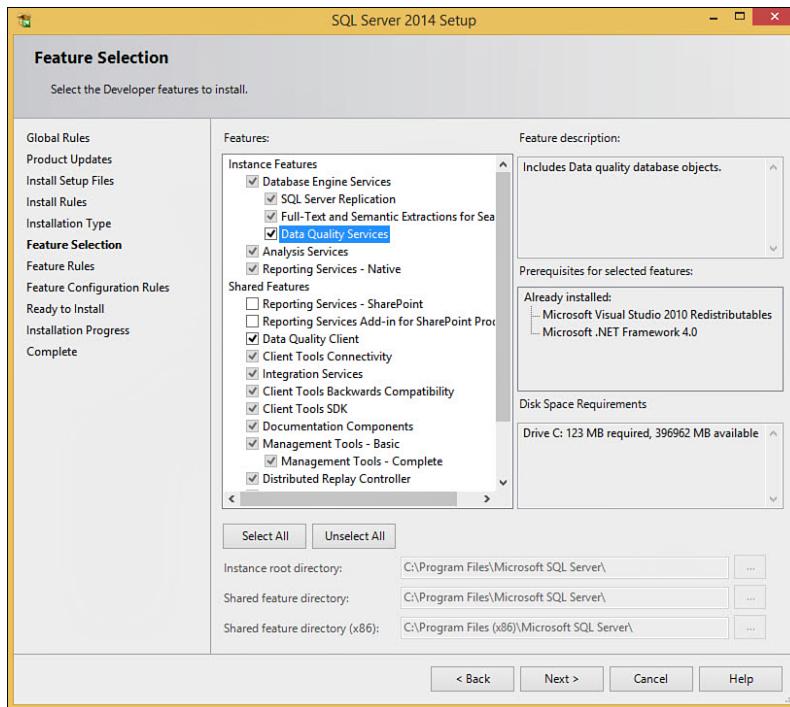


FIGURE 49.1 DQS feature selection during installation.

Once the DQS server and client components are installed, there is a second step of creating all the DQS Databases, Objects, and Accounts to use with DQS. This is done by running the DQS Server installation process (script) as shown in Figure 49.2.

Once the creation script ends, you can take a look at the target SQL Server instance and see the Databases, objects, and accounts that were just created (as shown in Figure 49.3).

Figure 49.4 illustrates that there are three primary activities related to data quality and knowledge management: knowledge discovery, knowledge building, and knowledge management. These are also done iteratively throughout the life of data (continuous data quality improvement). During this iterative cycle, you build up your knowledge base and, in turn, the knowledge base enables data cleansing, matching/de-duplication, data profiling and monitoring, and data enrichment.

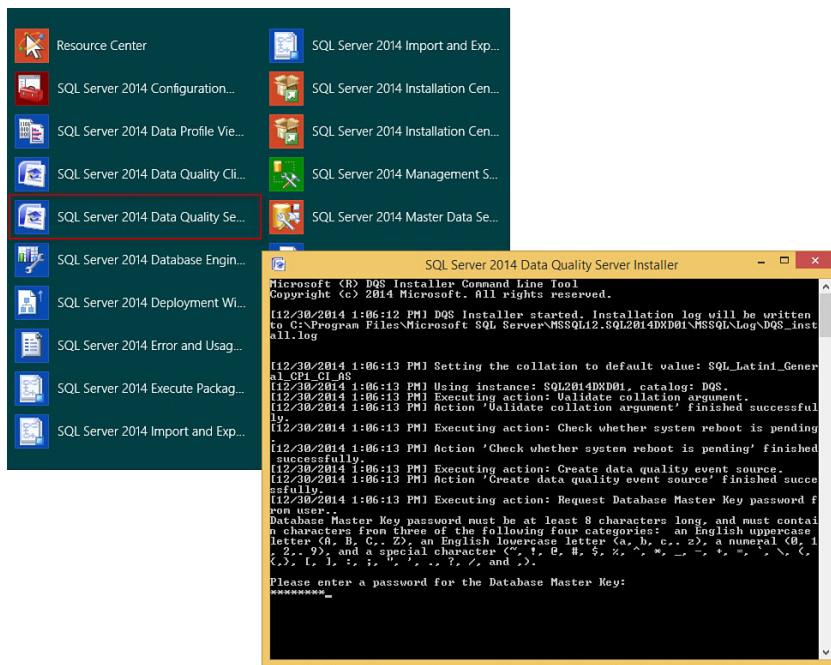


FIGURE 49.2 DQS Server objects creation script.

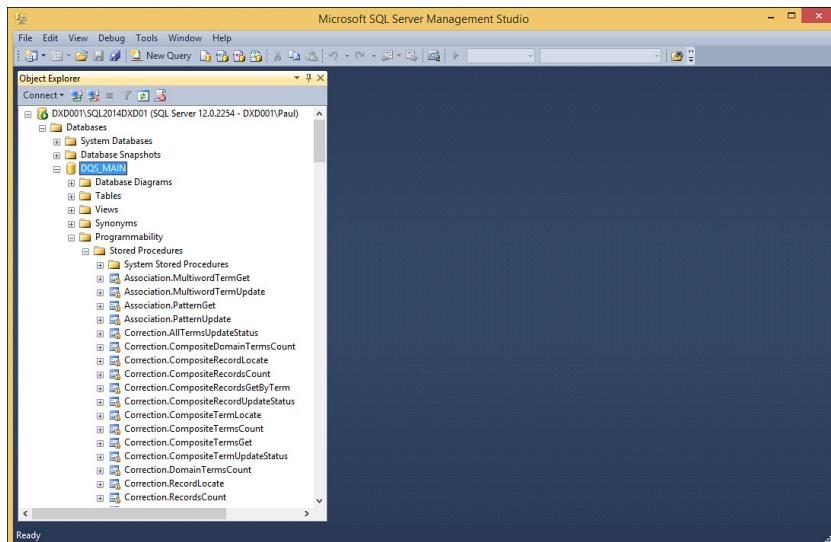


FIGURE 49.3 DQS Server objects as seen in SQL Server 2014.

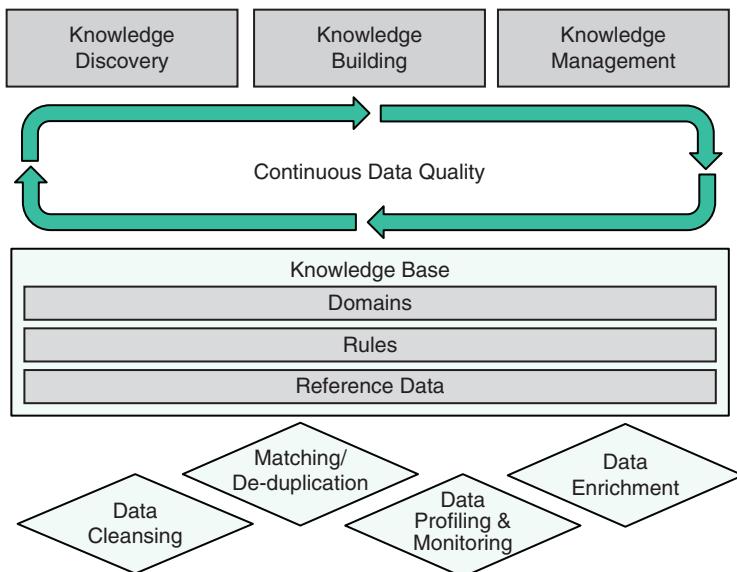


FIGURE 49.4 Iterative nature of data quality and knowledge management.

DQS is also shared with MDS in that they leverage each other. DQS capabilities like data cleansing can be tightly integrated easily. Figure 49.5 shows how they share portions of the underlying data models as well (domains, attributes, hierarchies, and so on). We would like you to begin thinking of these as a single enterprise information management platform that will continuously enhance and improve core data across your landscape.

This knowledge-driven platform has many tools and methods to discover and build up its knowledge base. The key components in this knowledge base are domains, composite domains, and matching policy. Figure 49.6 further shows that within these main knowledge base categories are several core components that help you define the complete data quality needs of your business.

In particular, domains would consist of trusted values, invalid values, erroneous errors, synonyms, term relationships, validation and business rules, survivorship rules (for de-duplication), and the ability to use third-party reference data from both internal and external sources.



FIGURE 49.5 MDS and DQS sharing of their underlying meta model.

Primary Capabilities of Data Quality Services

There are two primary components to DQS: a Data Quality Server and Data Quality Client. Both of these components work together to provide a series of robust data quality capabilities:

- ▶ **Data cleansing**—The modification, deletion, or enrichment of data that is incorrect or incomplete.
- ▶ **Matching/de-duplication**—The identification of duplicates using a rules-based process that enables you to determine what constitutes a match and what to do to remove the duplicates.
- ▶ **Reference data**—Enhance the quality of your data using the services of a third-party reference data provider.

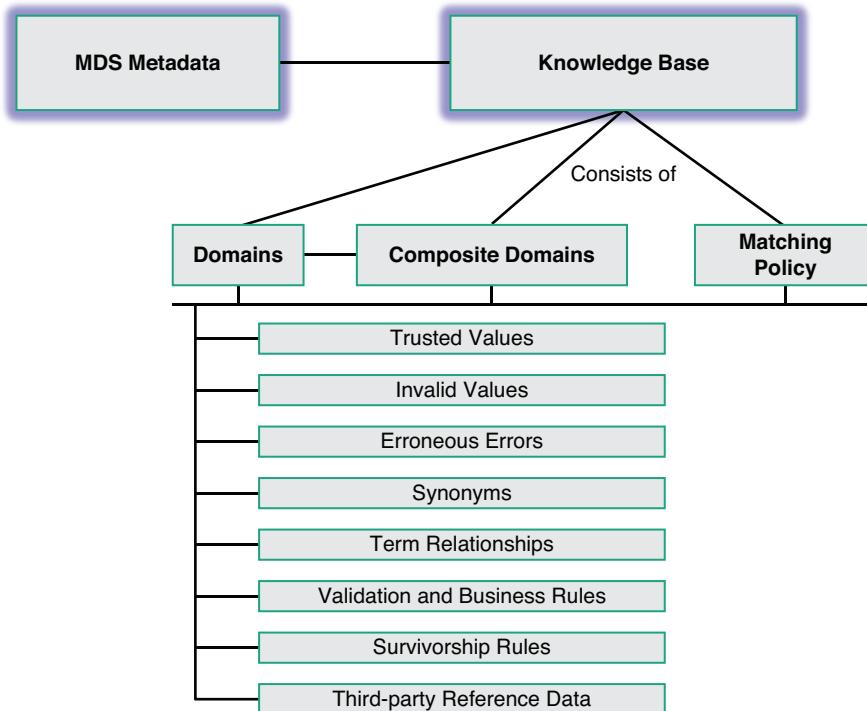


FIGURE 49.6 The knowledge base and enabling components.

- ▶ **Data profiling**—The analysis of a data source to provide data knowledge and insight into the quality of the data at every stage in the knowledge discovery, domain management, matching, and data cleansing processes.
- ▶ **Activity/monitoring**—The tracking, monitoring, and determination of the state of data quality activities.
- ▶ **Knowledge base/management**—Create data quality processes that continually enhance the knowledge about your data and continually improve the quality of the data.

In the remainder of this chapter, we cover the main DQS concepts, import domain data from an Excel file using the DQS Client, and further explain some of the other capabilities of the DQS environment, including security and administration. A full-blown example of all functionality cannot be attempted in one chapter; this truly warrants a 700-page book at the very least. Okay, let's keep digging into DQS.

Data Quality Server

Data Quality Server is implemented as three SQL Server catalogs (databases):

- ▶ `DQS_MAIN`, which includes DQS stored procedures, the DQS engine, and published knowledge bases
- ▶ `DQS_PROJECTS`, which includes data that is required for knowledge base management and DQS project activities
- ▶ `DQS_STAGING_DATA`, which provides an intermediate staging database where you can copy your source data to perform DQS operations and then export your processed data

Looking back at Figure 49.3, you can see the three primary databases that comprise DQS. Note the rich set of stored procedure functionality that most of the DQS operations are created with.

Data Quality Client

The Data Quality Client is a SQL Server shared feature that is usable by anyone needing to manage, analyze, or visualize his or her data quality. It is a standalone executable application that performs knowledge discovery, domain management, matching policy creation, data cleansing, matching, profiling, monitoring, and server administration. Many operations in the Data Quality Client are wizard driven for ease of use. To cleanse data, you have to have knowledge about the data. To prepare knowledge for a data quality project, you build and maintain a Knowledge Base (KB) that DQS can use to identify incorrect or invalid data. DQS enables you to use both computer-assisted and interactive processes to create, build, and update your KB. Knowledge in a KB is maintained in domains, each of which is specific to a data attribute (or field). The KB is a repository of knowledge about your data that enables you to understand your data and maintain its integrity. These features are also available as DQS cleansing component in Integration Services and MDS data quality functionality, as shown in Figure 49.7.

High-level capabilities of DQS include the following:

- ▶ **Knowledge discovery**—Process that analyzes samples of your organization's data to build knowledge about the data itself. After you have the results of the analysis, you can validate and enhance the KB and use it to perform data cleansing, matching, and profiling.
- ▶ **Domain management**—Process that enables creation of new domains in the KB or enables changes of the knowledge that has been generated by the knowledge discovery process. The KB consists of data domains that contain domain values and their status, domain rules, term-based relations, and reference data. You can also import or export domains very easily and even create composite domains that aggregate more than one individual domain.

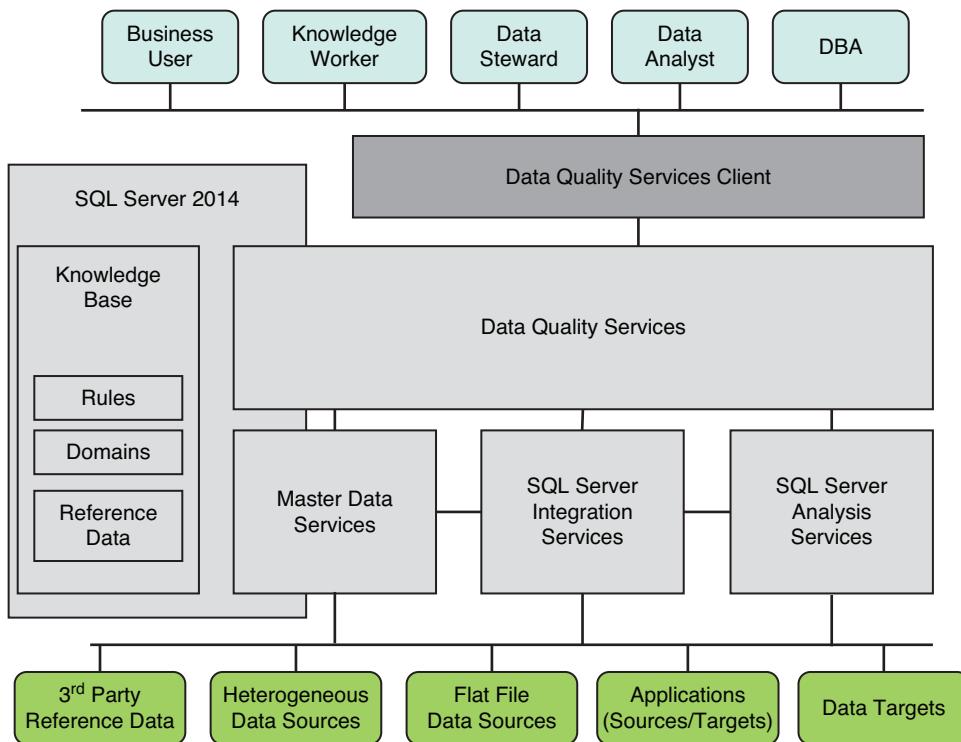


FIGURE 49.7 The DQS Server and Client complete environment.

- ▶ **Matching policy**—Defines matching rules that you want to use in the data deduplication process. The matching policy process enables you to create matching rules, fine-tune them based on matching results and profiling data, and to add the policy to the KB.
- ▶ **Reference data**—External (third-party) data used to validate, correct, and enrich your company's data. You can use the services of Windows Azure Marketplace to connect to reference data providers (that keep fresh public data with high quality), or you can use a direct connection to a reference data provider.
- ▶ **Data cleansing**—Brings many capabilities and processing together such as matching, leading value correction, and so on to cleanse the data as it is being processed from various platforms to help you standardize your data.
- ▶ **Shared DQS functionality**—Data quality functionality provided by DQS is built in to a component of SQL Server Integration Services (SSIS) and into features of MDS to enable you to perform data quality processes within those services.

Domain Management

Domain management enables the information worker (user) to dynamically change and enhance the metadata that is generated by the knowledge discovery activities. Each change that is made is done at the KB domain level.

The main features of domain management are as follows:

- ▶ Creating new domains. A new domain can be linked to or copied from an existing domain.
- ▶ Setting domain properties that apply to each term in the domain.
- ▶ Applying domain rules that perform validation or standardization for a range of values.
- ▶ Applying changes to any specific data value in the domain.
- ▶ Using the DQS Speller to check the syntax, spelling, and sentence structure of string values.
- ▶ Importing a domain from a .dqs data file or domain values from a Microsoft Excel file.
- ▶ Importing values that have been found by a cleansing process in a data quality project back into a knowledge base.
- ▶ Attaching a domain to the reference data maintained by a reference data provider, with the result that the domain values are compared to the reference data to determine their integrity and correctness. You can also set data provider settings.
- ▶ Applying term-based relations for a single domain.

Leading Value Corrections

One very strong capability is that of the leading value correction feature. This is specified as the default when you are creating a domain. Leading value correction applies to domain values that have synonyms and when the user wants to use one of the synonym values as the leading value instead of others for the consistent representation of the value.

For example, England, UK, and Great Britain are synonyms, and your company wants to use UK as the leading value instead of England or Great Britain. DQS fully supports leading value correction during the cleansing process to help you standardize your data.

Term-Based Relations

Term-based relations (TBRs) enable you to make a correction to a term that is a value in a domain. It enables multiple values that are identical except for the spelling of a common part of them to be considered identical synonyms. For example, you can set up a TBR that changes the term Co. to Company. The term Co. will be changed each time it occurs in the domain. All instances of Data by Design Co. will be changed to Data by Design Company, and the two values will be considered exact synonyms in any matching.

The Matching Process

The DQS data matching process enables you to reduce data duplication and improve data accuracy in any data source. This matching feature can analyze the degree of duplication in all records of a single data source, returning weighted probabilities of a match between each set of records compared. You can then decide which records are matches (and to what degree they match) and take the appropriate action on the source data. Matching enables you to ensure that values that are equivalent, but were entered in a different format or style, are represented the same way. Matching identifies exact and approximate matches, enabling you to remove duplicate data as you define it. You define the point at which an approximate match is in fact a match (within a range/threshold level). You define which fields are assessed for matching and which are not. And, of course, you can perform the matching process in conjunction with other data cleansing processes to improve overall data quality such as performing data de-duplication using DQS functionality built in to MDS and SSIS.

Data Profiling

Data profiling is the process of analyzing the data in an existing data source and displaying statistics about that data. This includes measurements of data quality of that data source. DQS profiling is integrated into DQS knowledge management and data quality projects.

Data profiling is always in the context of the specific process (or activity) that you are performing. You can easily determine whether source data is better after cleansing or de-duplication, and by how much. All profiling numbers refer to the number of appearances of a value, and in many cases the percent of the total, with the exception of uniqueness metrics. Uniqueness metrics refer to the absolute number of values, regardless of the number of appearances of those values.

Essentially, data profiling sheds light on two primary data quality characteristics:

- ▶ **Data completeness**—The extent to which data is present for use in its intended purpose
- ▶ **Data accuracy**—The extent to which data is in a usable state (form, consistency, and integrity) for its intended use

Data Quality Services Administration

DQS administration features enable a variety of administrative tasks such as activity monitoring and configuration. Activity monitoring primarily shows the status and state of each data quality process/activity performed within DQS and enables administrators to terminate a process. The Configuration option enables you to configure reference data service settings, set the threshold values for the cleansing and matching activities, enable/disable profiling notifications, and configure severity levels for the DQS log files.

DQS Security

DQS security is based on the SQL Server security framework (and ANSI SQL standards). A database administrator grants a user a set of permissions by associating the user with a

DQS role. Doing so determines the DQS resources and functions that the user has access to and is allowed to perform. There are four DQS roles. One is for the database administrator (DBA), who deals primarily with product installation, database maintenance, and user management. This role is utilized within SSMS and not via the DQS client application. This server role is sysadmin.

The three other DQS roles are for the other type of users of DQS: SSIS developers, information workers, business users, data stewards, and so on.

These DQS roles are as follows:

- ▶ **dqs_administrator**—Role that can do everything in the scope of the product. The DQS administrator can edit and execute a project, create and edit a KB, terminate an activity, stop a process within an activity, and can change the configuration and Reference Data Services settings. The DQS administrator cannot, however, install the server or add new users.
- ▶ **dqs_kb_editor**—Role that can perform all the DQS activities, except for administration. The KB editor can edit and execute a project and create and edit a KB. They can see the activity monitoring data but cannot terminate or stop an activity or perform administrative duties.
- ▶ **dqs_kb_operator**—Role that can edit and execute a project. They cannot perform any kind of knowledge management; they cannot create or change a KB. They can see the activity monitoring data but cannot terminate an activity or perform administrative duties.

User Management

The DBA creates DQS users and associates them with DQS roles in SQL Server Management Studio. The DBA manages their permissions by adding SQL logins as users of the `DQS_MAIN` database and associating each user with one of the DQS roles. Each role is granted permissions to a set of stored procedures on the `DQS_MAIN` database. The three DQS roles are not available for the `DQS_PROJECTS` and `DQS_STAGING_DATA` databases.

Quick DQS Client Exercise

We'll use the DQS client to create a KB, define a few data domains, and import data domain values into a domain from an Excel file to give you a good start on creating a functioning KB for data quality. For your convenience, we have provided an Excel file (named `PartnersDQS.xls`) that contains a set of partner records and some worksheets with partner domain values ready to be harvested for DQS. You can find this in the sample files and code listings folder for this book on the Web. You'll want to grab this for use in this exercise.

Now, go ahead and launch the DQS Client from the program group option of Data Quality Services within the SQL Server 2014 program group. As you can see in Figure 49.8, the DQS Client has three primary sets of functionality; Knowledge Base Management, Data Quality Projects, and Administration.

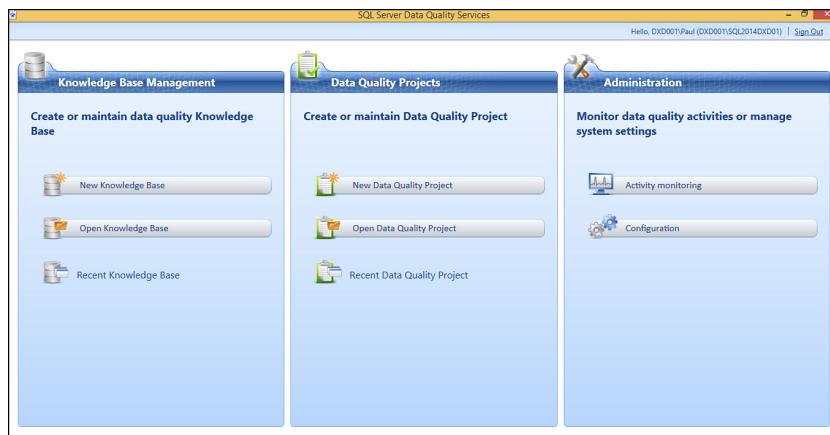


FIGURE 49.8 The DQS Client.

Click on the New Knowledge Base option and let's define a KB named "**Partners**." Figure 49.9 shows this new KB. You now want to select the Domain Management option and click Next so that you can define domains within this KB.

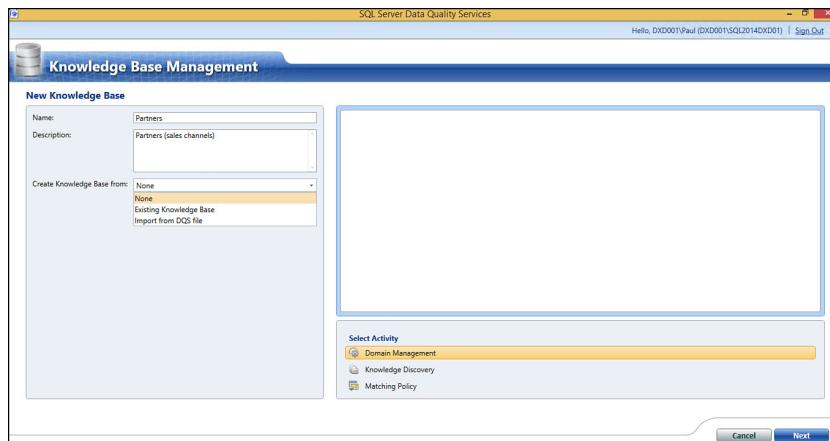


FIGURE 49.9 Selecting the Domain Management activity within a new KB.

There are no domains yet, so we need to define the ones we need for use within the Partners KB. As you remember, domains are really kind of like attributes, but they actually define the set of all possible values and characteristics that similar attributes have. These domains are what controls what valid values can go into an attribute.

For this Partners example, the domains will be as follows:

- ▶ Partner Number
- ▶ Partner Company Name
- ▶ Address
- ▶ City
- ▶ State
- ▶ Zipcode
- ▶ Country

Even the Address domain may prove to be a very valuable data domain for this company in their quest to raise data quality of their customer and partner KB. For each of these domains, you need to create a domain as shown in Figure 49.10. Use the default data types (string) and the other defaults, as well, and click Finish for each.

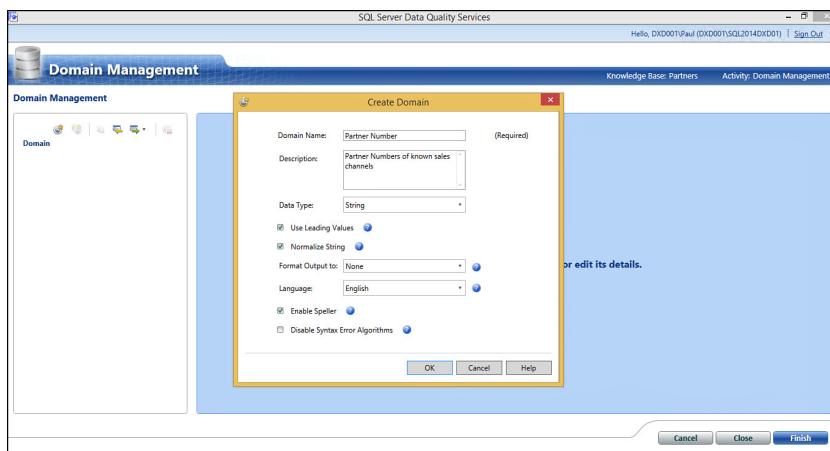


FIGURE 49.10 Creating domains for the Partner KB.

After you have created each of the seven domains for the Partner KB, you will see a list of these domains along the left pane of the Domain Management page. When you select one of the domains, a set of five tabs with different management functions is available to you. Figure 49.11 shows the Domain Management page and the Country domain selected.

These management tabs enable you to adjust domain properties, link or populate with external reference data, define domain rules, populate domain values, and create term-based relations for the domain. Now, click the Partner Company Name domain, go to the Domain Values tab, and click the Import from Excel icon over on the right side, as shown in Figure 49.12.

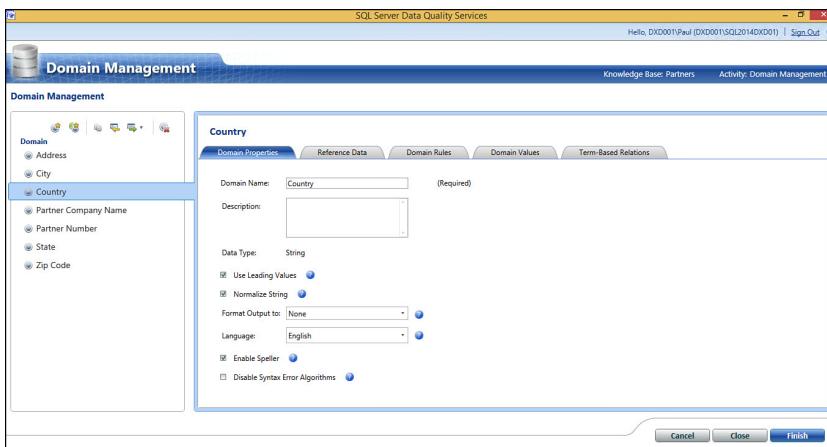


FIGURE 49.11 Selecting the Country domain from Domain Management.

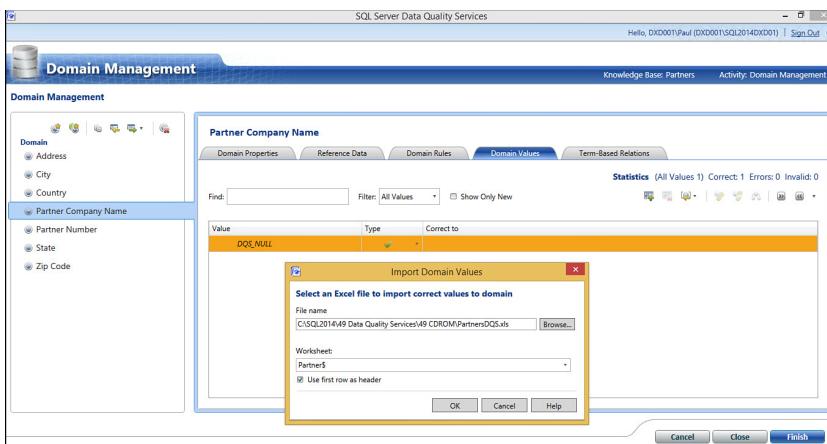


FIGURE 49.12 The Partner Company Name domain and the Domain Values tab to import values.

The intention now is to import some data values into this domain via the Excel file we provided you. These initial data values will become the KB for the Partner Company Name domain. In Figure 49.12, you should also see the Import Domain Values dialog box. Browse to the location where you have the PartnersDQS.xls Excel file, specify the Partner worksheet (`Partner$`), check the Headers Included box, and click OK.

As you can see in Figure 49.13, this domain becomes populated with the unique Partner Company Name values from the Excel file.

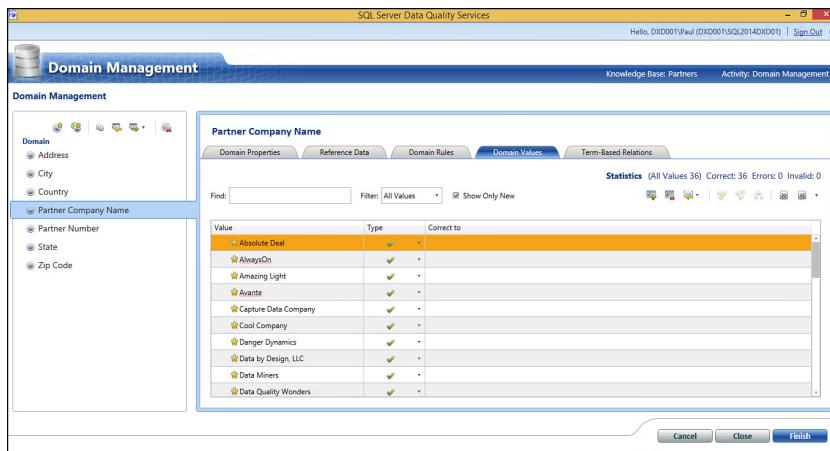


FIGURE 49.13 Results of fully populating the Partner Company Name domain from an Excel file.

Great job; these domain values are now usable in DQS and MDS.

In DQS, you can further define matching policies with your KB from the SQL database, Excel files, and other data sources (as shown in Figure 49.14).

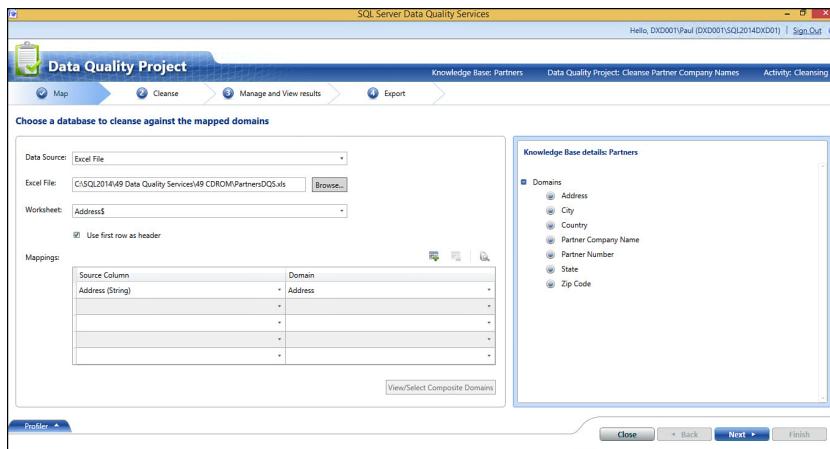


FIGURE 49.14 KB management: defining matching policies from data sources.

As you can also see In Figure 49.15, we ran the cleansing process against the values in the PartnersDQS Excel spreadsheet for addresses. The results of that run are displayed.

Figure 49.16 shows the Administration page of DQS and a list of recently processed data quality tasks. You can see our data cleansing task that we just ran (completed).

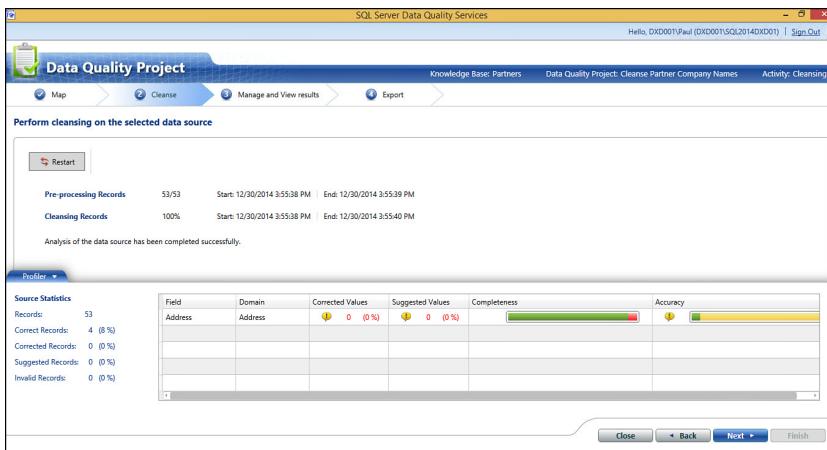


FIGURE 49.15 KB management: running the cleansing process and results.

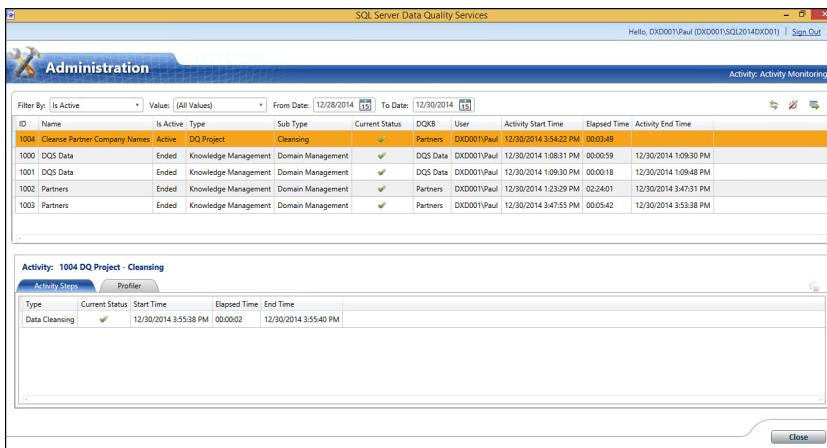


FIGURE 49.16 The DQS Administration activity/tasks page.

As you can see in Figure 49.17, the Reference Data capability within the Administration functions enables you to use third-party reference data such as from the Microsoft Azure Marketplace Data offerings.

DQS Cleansing in Integration Services

The DQS cleansing component in Integration Services enables you to perform data cleansing as part of an Integration Services package. When the package is run, data cleansing is run as a batch file. This is an alternative to running a cleansing project in the DQS Client application. In other words, you can easily include the data cleansing process within an SSIS data flow that contains other Integration Services components, as shown in Figure 49.18.

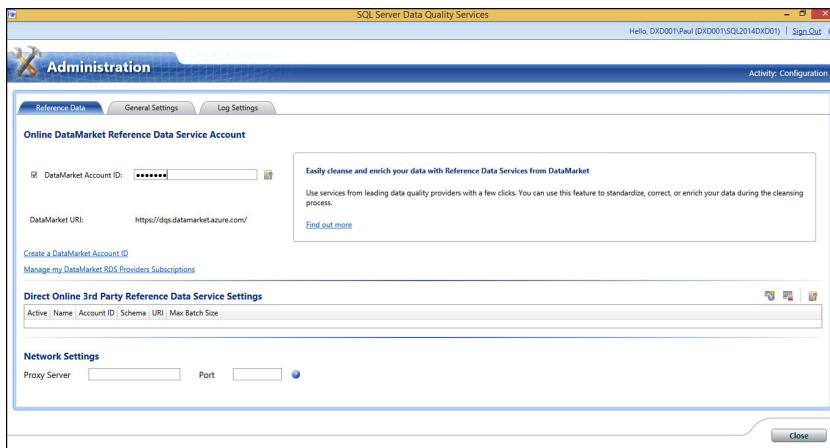


FIGURE 49.17 The DQS Administration Reference Data Settings page.

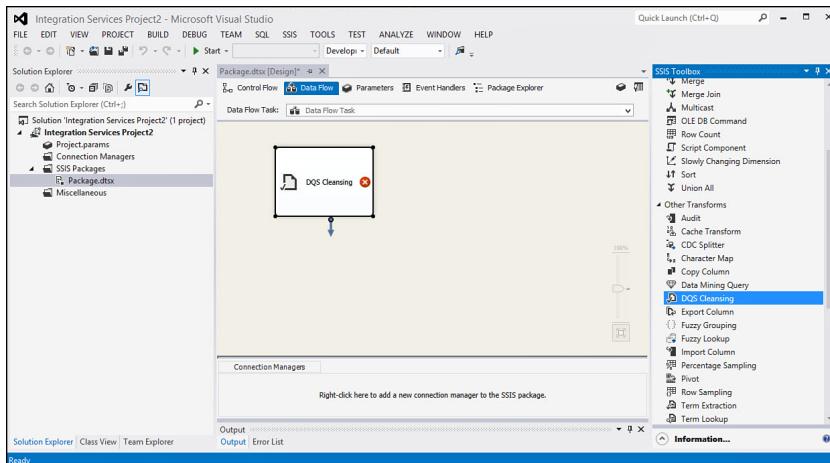


FIGURE 49.18 SSIS DQS cleansing transforms from within Visual Studio.

Summary

This chapter was jam-packed with the data quality concepts and capabilities. We really just scratched the surface on a very complex data need. Understanding the primary concepts and data model with MDS is essential to using this capability properly. For more on this, read ahead to the next chapter. You also saw how DQS can be complementary to MDS and to other environments that manage data and can benefit from data quality

checking. We have found that it is best to start small with these robust capabilities. For DQS, focus on building up the knowledge base with common domains, matching policies, and data cleansing rules. These will pay off over and over again very quickly.

The next chapter, Chapter 50, “Master Data Services,” describes all of the complex MDS terms, concepts, and installation you’ll need with DQS.

CHAPTER 50

Master Data Services

IN THIS CHAPTER

- ▶ Master Data Services
- ▶ Data Quality Services
- ▶ What's New in MDS
- ▶ Master Data Management

A big area where Microsoft continues to invest in is the master data management and data quality areas. Being able to retrieve data quickly or deliver data to any data consumer is one thing, but if the data has low quality (and integrity), it is simply useless (or garbage, putting it more bluntly). Addressing the data quality and integrity aspects of data is crucial in our emerging data-driven society. Putting into place the capabilities to target core data categories and core reference data is front and center in this quest to deliver the highest quality data to whoever needs it. Master Data Services (MDS) and the Data Quality Services (DQS) capabilities with SQL Server 2014 have made great strides toward delivering on this promise. From MDS's early years with SQL Server 2008 R2 to its current incarnation in SQL Server 2014, we have seen both functionality increases and, even more important, time-tested implementations in companies around the world. The time is now for MDS and DQS to become a part of your production data ecosystems and for you to rely on them with your crown jewels (your high-business-impact data). This chapter introduces you to the general master data management concepts and will jump into a working example of both MDS and DQS (the DQS example is in Chapter 49, "Data Quality Services").

Master Data Services

Now in its seventh year of maturity, Master Data Services (MDS) continues to provide a much-needed capability around the management of core data categories within a complex data environment. Microsoft recognized that key to delivering data to applications also meant developing

technology around improving data quality. By using Microsoft's MDS, organizations can align operational and analytical data across the enterprise and across lines of business systems with a guaranteed level of data quality for most core data categories (such as customer data, product data, and other core data of the business).

This comes in the form of explicit data stewardship capabilities complete with workflows and notifications of any business user who might be impacted by core data change. Managing hierarchies is also an important part of mastering data that has a natural hierarchical structure, such as customer hierarchies (parent company to subsidiaries and so on). Each master data change within the system is treated as a transaction; and the user, date, and time of each change are logged, as well as pertinent audit details, such as type of change, member code, and prior versus new value; and, where appropriate, the transaction log can be used to selectively reverse changes. Extensive if/then data quality rules can create default values, enable data validation, and trigger actions such as email notifications and workflows. These rules can be built by IT or business users directly from the stewardship portal. You can now work with an easy-to-use MDS Excel plug-in or work directly with MDS via MDS web services/APIs using standard programming languages (C# and so on) and with SSIS. In addition, a browser-based MDS management capability provides drag-and-drop master data management, along with extensive MDS configuration. There are also emerging third-party products such as Maestro (from Profisee Corporation) that provide a much more intuitive user interface on top of the complexities of MDS and DQS.

Data Quality Services

Within the same MDS family is SQL Server Data Quality Services (DQS) that complements MDS and is usable by other key data manipulation components within the SQL Server environment. This feature allows you to build a knowledge base of data rules and use them to perform a variety of critical data quality tasks, including correction, enrichment, standardization, and de-duplication of your data. DQS is covered in detail in Chapter 49.

What's New in MDS

No new improvements and enhancements have been added in the SQL Server 2014 version of MDS. The core of MDS is pretty much still the same.

Following are some of the top features of MDS:

- ▶ **Use Excel to manage master data**—You can now manage your master data in the Master Data Services Add-in for Excel. You can use this add-in to load a filtered set of data from your Master Data Services database, work with the data in Excel, and then publish the data back to the database. This was recently updated for 2014 and now provides a 4x performance improvement.
- ▶ **Matching data before loading**—Before you load data, you can now confirm that you are not adding duplicate records using SQL Server DQS matching to compare two sources of data: the data from MDS and the data from another system or spreadsheet.

- ▶ **Combined data loading**—Load all members and attribute values for an entity at one time. Previously you had to load members and attributes in separate batches.
- ▶ **Deployment model tools**—A higher-performance `MDSModelDeploy` command-line tool is available to create and deploy packages with data. In addition, there is a Model Deployment Wizard in the web application that is used to deploy model structures only. And, lastly, a new Model Package Editor enables you to deploy selected parts of a model, rather than the entire model.
- ▶ **Recursive hierarchies**—Creating a recursive hierarchy with three or more levels can now be supported in MDS. This enables you to build a multilevel derived hierarchy with one recursive relationship and one or more nonrecursive relationships spread over different levels. More on this later.
- ▶ **SharePoint workflow integration**—Master Data Manager web UI retrofitted to fit directly within SharePoint. Also, MDS query files (XML) can be shared directly with SharePoint.
- ▶ **Model-level security changes**—You cannot assign model object permissions to the Derived Hierarchy, Explicit Hierarchy, and Attribute Group objects.

Master Data Management

In today's complex shared data environments, all too often data becomes the stumbling block because core data is not being managed effectively. You can distribute, dimensionize, and surface data in so many ways now, but if that data has poor data quality it is useless. Managing, or mastering, data is central to providing high-quality, high-integrity data to the information consumers of your organization and beyond. If you get this right, the whole company benefits. Mastering data starts with

- ▶ Identifying what data is core to your organization and assessing how it is currently being managed.
- ▶ Mastering data with data governance around it to enforce data policy and uniform consumption of that core shared data.
- ▶ And lastly, some type of mastering tools to aid each of the core data stakeholders; whether this is the producers of the data, the data stewards (who keep it in tip top shape), the platform owners (the data custodians; often an IT department), or the data consumers who will leverage the higher-quality data results from a proper data mastering effort.

As you can see in Figure 50.1, an enterprise has many different types of data that are candidates for mastering. These are often shared across many business processes and business domains. Usually, you go after the most important data first (like, Customer first!).

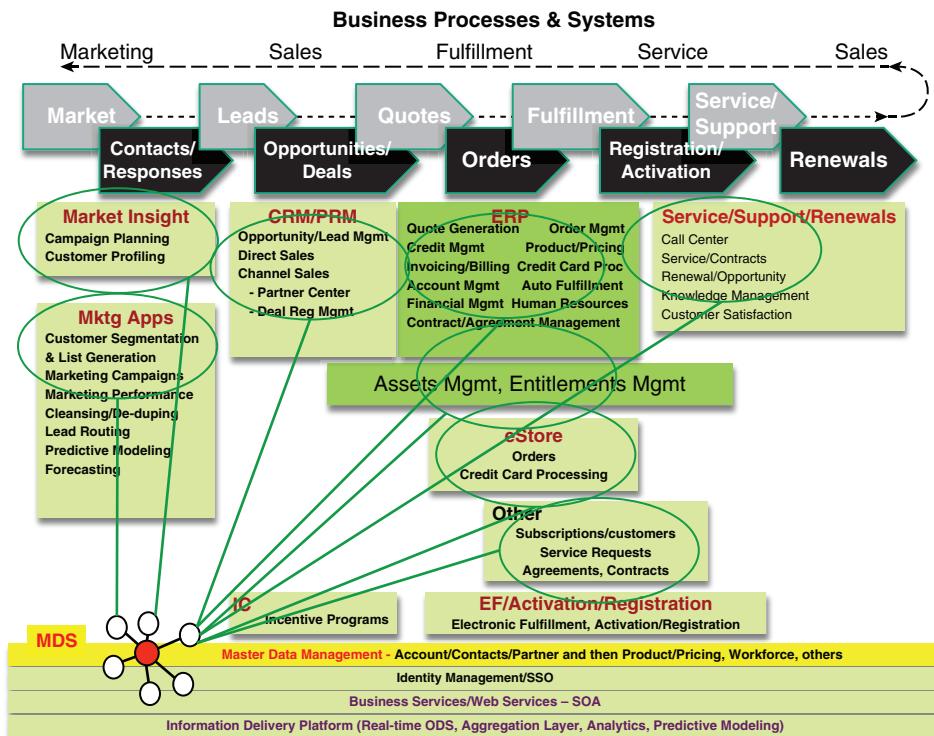


FIGURE 50.1 Enterprise view of business processes and master data candidates across the landscape.

After you have identified the data you are going to master, you must clearly define how it will be mastered, what validations and business rules must be embedded in MDS, and how this is integrated with the producers and the consumers of the mastered data. This is called *understanding the participation model*. Figure 50.2 shows an overall master data management approach (with MDS at the heart) and the different things that must be clearly understood when mastering core business data.

Microsoft directly supports both the management of master data and the data quality capabilities needed to be successful. Now, let's introduce you to some of the major terms and concepts used with these capabilities.

Master Data Services Terms and Concepts

- **Model**—Is the primary object for data organization in MDS. It defines the overall content and structure you are trying to master. Most other objects are subordinate to (relate to) this main concept. There can be one or more models. All deployments are done at the model level.

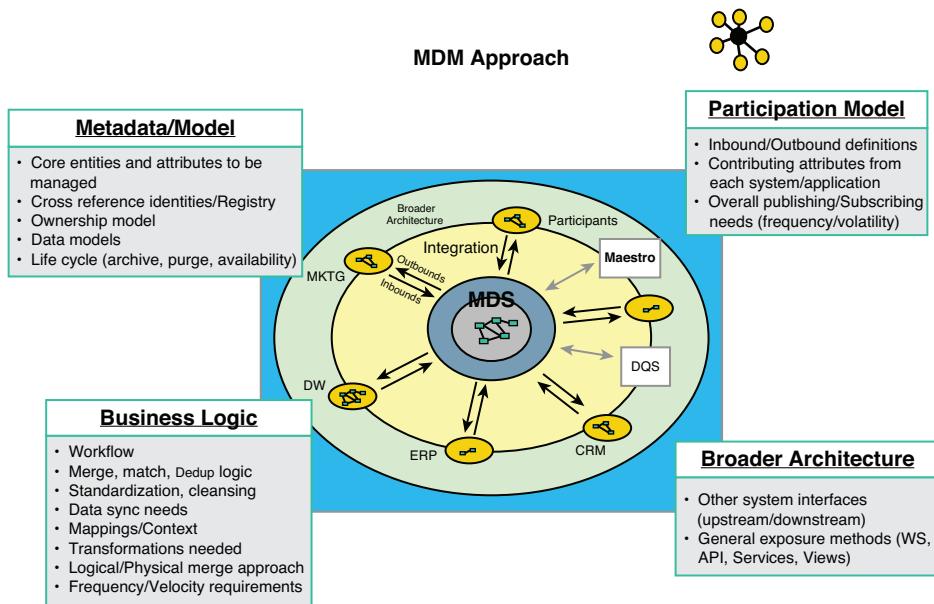


FIGURE 50.2 A master data management approach: the participation model.

- ▶ **Version**—You can create multiple versions of the master data within a model. Versions can be locked while you validate your data and committed after the data is validated. Committed versions form an auditable record of changes. Each version you create contains all members, attribute values, hierarchy members, hierarchy relationships, and collections for the model.
- ▶ **Entity**—Entities are objects that are contained in models. Each entity contains members, which are the rows of master data that you manage. You can have as many entities in a model as are needed (and that make sense). Product, Product Category, and Product Type are examples of entities in a Product model.
- ▶ **Subcategory entity**—Further categorizes entities that have distinct, non-overlapping subsets. It is a very important part of mastering data. Examples might be things like Full-Time Employees, Contingent Workers, Part-Time Employees, and so on.
- ▶ **Member**—Members are the physical master data. You can think of members as rows in a table. Related members are contained in an entity, and each member is defined by attribute values.
- ▶ **Collection**—Is a group of leaf and consolidated members from a single entity. Use collections when you do not need a complete hierarchy and you want to view different groupings of members for reporting or analysis, or when you need to create a taxonomy. Collections do not limit the number or type of members you can

include, as long as the members are within the same entity. A collection can contain leaf and consolidated members from multiple mandatory and non-mandatory explicit hierarchies. You are not creating a hierarchical structure; you are creating a flat list of members.

- ▶ **Business rule**—Is a rule that you use to ensure the quality and accuracy of your master data. You can use a business rule to validate most if/then type logic, to automatically update data, to send email, or to start a business process or workflow.
- ▶ **Subscriber view/view**—Standard view formats provide visibility (access) to all members and their attributes, all collections and their attributes, members in a derived hierarchy, and all explicit hierarchies for an entity, in either a parent/child or level format.
- ▶ **Attribute**—Objects that are contained in entities. Attribute values describe the characteristics of the members of an entity. An attribute can be used to describe a leaf member (lowest-level member), a consolidated member, or a collection.
- ▶ **Attribute group**—Attribute groups help organize attributes in an entity. When an entity has lots of attributes, attribute groups improve the way an entity can be utilized more effectively to match the business needs.
- ▶ **Domain-based attribute**—Is an attribute with values that are populated by members from another entity. Think of these as constrained lists that are often used to prevent users from entering attribute values that are not valid; to select an attribute value, the user must pick from a list.
- ▶ **Explicit hierarchy**—Arranges members into parent/child relationships from a single entity (same member types). Employee hierarchy and Product hierarchy are examples of explicit hierarchies.
- ▶ **Derived hierarchy**—Derived hierarchies are “derived” based on the domain-based attribute relationships and can span members from multiple entities. Customer Sales Region hierarchy is an example (where customer and region are different entity types and customers are at the lowest level in the hierarchy).
- ▶ **Recursive hierarchy**—Is a derived hierarchy that includes a recursive relationship. A recursive relationship exists when an entity has a domain-based attribute based on the entity itself. An organizational hierarchy is an example of this type of derived hierarchy.
- ▶ **Derived hierarchy with caps**—When the levels from an explicit hierarchy are used as the top levels of a derived hierarchy, this is called a derived hierarchy with an explicit cap (top). The explicit hierarchy must be based on the same entity as the entity at the top of the derived hierarchy; for example, a Product Category hierarchy that has varying subproducts (derived product hierarchy).
- ▶ **Annotations**—Are comments that you enter to provide details about transactions. You can annotate a transaction to provide information about why an action was taken.

- **Transactions**—Are recorded each time action is taken on a member. Transactions can be viewed by all users and reversed by administrators. Transactions show the date, time, and user who took the action, along with other details. Users can add an annotation to a transaction, to indicate why a transaction took place.

This is all a pretty complex set of terms and concepts, Figure 50.3 shows a high-level data model of these concepts (of the MDS metadata) and how they relate to each other. Keep this close to you as you navigate through the complexities of MDS.

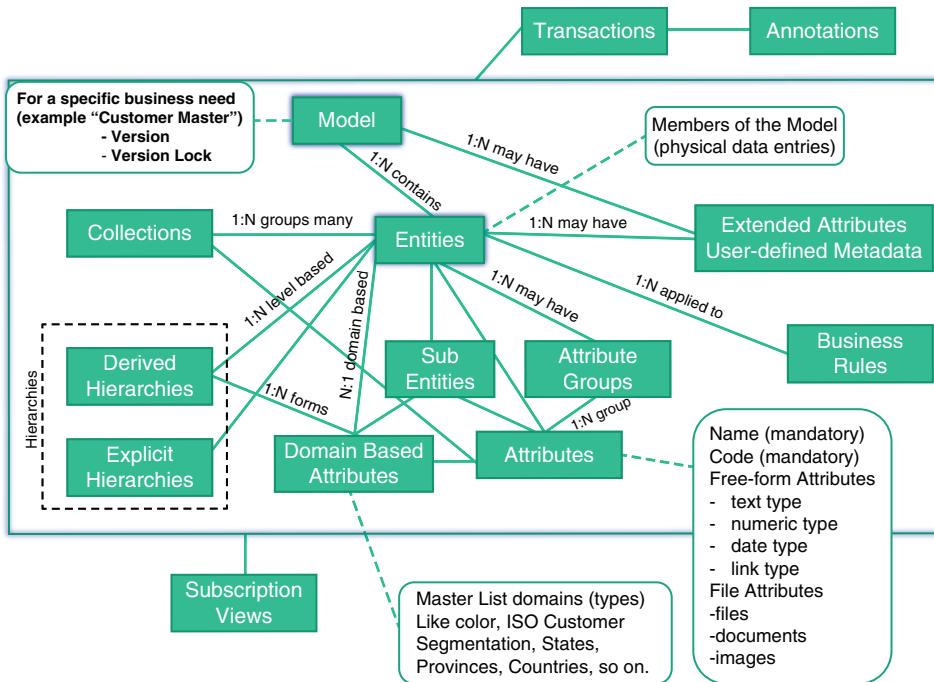


FIGURE 50.3 Master Data Services (MDS) data model.

It's now time to venture into the land of MDS and the major components that are available to master data with. We are assuming that you have successfully installed MDS (and DQS) as a part of your SQL Server instance installation. This also requires you to activate your Internet Information Services (IIS) feature on the server you are testing from. This is how you will use the MDS web-based management user interface (UI). We also recommend that you download and install the Master Data Services Add-in for Excel on any workstation (or laptop) you would use with MDS.

Master Data Services

Getting to know the MDS environment and capabilities can be a bit overwhelming. For this reason, we have provided an Excel file (.csv) that contains a Sales Territory model

(SALES_TERRITORY.csv). You can find this on the book's website (under this chapter's material). We will set out to achieve three primary goals:

- ▶ To understand a typical model within MDS
- ▶ To use the MDS manager web-based UI to define and manage an MDS model (the Sales Territory model)
- ▶ To use the Master Data Services Add-in for Excel to define and manage the same MDS model (Sales Territory model)

After we have done this for both management methods, you will have a pretty decent understanding of what a model is, how to manage it, and how to publish the results for others to use.

Sales Territory Model Description

Rather than start with a complex customer mastering exercise, we have found it best to start with something a little less scary. For this reason, we have identified a classic Sales Territory model that has a few core entities and an explicit hierarchy. Figure 50.4 shows the Sales Territory data model within the MDS concepts. It also reflects a typical global sales organization's territory requirement and is central to the success of this company's sales team.

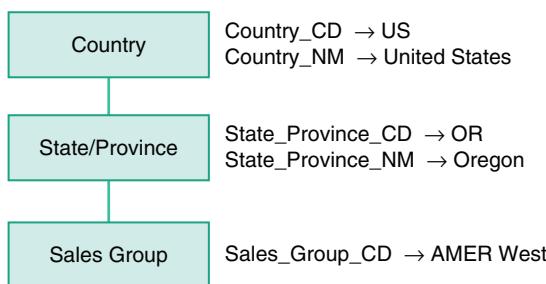


FIGURE 50.4 Sales Territory MDS model.

The primary entities in this Sales Territory are Country, State/Province, and Sales Group (also referred to as Sales Territory Group). There is a natural hierarchy that effectively organizes the sales team (sales groups) by the country and state/province they must cover. For instance, AMER South sales group must cover all of the southern states of the United States and various other places such as the Virgin Islands and Puerto Rico. The values in the Sales Territory spreadsheet will become the entity members (and can also comprise the domain attributes for Country, State/Province, and Sales Group), and the Sales Territory hierarchy organization can become an explicit hierarchy in MDS. This example will import this Excel-based file into MDS, turn it into an MDS model, and then deploy it for use (within the MDS Framework).

Master Data Services Manager

By now, you should have the MDS components installed and ready to use for this example. This is done at SQL Server instance installation time, as shown in Figure 50.5.

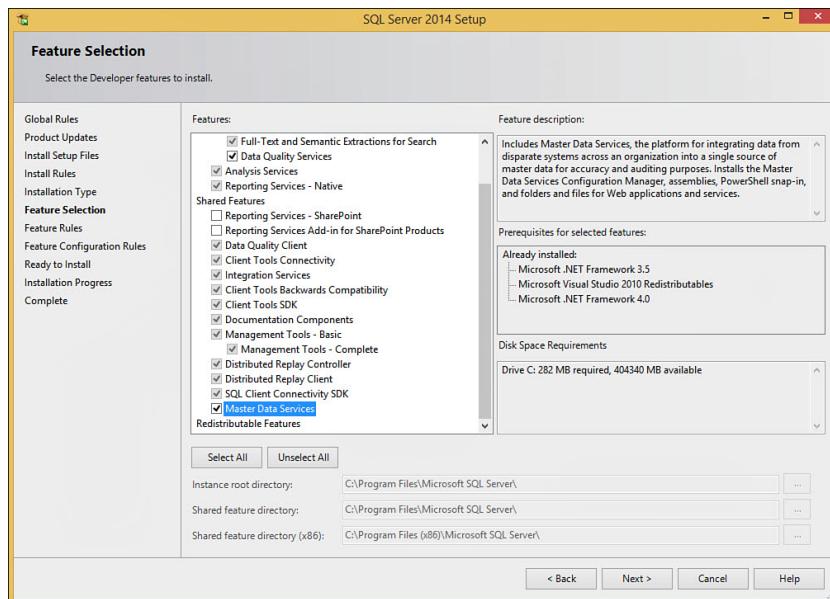


FIGURE 50.5 MDS feature selection during installation.

In addition, make sure that IIS is enabled on your machine. Figure 50.6 shows IIS running but no MDS websites installed yet. We'll use the MDS Configuration Manager to create a connection to the SQL Server instance we want to use for MDS and to configure the MDS Web Manager to be served up using IIS.

Okay, now we'll just have to start up the MDS Configuration Manager option from within the MS SQL 2014 program options. Figure 50.7 shows the MDS Configuration Manager and the prerequisite checks that are done when it is started up. In particular, it shows that the Windows PowerShell and the IIS features are available and which versions of these are installed. If any of these checks aren't passed, you will have issues with MDS and MDS Manager. The figure below shows a warning on the IIS component that is a known bug with the MDS install. The most problematic failures with the MDS configurations usually have to do with the IIS feature, not having full administrator permissions to do all the actions and making sure you have all the .NET, HTTP, and SVC handler features activated. There are a couple good community forums that describe all of the things that can go

wrong with the IIS and the MDS install (sorry, but there are often many issues). The good news is that once these are resolved, the MDS platform becomes rock solid. A great place for troubleshooting both 2012 and 2014 issues is:<http://social.technet.microsoft.com/wiki/contents/articles/2390.troubleshoot-installation-and-configuration-issues-master-data-services-in-sql-server-2012.aspx>

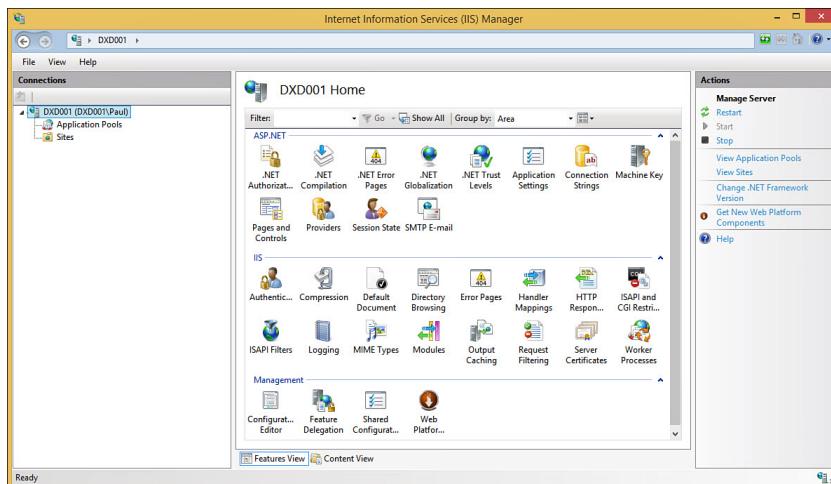


FIGURE 50.6 IIS feature enabled and ready to use by MDS.

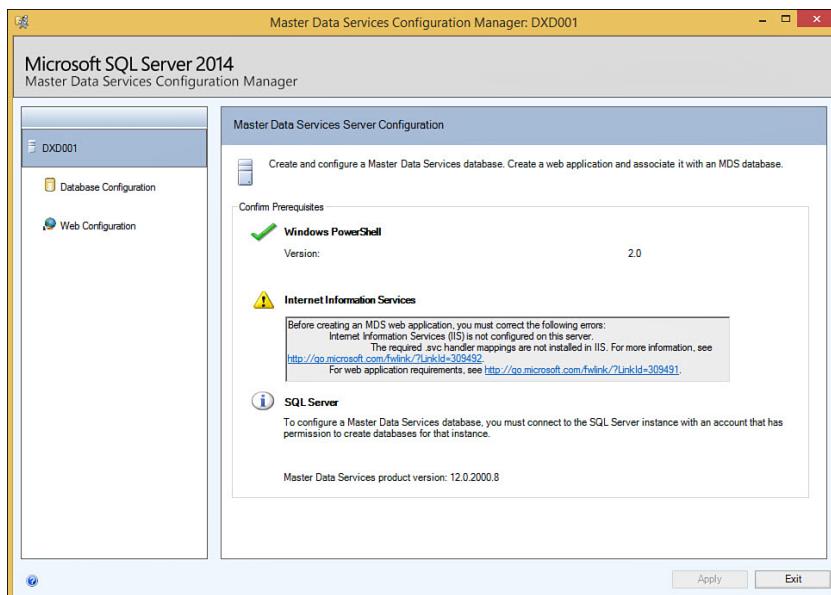


FIGURE 50.7 MDS Configuration Manager and prerequisite checks.

Now, go ahead and click the Database Configuration option in the left pane and then click the Create Database button option on the right side, as shown in Figure 50.8. This will start the Create MDS Database Wizard. If you already have an MDS database created and are trying to update or change something using the Configuration Manager, choose the Select Database option instead. You have other options there, as well, such as Repair Database and Update Database. We do not cover those options here.

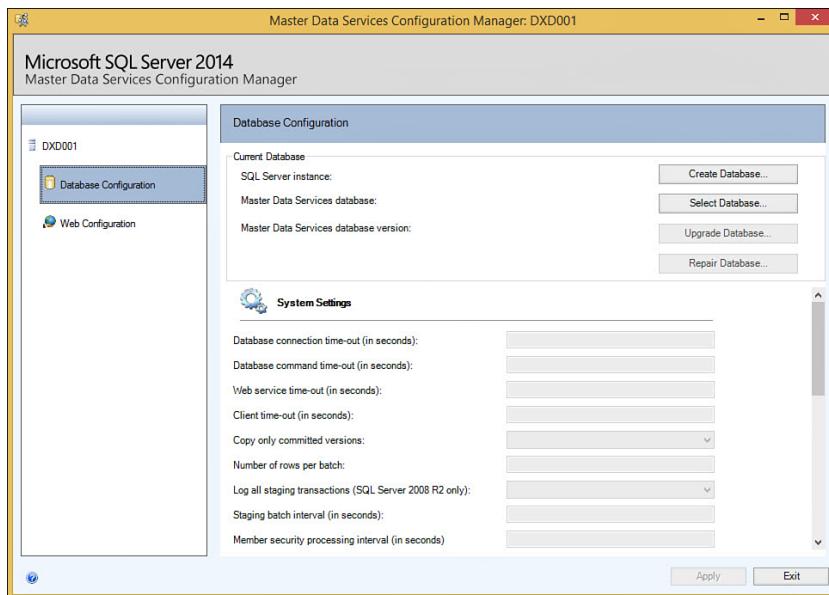


FIGURE 50.8 MDS Configuration Manager: creating an MDS database option.

For our example, we'll create an MDS database that will hold our Sales Territory model (and others) on a local SQL Server instance and will specify an MDS database name of `MDS`.

Figure 50.9 shows each of the steps performed during the Create Database Wizard (from left to right, top to bottom), including the final verification step.

When the wizard finishes, you will see the full database configuration information (the MDS database version, systems settings, and all the configuration options for this MDS database), as shown in Figure 50.10. We will keep the Copy Only Committed Versions option marked as Yes, for now, but as you work with your MDS configuration, you might want to flip this back to No to be able to make intermediate versions of MDS models (that aren't in the committed status yet). The rest of these options defaults are fine.

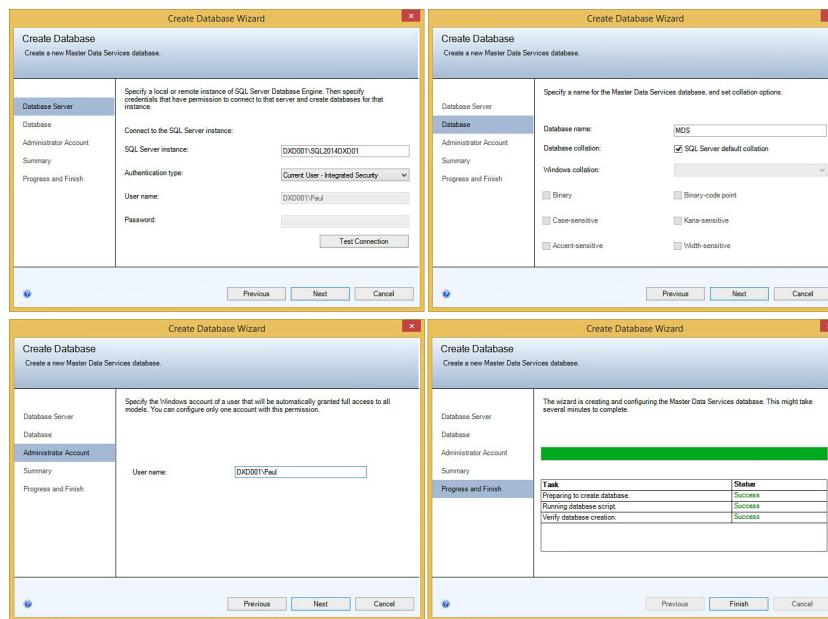


FIGURE 50.9 MDS Create Database Wizard steps.

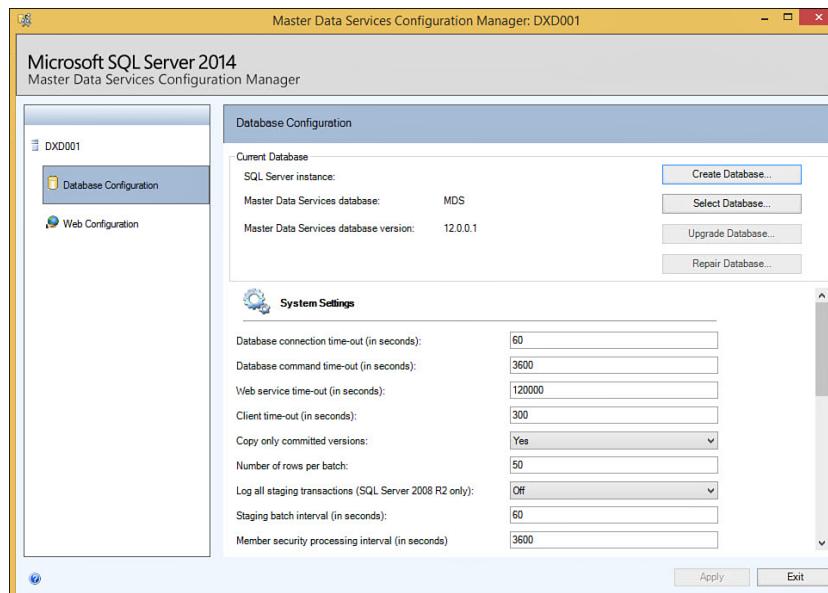


FIGURE 50.10 MDS database configuration creation finished.

NOTE

You can change the user account that is designated as the MDS system administrator by executing the [mdm].[updSecuritySetAdministrator] system stored procedure from the MDS database you have created. Watch out, though; the former system administrator user account gets deleted!

```
USE MDS
Go
EXEC [mdm].[updSecuritySetAdministrator] @UserName='DXD001\Paul',
@SID = 'S-1-5-21-1935655697-515967899-682003330-262929', @PromoteNonAdmin = 1
Go
```

Where @SID is the SID value from [mdm].[tblUser] table for the new administrator.

Once an MDS database is created, we can do the web configuration and use the MDS management services from any browser in your network. This is extremely useful; no client installation is needed for MDS administrators and data stewards. As you can see from SQL Server Management Studio (SSMS), Figure 50.11 shows the MDS tables that were just created with the MDS database during the configuration steps. As you can see, these are all mdm.xxxx named tables (under the MDM schema), each for its own purpose (EN is for Entities and so on). You do not ever update these tables directly. However, they can be queried for your information purposes.

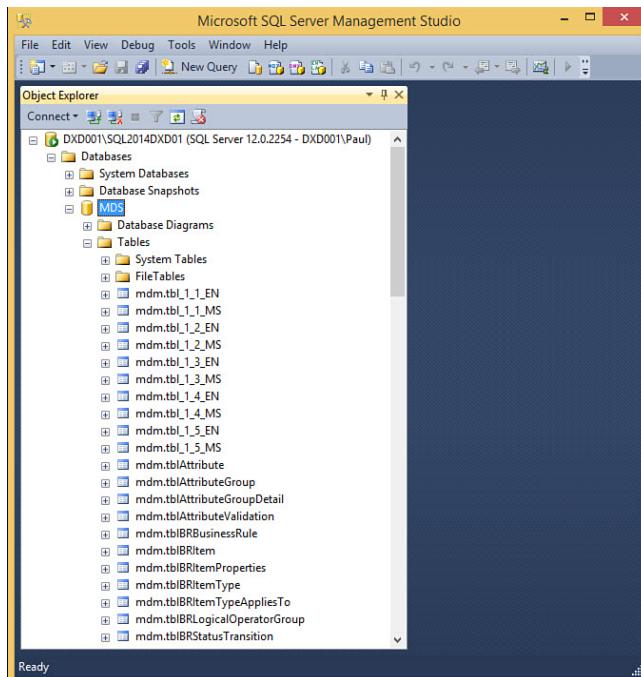


FIGURE 50.11 MDS database and internal tables for master data management.

Now, choose the Web Configuration option in the left pane of the MDS Configuration Manager and select the Create New Website option. You'll be asked to name an internal website for MDS and specify the binding information used to access this site. This will include specifying an MDS website name (just use MDS), a port (something like 80 or 90) that this site will be bound to and is unique within your network, and an MDS application pool (something like MDS application pool, MDS application pool 1 or 2) that is used by the web MDS Manager application, as shown in Figure 50.12.

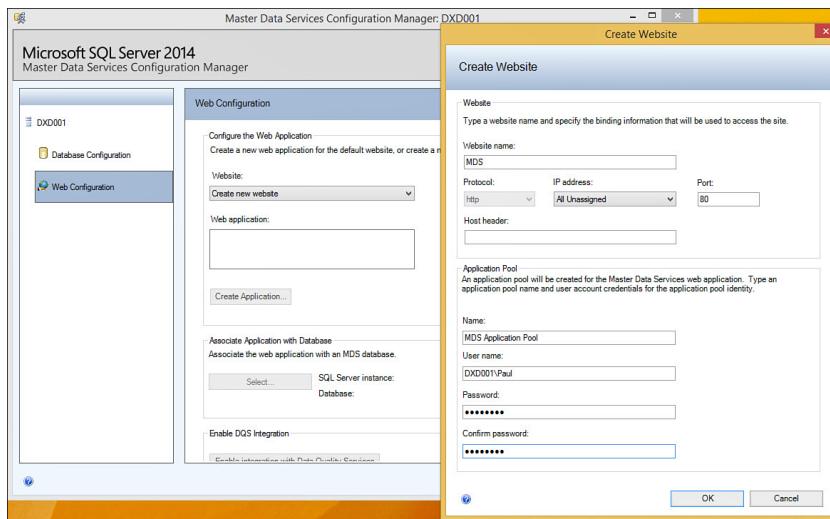


FIGURE 50.12 Creating the MDS internal website.

Once created, you will associate the MDS database with the website, as also shown in Figure 50.12. You need to click the Apply button at the bottom right. MDS Configuration Manager will now ask whether you want to launch the site. For now, choose No. We will bring this up directly in a browser with the full URL value and port later.

If you take a quick peek at IIS (feature view), you will see that an MDS website has been created and has been bound to a port and the MDS application pool (see Figure 50.13). You can also launch (browse) the site, stop it, or restart it from IIS (Manage Web Site and Browse Web Site options in the IIS pane on the right).

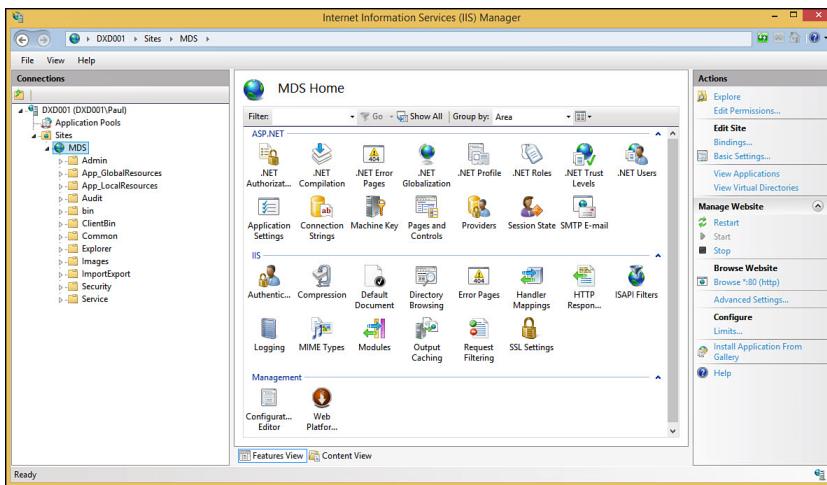


FIGURE 50.13 IIS Manager and the new MDS website bindings.

Once completed, you can bring up the MDS Master Data Manager within your browser to manage the MDS models, members, and hierarchies. We also suggest that you enable the DQS integration as well (as you can see in the lower-left portion of the web configuration pane back in Figure 50.12). This will provide access to some DQS capabilities from within MDS when (and if) you intend to leverage this capability. Right now, you are essentially ready to utilize MDS; however, our example later in this chapter will use Excel and the Master Data Services Add-in for Excel for data manipulation. Let's go ahead and install this on your machine now.

Master Data Services Add-in for Excel

Assuming you have Excel installed on a machine for you to use, you simply choose the option from the MDS Master Data Manager website you have just deployed (shown in Figure 50.14), or pop out to Microsoft.com and locate the Master Data Services Add-in for Excel for download (<http://www.microsoft.com/en-us/download/details.aspx?id=42298>). Go ahead and install this now.

If you had chosen to launch MDS Master Data Manager site directly from the Configuration Manager, you would have received an intermediate web page (something like <http://mac4cz130rqv:592/gettingstarted.aspx>) that has some references on getting started with MDS, known issues, and an option to deploy some sample models and data to play around with. Once you get a little better at MDS, go back and review those if you want; they will make a lot more sense later.

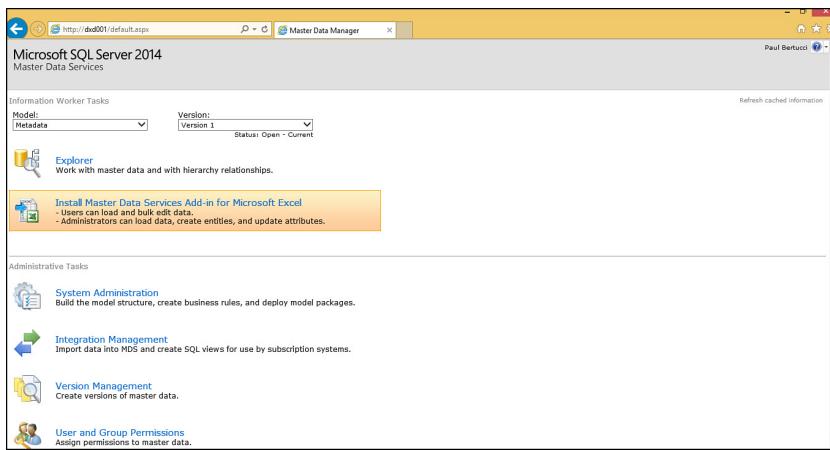


FIGURE 50.14 Newly deployed MDS Master Data Manager website and the Master Data Services Add-in for Microsoft Excel option.

Using the MDS Master Data Manager

From your browser, bring up the MDS Master Data Manager website you have configured and deployed by typing in the URL that was created. This usually consists of the machine name (or localhost) and the port number that you provided during configuration. An example is <http://dxd001:80/default.aspx>, where dxd001 is the machine name and 80 is the assigned port number. Often, the port number isn't necessary if only one MDS service is running on a single machine (in which case the URL would be simply: <http://dxd001/default.aspx>). Remember, this site has been linked (bound) to your MDS database already and you can now manage MDS entries through the website. As a part of our example, we'll use that Excel file that has some valid master data member data for the Sales Territory (`SALES_TERRITORY.csv`). This file was actually the primary way a company was managing their sales territory values for years. We'll quickly make this a centralized capability that can be shared by the entire company with confidence and ease of management.

First we need to create a model within MDS to contain the entities and member values for our Sales Territory example. Once created, we'll upload the entries from the Excel file (`SALES_TERRITORY.csv`) to our model in the MDS database and confidently manage these from MDS.

Again, starting from the MDS Master Data Manager home (<http://dxd001/default.aspx> in this example), click the System Administrator option, as shown in Figure 50.15, so that you can create a new model to contain the Sales Territory entities, attributes, and members (values).

Figure 50.16 shows the Add Model option in the MDS Master Data Manager. We'll create a new model named `SALES_TERRITORY`. Again, we need to create a model first before we can add entities to it. The model is the central concept within MDS.

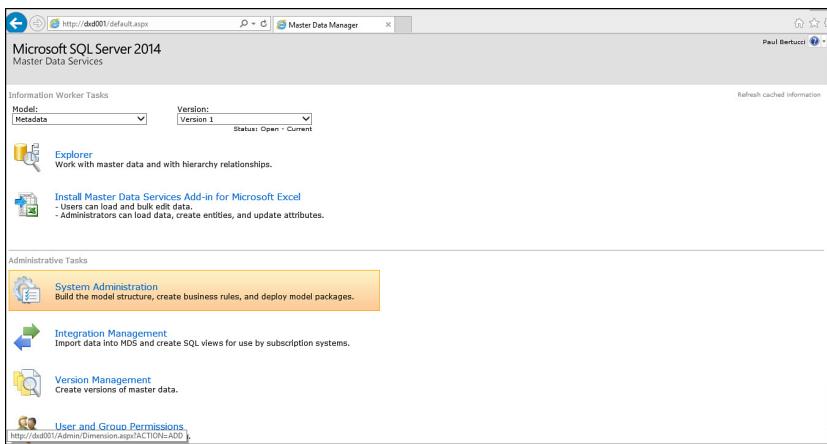


FIGURE 50.15 MDS Master Data Manager and the System Administration option to create a new model.

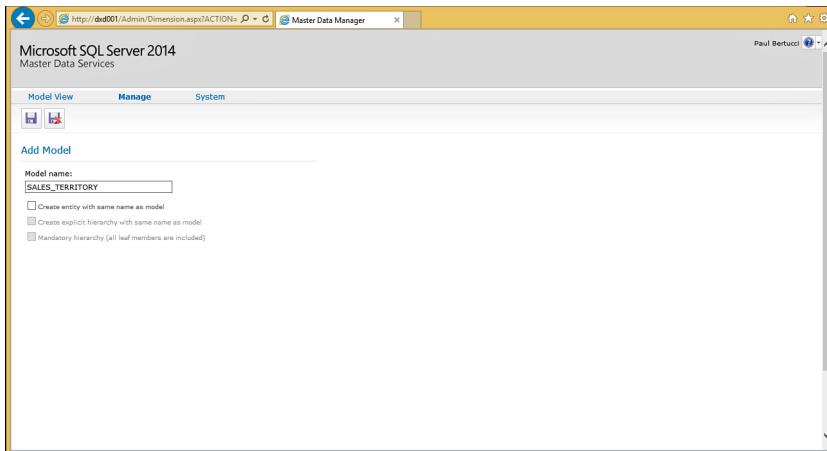


FIGURE 50.16 MDS Master Data Manager Add Model: SALES_TERRITORY.

After the model has been created, the Model Maintenance page appears (see Figure 50.17). Basically, the model is empty, but it is ready to be used now. We will do most of the buildup of entities, attributes, and member values from the Excel add-in. This will include uploading (publishing) the member entries (values) from the sales Territory.csv Excel file.

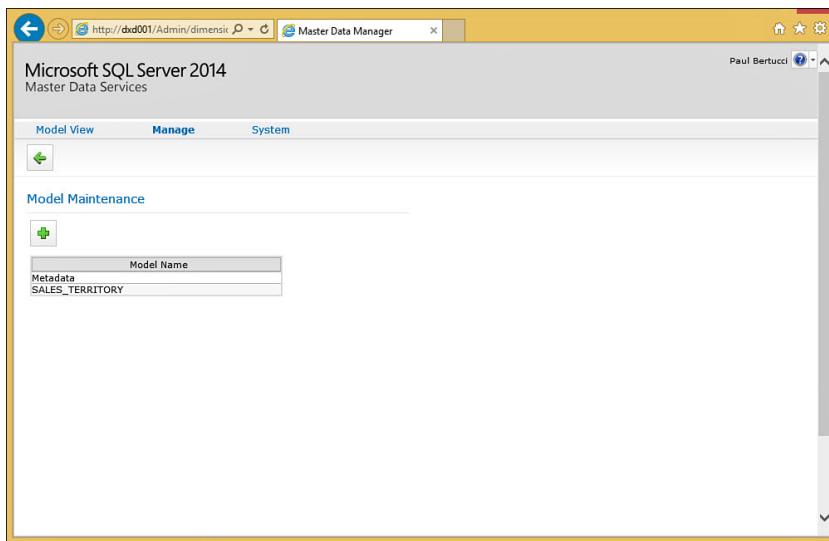


FIGURE 50.17 MDS Master Data Manager Model Maintenance page.

Using the Master Data Services Add-In for Excel

When you open Excel for the first time after you have installed the MDS Excel add-in, you will notice a new option at the top bar called Master Data. Figure 50.18 shows this new Excel option and also shows how you will have to establish a connection to the MDS database. Go ahead and select this option. The next dialog box (Select a Connection) will prompt you to either make a new connection or connect with an existing connection. We'll be adding a new connection to the MDS database and server.

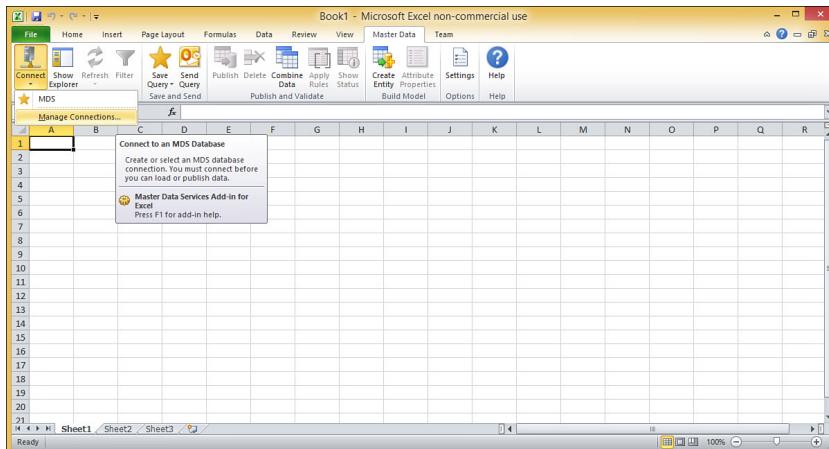


FIGURE 50.18 Excel and managing connections to the MDS database.

The next dialog box prompts you for the connection information for the MDS server (see Figure 50.19). Here you put the same URL information you did before to bring up the MDS Master Data Manager page, except that you do not add the `default.aspx` part of that URL. This would be something like `http://dxd001`. This is what MDS is known as from the network service point of view.

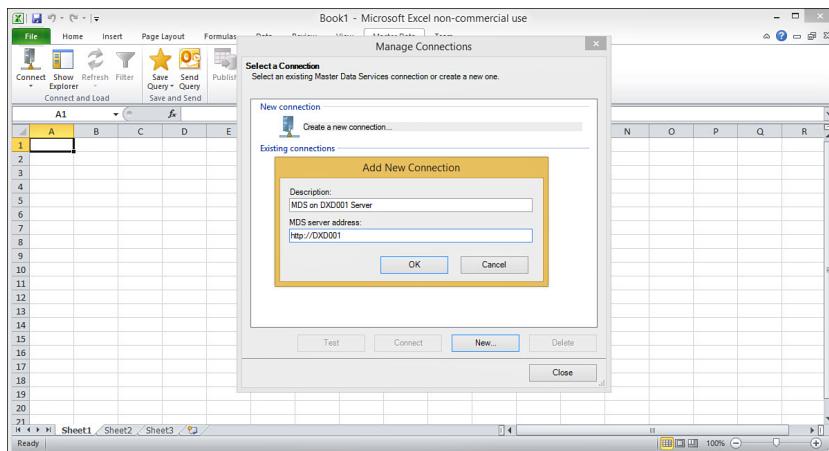


FIGURE 50.19 MDS Add New Connection and Server Address.

After you have created this connection (you can test it quickly with the TEST button), it will appear in the existing connection list. Select it for use and Excel will now be directly connected to our MDS server and database just like a client application. Figure 50.20 shows Excel and the Master Data Explorer (with the MDS server connection) at the far-right pane in Excel. You can use the pull-down Model list to locate the `sales_Territory` model we just created back in MDS. Click it so that it becomes the current model for our use.

Okay, now let's open the `Sales_Territory.csv` file with Excel as the active worksheet (as shown in Figure 50.21). The data has column headers that will become the attributes (`Country_CD`, `Country_NM`, `State_Province_CD`, `State_Province_NM`, and `Sales_Group_CD`). The data itself will be the member values for each attribute within the entity. The data value `N/A` means that there is no `state_province` value for the entry. (Except for Canada, most countries in our file don't have these.)

Now, the magic will happen. Highlight the rows and columns that contain the column headers and all the member data and click the Create a New Entity in the MDS Database option (as shown in Figure 50.22).

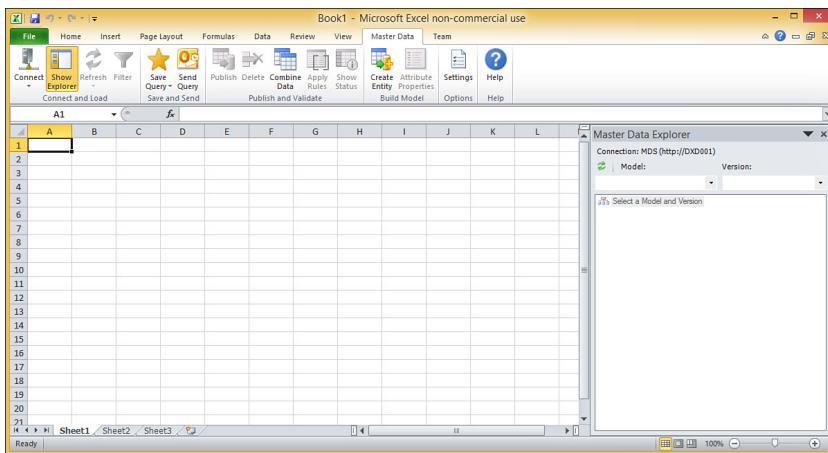


FIGURE 50.20 Master Data Explorer within Excel.

SALES_TERRITORY - Microsoft Excel non-commercial use				
A	B	C	D	E
1 COUNTRY_CD	COUNTRY_NM	STATE_PROVINCE_CD	STATE_PROVINCE_NM	SALES_GROUP_CD
2 AL	Albania	N/A	N/A	EMEA
3 AT	Austria	N/A	N/A	EMEA
4 AU	Australia	N/A	N/A	APAC ANZ
5 BA	Bosnia and Herzegovina	N/A	N/A	EMEA
6 BE	Belgium	N/A	N/A	EMEA
7 BG	Bulgaria	N/A	N/A	EMEA
8 CA	Canada	AB	Alberta	AMER West
9 CA	Canada	BC	British Columbia	AMER West
10 CA	Canada	MB	Manitoba	AMER Central
11 CA	Canada	NB	New Brunswick	AMER East
12 CA	Canada	NL	Newfoundland and Labrador	AMER North East
13 CA	Canada	NS	Nova Scotia	AMER North East
14 CA	Canada	NT	Northwest Territory	AMER West
15 CA	Canada	NU	Nunavut	AMER West
16 CA	Canada	ON	Ontario	AMER Central
17 CA	Canada	PE	Prince Edward Island	AMER North East
18 CA	Canada	QC	Quebec	AMER East
19 CA	Canada	SK	Saskatchewan	AMER West
20 CA	Canada	YT	Yukon Territory	AMER West
21 CH	Switzerland	N/A	N/A	EMEA

FIGURE 50.21 Sales_Territory data in the active Excel worksheet.

Figure 50.23 shows the Create Entity dialog, the range of columns and rows within the active worksheet in Excel where the data is being published from, the target model (Sales_Territory), the default version (Version 1), the entity name we want this to be captured under (Sales_Group), and that we want MDS to generate internal MDS code values automatically for the member entries. These will be visible (and are the internal reference) for each member type and member entry (data value) in the entity. Once this data is published to MDS, these internal code reference values will show on the far left of the Excel worksheet.

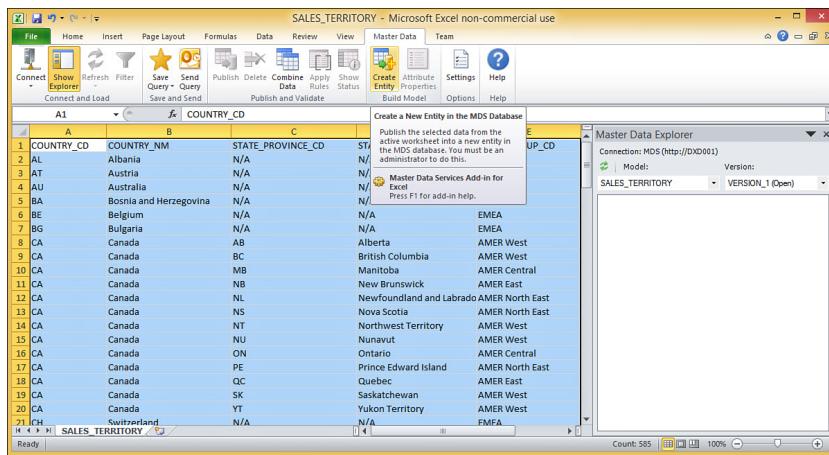


FIGURE 50.22 Create a New Entity in the MDS Database option.

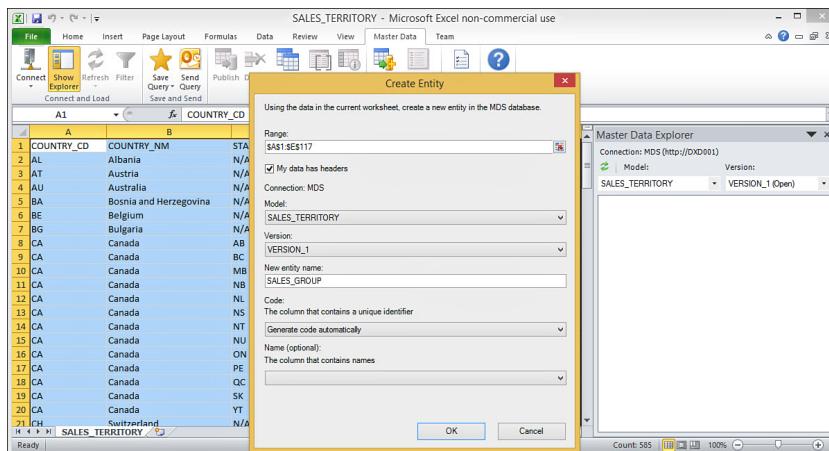


FIGURE 50.23 Create entity range, model, and new entity name.

Figure 50.24 shows the results of the create entity and data publish process. Notice an MDS connection line appears in line 1 of the worksheet, and on line 2 the column headers are now all pull-down lists. These column headers have become the attributes in MDS. There are also two new columns on the left of the worksheet that are used by MDS (Name and Code columns). These (not always both) are the internal values that were auto-generated by MDS. All of your member data is now contained within the MDS database and can be edited (modified) and published as a corporate information asset from this point forward.

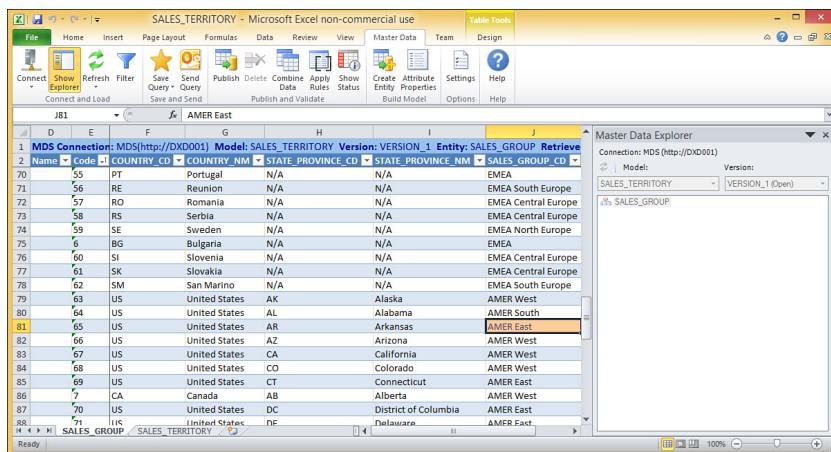


FIGURE 50.24 MDS managed worksheet from within Excel.

As you can see, about halfway down in the worksheet (line 81), we are also already changing the Sales_Group_CD value for the state of Arkansas from AMER Central to AMER East (the cell will also change to an orange color). Click Publish, and the Sales_Group_CD value is updated back in the MDS database. Also note that the cell (entry) that we are changing will turn orange (within your Excel spreadsheet). When you publish to an existing model, you will also be prompted for some type of annotation that reflects the type of update you are doing (good audit trail policy). Figure 50.25 shows this dialog.

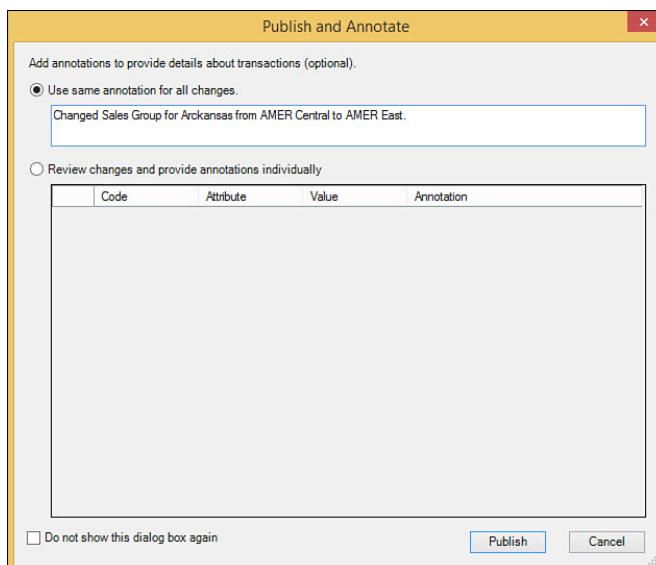


FIGURE 50.25 MDS Publish and Annotate dialog.

You are in MDS business now. Your model contains the entities, attributes, and member values that are fully managed by MDS. Updates, additions, deletions, and publishing all occur from this central MDS repository now.

Master Data Services Transactions

MDS Master Data Manager can also show you all transactions that have occurred in your model. You can also selectively back (revert) one or more transactions as the need arises. Figure 50.26 shows the transaction management option with MDS Master Data Manager. As you can see, all the Sales_Group entity transactions that have just occurred are listed in this transaction manager view, including the before and after values of the Sales_Group_CD change we made for Arkansas.

Entity	Member Code	Explicit Hierarchy	Member Type	Attribute	Added Date	Prior Value	New Value	User
SALES_GROUP	65	Leaf		SALES_GROUP_CD	12/29/2014 5:45:18 PM	AMER_Central	AMER_East	DXD001Paul
SALES_GROUP	116	Leaf		SALES_GROUP_CD	12/29/2014 5:37:30 PM		EMEA_Central_Europe	DXD001Paul
SALES_GROUP	116	Leaf		STATE_PROVINCE_NM	12/29/2014 5:37:30 PM	N/A	N/A	DXD001Paul
SALES_GROUP	116	Leaf		COUNTRY_NM	12/29/2014 5:37:30 PM	Kosovo	DXD001Paul	
SALES_GROUP	116	Leaf		CD	12/29/2014 5:37:30 PM	XX	DXD001Paul	
SALES_GROUP	115	Leaf		SALES_GROUP_CD	12/29/2014 5:37:30 PM		AMER_West	DXD001Paul
SALES_GROUP	115	Leaf		STATE_PROVINCE_NM	12/29/2014 5:37:30 PM	Wyoming	Wyoming	DXD001Paul
SALES_GROUP	115	Leaf		CD	12/29/2014 5:37:30 PM	WY	WY	DXD001Paul
SALES_GROUP	115	Leaf		COUNTRY_NM	12/29/2014 5:37:30 PM	United States	United States	DXD001Paul
SALES_GROUP	115	Leaf		CD	12/29/2014 5:37:30 PM	US	US	DXD001Paul

FIGURE 50.26 MDS transaction management.

If you highlight any one (or more) transaction entries in this view, you can click the Revert Transaction option just above the transaction list on the left. But be careful; use this sparingly. You are basically “undoing” work. Better know what you are doing (usually reserved for administrators).

Master Data Services Hierarchies

MDS Master Data Manager also enables you to create both explicit and derived hierarchies. If you would like to set up some explicit hierarchies, you will start from the Model View option. You can also create domain attributes for each of the attribute groups you may have in a hierarchy and use derived hierarchies to relate these to each other (thus creating a derived hierarchy). Figure 50.27 shows the first detail specification page for explicit hierarchy definitions.

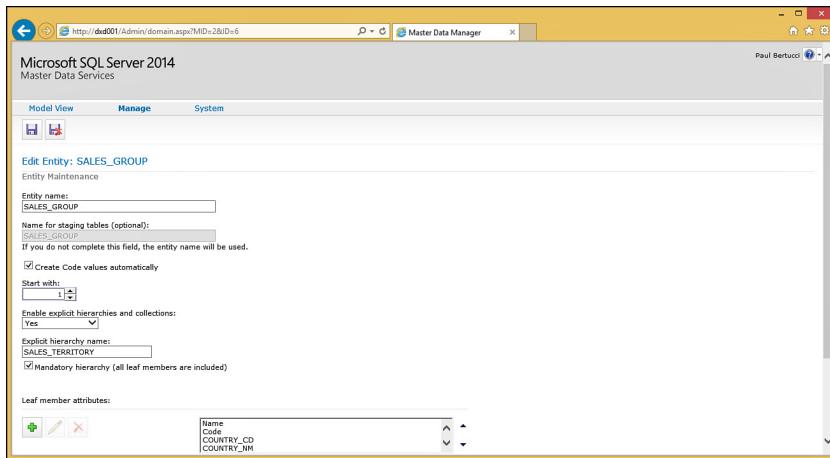


FIGURE 50.27 MDS model view and explicit hierarchies.

It is also very easy to explore each entity and make changes as you need directly from the MDS Explorer (for Entities, Hierarchies, and Collections) as seen in Figure 50.28. You can also create subscription views against entities as you need them that will expose this data to others via normal SQL Server views (but managed from MDS).

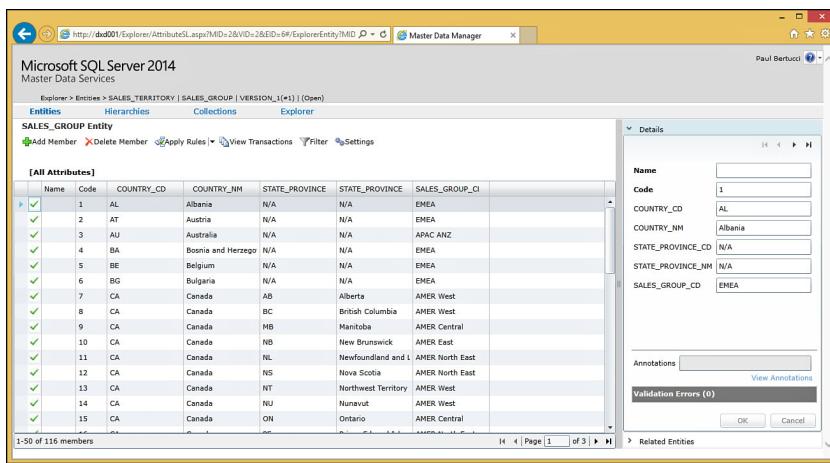


FIGURE 50.28 MDS Explorer of the Sales_Group entity.

Many third-party companies have sprung up recently to create administration and data steward client interfaces on top of the MDS Framework. One of the best we've seen is from Profisee Corporation called Maestro. Below (Figure 50.29) is an example of the Maestro product at work on a complex MDS implementation. The figure shows two screens, one

for the advanced modeling view into MDS, the other showing how to create complex matching strategies for master data management.

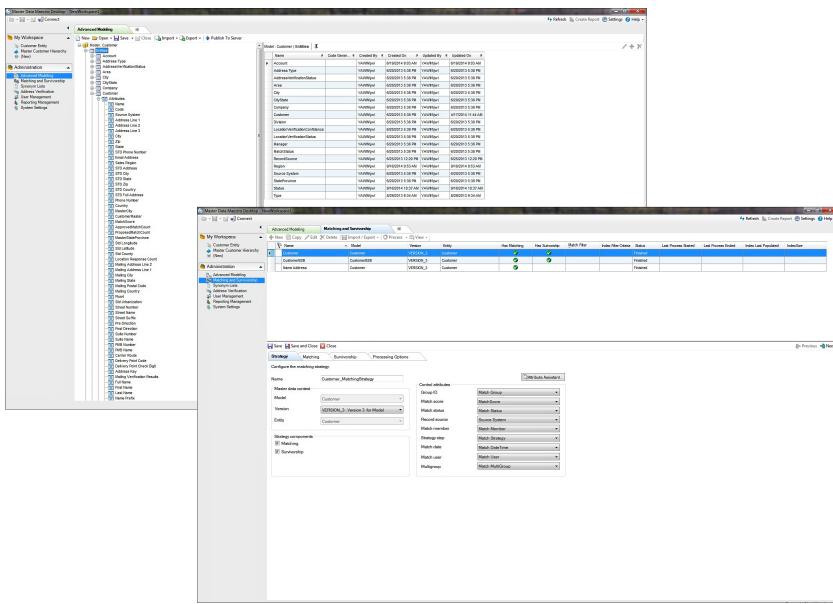


FIGURE 50.29 Maestro Advanced Modeling and Matching Strategies UI for MDS (from Profisee).

What was previously a bunch of distributed Excel spreadsheets scattered across various business groups has now been turned into a single, centrally managed, protected, and publishable master data repository that any company can rely on for control, accuracy, auditability, and policy enforcement. It is also completely accessible from applications, the MDS Master Data Manager, and via the Excel add-in.

A Master Data Services Application (Client Application)

Figure 50.30 shows a large visionary customer hub-and-spoke business application requirement that has a central customer repository (to be implemented with MDS) and several major systems that produce and consume core customer data. Since up-to-date customer data is essential to each major application depicted in this picture, a publish-and-subscribe model will fit well to achieve this (often referred to as *hub-and-spoke architecture*). When any new customer is created (or updated) in any of the producing (managing) applications, all other consuming applications will receive (share) the customer updates quickly.

And as you can also see in the lower left of this master data requirement, a part of the need is to onboard customer data into MDS from merger and acquisitions (M&A), which this company does often. This particular M&A functionality must first check whether the customer already exists in the MDS repository (search before create concept). If it doesn't

exist, it creates the customer, and MDS publishes to all applications (spokes), thus eliminating duplicate customer possibilities from onboarding customer data.

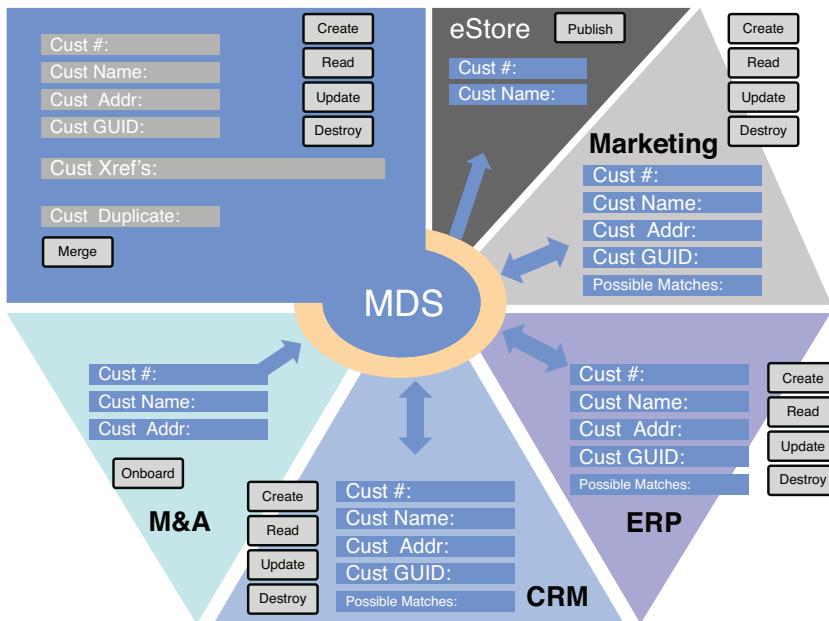


FIGURE 50.30 Hub-and-spoke master data requirement example.

Figure 50.31 shows the corresponding C# web application (`MasterDataDemo`) that implements this complete MDM hub-and-spoke requirement. Each tab along the top is the interface into the different application systems (ERP, CRM, Marketing, and so on) that will produce or consume core customer data attributes. Basically, this client application can see customer entities and their core attributes (members) in each of these back-office applications (the applications that manage customer data) and also see what the MDS repository has for them (centrally). When one of the applications changes an attribute of a customer, it publishes it to MDS (centrally), which in turn pushes it to all other applications (spokes). This hub-and-spoke approach is also very efficient in that it eliminates five (five-factorial) point-to-point interfaces that would normally have been created to pass updates to and from each of the applications (with each other).

As you can also see from Visual Studio (shown in Figure 50.32), this `MasterDataDemo` C# application can utilize any MDS class (visible in the class view or object browser in VS). Nothing much changes with regard to full application development approaches and addressability. This particular application leverages the MVC pattern as well.

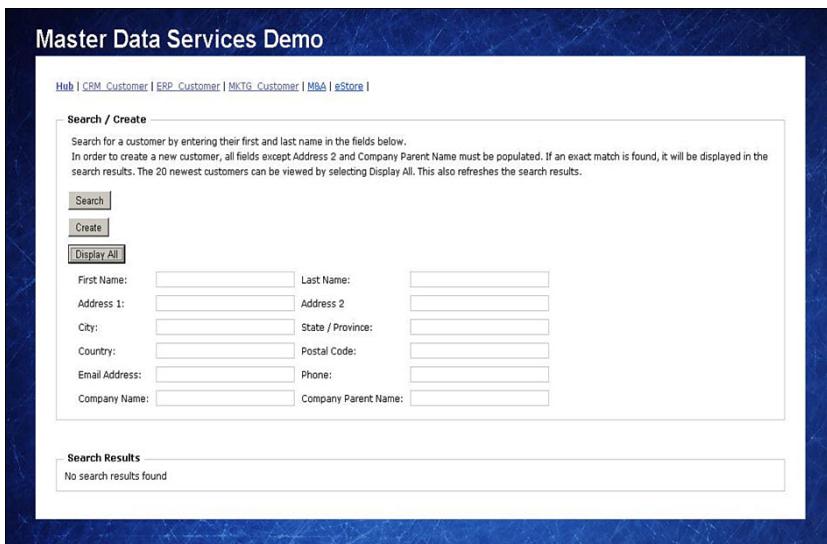


FIGURE 50.31 C#/.NET MDS client application of a full hub-and-spoke requirement.

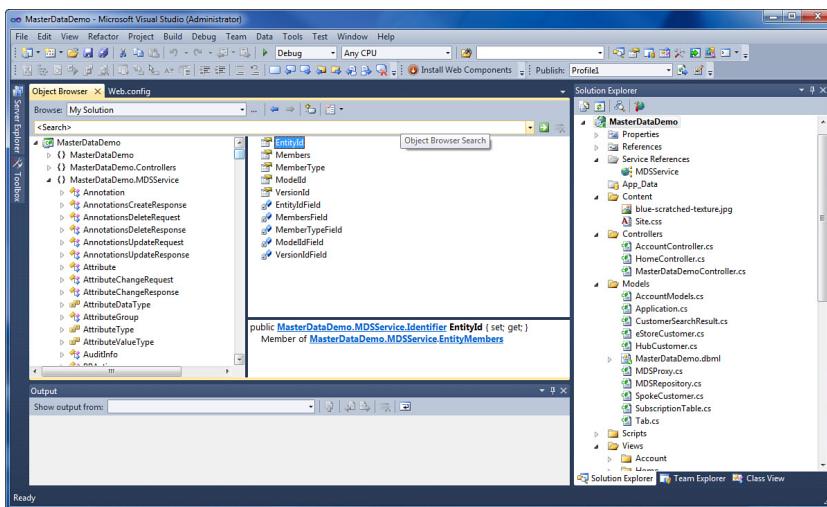


FIGURE 50.32 MasterDataDemo C# MDS application from within Visual Studio.

Master Data Services Deployment Tool

To use a set of mastered data that is managed by MDS, a deploy package approach is used that bundles up the model with the data for use by most applications and integration components. This is done using the command-line deployment tool for MDS called `MDSModelDeploy.exe`.

You run this utility from the server containing your MDS server; it is quite simple to use. You just basically need to create an MDS package and then deploy it. Step one is the create package step. The `MDSModelDeploy.exe` utility is in the configuration folder under the MDS directory structure (`C:\Program Files\Microsoft SQL Server\120\Master Data Services\Configuration\`). From a DOS command prompt, you'd specify the full path to this folder along with the `MDSModelDeploy.exe` utility name, the `createpackage` action option, followed by the model, version, service name, the destination of the MDS deployment package, and finally the `includedata` flag to make sure that the deployment package has the MDS data as a part of the package. Here is the full command prompt specification for the `createpackage` MDS deployment package:

```
C:> "C:\Program Files\Microsoft SQL Server\120\Master Data Services\Configuration\MDSModelDeploy.exe" createpackage -model SALES_TERRITORY -version INITIAL_LOAD -service MDS -package C:\data_files\sales_territory.pkg -includedata
```

Very often, the package file can then be copied to a shared network drive or pushed to a software code management repository such as Perforce (good habit).

Step two is used to deploy this package file to any other MDS environment for use. You run the same `MDSModelDeploy.exe` program, but with different parameters, as shown in the following example (in this case, using the `deploynew` option).

```
C:> "C:\Program Files\Microsoft SQL Server\120\Master Data Services\Configuration\MDSModelDeploy.exe" deploynew -package sales_territory.pkg -model SALES_TERRITORY -service MDS
```

After you have deployed any new package file, you need to complete some post-deployment tasks:

1. In the MDS web interface of the target environment, select the Version Management option from the Main menu.
2. Select the model that you deployed.
3. Click the Validate Version option.
4. Click the Validate button and confirm.
5. Return to the Version Management screen.
6. Select the model that you deployed.
7. Click the Lock button and confirm. This will prevent further changes being made to the loaded version of the data, allowing it to be used for rollbacks.
8. Click the Copy button and confirm.

9. If you think it is necessary for future clarity, rename the new copy by double-clicking the Name and Description fields.
10. Double-click the Flag field of the new copy and select Current to indicate that this new version is the one to be used by any application or SSIS (ETL) program.

Summary

This chapter was jam-packed with master data services concepts and capabilities. We really just scratched the surface on a very complex data need. Understanding the primary concepts and data model with MDS is essential to using this capability properly. You also now know how DQS can be complementary to MDS and to other environments that manage data and can benefit from data quality checking. We have found that it is best to start small with these robust capabilities. Get them working well and then add on as you need to. Most companies choose one core data category (like customer data) to focus on first. The others will come later.

The next chapter, Chapter 51, “Parallel Data Warehouse,” describes how you can move to a highly scalable data warehouse architecture for your enterprise-worthy data needs.

CHAPTER 51

Parallel Data Warehouse

IN THIS CHAPTER

- ▶ What's New in Parallel Data Warehouse
- ▶ Understanding MPP and PDW

Microsoft continues to attack the data warehousing (DW)/business intelligence (BI) and Big Data market by pouring millions upon millions of dollars into this area. Microsoft knows that the world is hungry for analytics and is betting the farm on it (pretty good bet, I'd say). This also means that they are going after both structured data (found in traditional data warehousing platforms and massively parallel processing [MPP] parallel data warehouses) and also including semistructured and unstructured data that can be surfaced from various data platforms (such as with Hadoop via the new PolyBase capabilities) to make all this data available to all of us mere mortals. And it's not only about massively parallel processing (MPP) or Big Data, it is now also about in-memory options such as with xVelocity. We'll even jump into a bit more discussion about columnstore indexes with xVelocity for good measure.

As a part of its original internal project called Madison, Microsoft acquired a few complementary technologies to accelerate its BI plans. One major acquisition was that of the MPP data warehousing appliance company DATAllegro and their MPP parallel data warehouse (PDW) capability. Now, completely re-architected and redeployed, we enter the PDW 2.0 era: SQL Server 2014 Parallel Data Warehouse Version 2.

What's New in Parallel Data Warehouse

The PDW platform has been through a series of big jumps in capabilities over the years. The original DATAllegro engine utilized an Ingres relational database management

system (RDBMS) engine (yep, that's right, Ingres), which had been modified to support MPP. Ingres began as a research project at the University of California, Berkeley, starting in the early 1970s and ending in 1985. The original code, like that from other projects at Berkeley, was available at minimal cost under a version of the BSD license. Ingres spawned a number of commercial database applications, including Sybase, Microsoft SQL Server, NonStop SQL, and a number of others. Postgres (Post Ingres), a project that started in the mid-1980s, later evolved into PostgreSQL. All of the most successful DW appliance vendors use MPP architectures to provide high query performance and platform scalability.

Following are some of the top new features and enhancements of PDW and related options:

- ▶ 50 times faster query performance over the prior version of Microsoft SQL Server PDW
- ▶ 15 times more data compression achieved
- ▶ 70% more storage capacity possible with the new PDW
- ▶ 50% faster data loading speed over earlier versions
- ▶ xVelocity updateable columnstores (even a clustered version)
- ▶ Query relational and non-relational data (from Hadoop) from familiar tools like Microsoft Excel

Other general benefits of PDW include the following:

- ▶ Up to 100 times faster query performance than other legacy DW platforms
- ▶ Double the rate of data loading speed over earlier versions
- ▶ Can support data capacity requirement from 1 to as large as 8 petabytes
- ▶ Lowest price per terabyte in the industry
- ▶ 100% SQL query (no proprietary language)
- ▶ Is now referred to as a component (cornerstone) of the Microsoft Analytics Platform System (APS)
- ▶ Fully integrated via PolyBase to Hadoop (HD Insight) and other Hadoop platforms (Cloudera, Hortonworks, so on).
- ▶ Can support up to 56 compute nodes, of 8 slices per compute node.

Understanding MPP and PDW

Today, data (information) is the new currency of the enterprise. This means organizations that know how to maximize the value of data will thrive. Gartner asserts that "by 2015, organizations integrating high-value, diverse, new information types and sources into a

coherent information management infrastructure will outperform their industry peers financially by more than 20%.” McKinsey agrees with Gartner, confirming that organizations that use data and business analytics to drive decision making are more productive and deliver higher return on equity than those that do not.

Enterprises are seeing the convergence of new information types and sources that are fundamentally transforming the industry. For example, the volume of information a typical organization needs to manage is exploding, and most of the growth is coming from beyond traditional data types in the form of semi-structured and unstructured data. Current software solutions cannot cope with the complexity of this data or the tremendous workload growth it is creating. Transitioning to back-end solutions that support nontraditional data sources (like Big Data) is the only way to effectively manage this data, but organizations are failing to make this transition because of vendor lock-in, costly deployments, and time-consuming migrations.

To help organizations successfully transition to this new thirst of data, Microsoft introduced SQL Server 2012 Parallel Data Warehouse (PDW), the next version of the scale-out MPP DW appliance. That PDW solution provided a generic data platform for customers to unlock the value of their data. Of course, PDW has now been in use for a number of years and with each new release continues to grow its capability, platform integration options, and total amount of data volume it can handle for the ever increasing data explosion that is happening in our world. Microsoft’s current product offering is still their core data warehouse appliance, but also offers the Polybase and HDInsight (Hadoop) “regions” (components) to cover all types of data (relational and non-relational). Microsoft refers to this new hybrid offering as the “Analytics Platform System,” or APS for short.

MPP Architecture

PDW is based on an MPP architecture. MPP architectures consist of independent processors or servers executing in parallel. Most MPP architectures implement a *shared-nothing architecture*, where each server operates self-sufficiently and controls its own memory and disk. Shared-nothing architectures have a proven record for high scalability and little contention. DW appliances (like PDW) distribute data onto dedicated disk storage units connected to each server in the appliance. This distribution allows DW appliances to resolve a relational query by scanning data on each server in parallel. The divide-and-conquer approach delivers high performance and scales linearly as new servers are added into the architecture.

As Figure 51.1 shows, there are two primary functions occurring in an MPP architecture: The Controller function, which controls where data goes and how it is retrieved and assembles MPP result rows; and the Parallel Data function, which consists of many parallel compute and storage node pairs. Each compute node and storage node pair control and store a portion of data for the overall database and then act in parallel to deliver their data to the control node for consumption by the data requestor.

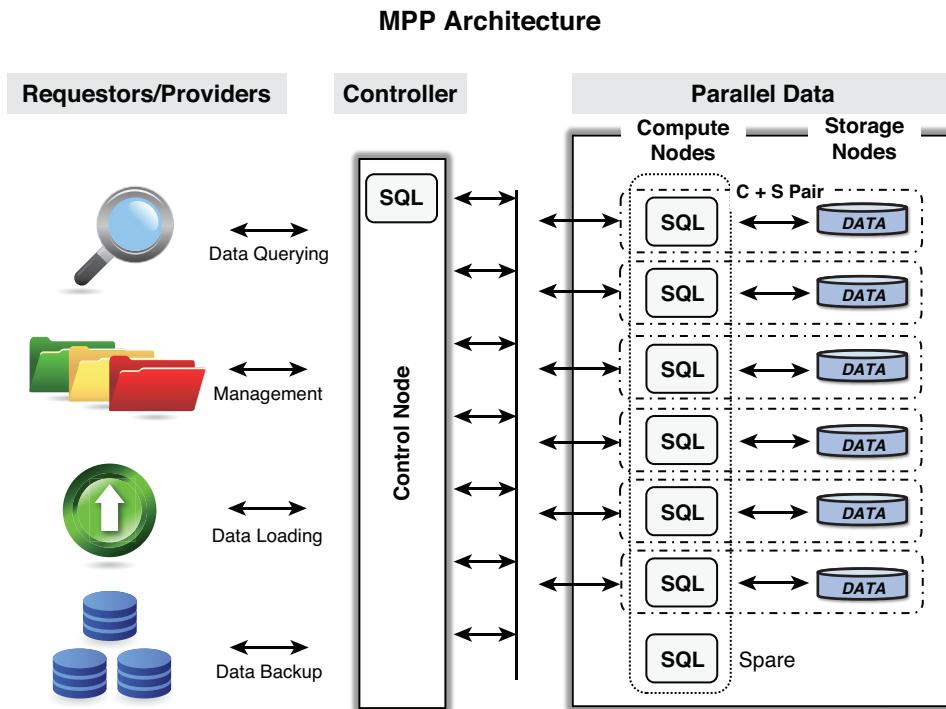


FIGURE 51.1 Massively parallel processing (MPP) architecture.

As you look at Figure 51.1, imagine that data is flowing from the far right to the left in parallel, to be assembled at the control node and then delivered rapidly to the data requestor (data query). The goal of MPP data architecture is to store data shallow and wide. Think of 100 people standing in water that is 1 inch deep and rocks (data) sitting just below the surface of the water ready to be picked up all at once by each of the 100 people (all at one time). That can happen in less than a second. That's a lot of rocks all at once. Now think of those same rocks covered by 100 inches of water and only one person given the duty to dive down to the bottom and carry as many of these as he can to the surface (but he cannot carry more than a few of these rocks at a time). That is a traditional database approach. For data that needs fast retrieval times, the parallel approach is optimal. Also notice in Figure 51.1 that the controller node and each of the compute and storage nodes is a SQL engine, each doing its separate and optimized job of managing and assembling (control node), storing and retrieving data (compute and storage nodes) in a massively parallel way.

MPP database architectures have a long pedigree: Teradata, Tandem, Britton Lee, and Sequent computers pioneered this approach decades ago (1980s) but were (are) extremely expensive. Open source RDBMS products, such as Ingres and PostgreSQL, reduce software license costs and allow DW appliance vendors to focus on optimization instead of just providing basic database functionality. In addition, advances in technology have reduced

costs and improved performance in storage devices, multicore CPUs, and networking components. SQL Server 2014 PDW can scale out to handle requirements of almost any size.

The PDW

SQL Server 2014 PDW is delivered as a prebuilt appliance with software, hardware, and networking components already preinstalled. And, as mentioned before, the PDW appliance has added other features such as xVelocity-based columnstore indexes and PolyBase query integration for retrieving structured and unstructured data together. As you can see in Figure 51.2, the PDW is truly an optimized MPP appliance that has both high availability and high performance all in one. But remember, it has been optimized for data retrieval, not for data updates, deletes, or inserts.

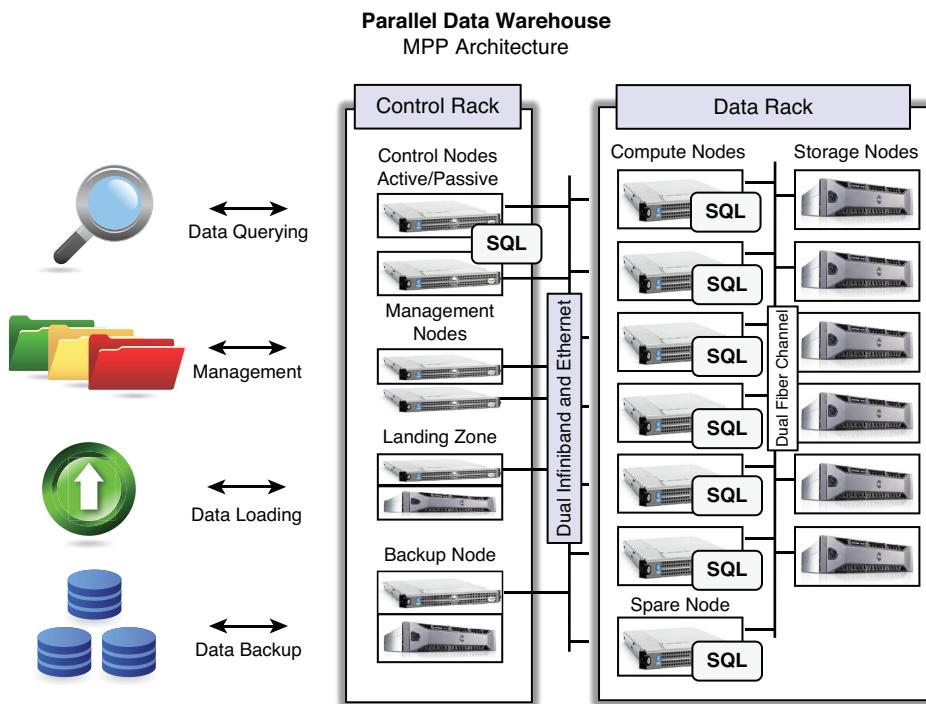


FIGURE 51.2 PDW appliance MPP architecture.

The controller functionality is embedded in the control rack part of the appliance. It consists of the Control nodes (a SQL engine in an active/passive high-availability configuration); Management nodes for common metadata, access control, patch management, and other management tasks; the Landing zone nodes for data loading (also in parallel) and data staging into (and out of) the PDW; and the Backup nodes used to provide optimized backups (and recovery) of the persisted data in the PDW. The controller node is the

traffic cop for the whole architecture. It knows how to distribute the data (distribute it out to the proper compute and storage node), assembles all data results from each parallel compute/storage node, and provides the results back to the end user.

The parallel data functionality consists of compute and storage node pairs that take on ownership of a portion of the data, in parallel. PDW calls this the data rack. There is also a spare compute node that can take over any active compute node at a moment's notice. Again looking at Figure 51.2, you can see the compute nodes and storage nodes are all connected via dual Fibre Channel connections for extreme data access and data movement speeds. As data is surfaced up from each parallel compute node in the data rack, it is passed on to the control rack via dual high-speed InfiniBand/Ethernet network connections. Basically, as fast as the data is surfaced in parallel from the data rack, it is being assembled for delivery to the requestor (from your query). When you need more capacity, you simply add more nodes (called *scaling out*). This can enable high-performance data access for business purposes up to petabytes of data and handle almost any complex and concurrent queries without requiring the flattening of existing data models or compromising data relationships.

As of this writing, the PDW appliance (hardware and software) is available from three primary vendors: HP, Dell, and now Quanta. You have the option to choose and may already have prior relationships with one of these vendors. We will not describe any of these hardware products or options in this book. Seek out details from your preferred vendor.

You can also add capacity with ease: With traditional data warehouses that scale up, organizations need to rebuild entire servers once the maximum physical capacity and performance of servers hosting the data warehouse is exceeded. With this MPP scale-out architecture, SQL Server 2014 PDW will scale out to nearly any or all future data requirements you may have, starting from the smallest requirements (0–15 terabytes) up to the very largest (8 petabytes). You start with only what you need; the base system starts with one rack. A full rack holds 10 nodes, and additional 10-node racks can be added up to a total of 40 nodes. Wow!

Here are some PDW statistics:

- ▶ **Compute node**—12 cores and 96GB RAM per node
- ▶ **Data**—960GB RAM per data rack (as a whole)
- ▶ **Data**—Up to 153TB of storage per data rack (as a whole)
- ▶ **Data scan rate**—Up to 2.6TB per minute data scan rate for a data rack

Data on a PDW

Taking a closer look at how data is stored on the PDW, you'll better understand how this architecture yields the data access speeds it delivers. Looking at Figure 51.3, we will define a typical star schema sales database that contains sales facts (measures) such as Sales Units (A), Sales Amounts (B), and Sales Returns (C). In addition, this star schema has three

dimensions: a Product hierarchical dimension, a Geography hierarchical dimension, and a Time dimension. Each fact in the fact tables is identified by all three of these dimensions.

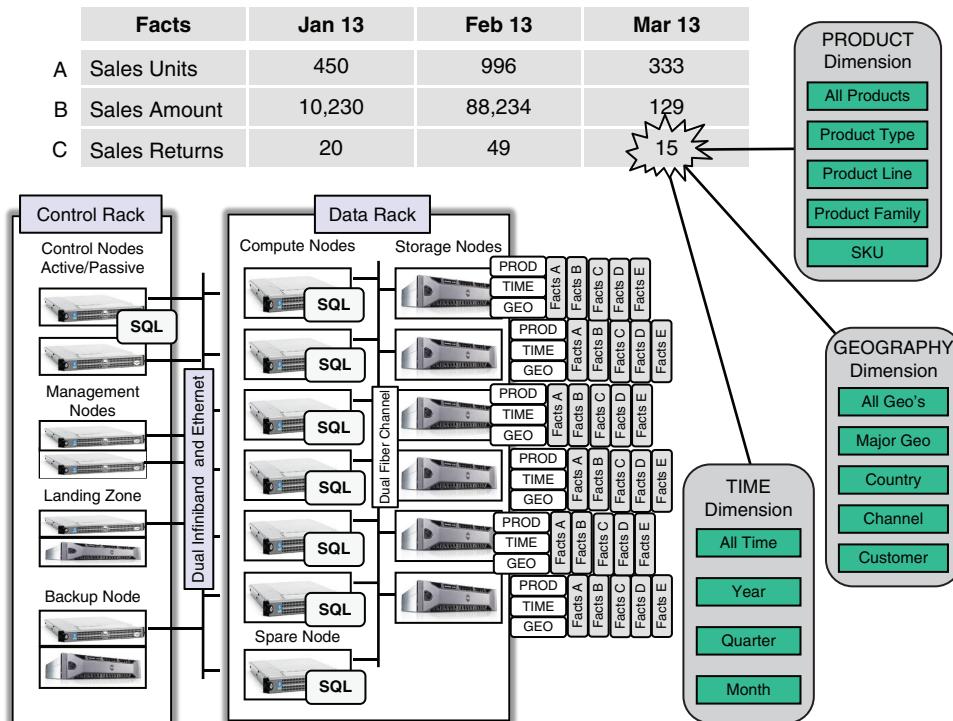


FIGURE 51.3 Sales data facts (measures) and dimensions on the PDW.

Now, as you can also see in Figure 51.3, the PDW storage of this data is as follows.

Each dimension table (Product, Time, and Geography) is typically replicated to each compute/storage node in the data rack. This is so that each node can fully resolve and navigate these dimensions to retrieve (join) the corresponding facts stored on their particular node.

Dimensions are almost always replicated. Dimension tables that are 5GB uncompressed or smaller should be replicated. A 5GB dimension would compress to around 2GB, which would take up a total of 20GB once it is replicated across a 10-node rack. Compression is automatic in SQL Server PDW. Replicating other tables is also common for fairly small tables that you might want on each node (such as reference type data) and that would also be referenced in a join. These are all called *replicated tables*. Then, each node will also store a range (hashed partition) of each of the fact table's data (Facts A, B, C... and E in this example). These are called *distributed tables*.

As you can see in Figure 51.3, fact table A's data is spread (distributed) across each compute/storage node evenly, as is fact table B's data, and so on. This makes each fact table's data very shallow and super-fast to retrieve and be assembled in parallel by the control rack. Each compute/storage node is retrieving only the qualifying rows that they contain (and store). This MPP architectural approach basically supports most standard or ad hoc BI queries that can be conceived of for the sales data across each (and all) of these dimensions.

DDL Examples on PDW

Here is an example showing how to create a SQL Server 2014 PDW database:

```
CREATE DATABASE Sales_PDW
WITH
    (AUTOGROW = ON,
     REPLICATED_SIZE = 1024 GB, -- (space per Node)
     DISTRIBUTED_SIZE = 16384 GB, -- (space for entire DB)
     LOG_SIZE = 1024 GB);
```

A replicated table that contains ZIP code to ZIP code geographic region cross references could be done fairly simply with the following `CREATE TABLE` syntax:

```
CREATE TABLE Zipcode (
    ZipCode          varchar(10),
    ZipcodeGeoRegion varchar(50))
WITH
    (DISTRIBUTION = REPLICATE);
```

The default is `REPLICATE` if the distribution clause is omitted.

Creating a replication table for each dimension is similar and would look like this:

```
CREATE TABLE ProductDIM(
    ProductType      INT           NOT NULL,
    ProdTypeDescription VARCHAR(50) NOT NULL,
    ProductCategory  INT           NOT NULL,
    ...
    ProductKey       INT           NOT NULL
)
WITH (DISTRIBUTION = REPLICATE);
```

Creating the fact table that is to be distributed across all the nodes entails a bit more effort and is done using the Hash distribution option. It is also important to understand which column makes the best candidate to base the distribution on (in the `HASH` statement). Here is an example of our Sales fact table that will distribute the product sales facts across each node by date ranges (6 months of data on each node):

```

CREATE TABLE Sales_Facts(
    TimeKey          INT      NOT NULL,
    GeographyKey    INT      NOT NULL,
    ProductKey       INT      NOT NULL,
    Sales_Units      INT,
    Sales_Amount     MONEY,
    Sales_Returns    INT,
)
WITH (DISTRIBUTION = HASH(ProductKey),
    CLUSTERED INDEX(TIMEKEY),
    PARTITION (TIMEKEY RANGE RIGHT FOR VALUES
        ('20100101',
        '20100701',
        '20110101',
        ...
    ))
);

```

Determining which is the best distribution column to use requires a bit of knowledge of both the data and of the type of general querying that will be done. Low cardinality column values create lousy distribution columns; data ends up on very few nodes. It is best to choose something that has high cardinality and distributed values (and that makes data query and business sense). In our example, the data queries are mostly quarter over quarter comparisons of sales data (sales trends).

PDW and Big Data (Hadoop)

Excuse me? Did you say that it is possible to access unstructured Big Data with structured PDW data? Yep, that's right: from a single SQL query that provides complete insulation of the underlying data sources that contain the data (structured or unstructured).

The secret sauce comes in the form of something called PolyBase, a breakthrough component that enables querying across Hadoop and relational data in addition to providing integration with the whole Microsoft BI stack.

In a nutshell, PolyBase can handle a standard Transact-SQL query that joins tables from any relational source with tables from a Hadoop cluster and seamlessly return the results to the user. In other words, you don't have to load data you already have in Hadoop storage to your PDW data warehouse. You also don't have to deal with or learn the MapReduce proprietary approach to getting to data in Hadoop either. It simply means that you can now include the data you have in Hadoop (unstructured and semi-structured data) with the data you have in your PDW and other SQL databases from a single SQL query.

From a high level architecture with the PDW appliance point of view, Figure 51.4 shows the primary components of the PDW that is outfitted with PolyBase and Microsoft HDInsight for combining structured data with unstructured data (Big Data). Figure 51.4 also shows how other cloud or appliance data islands can fit easily with the PDW/PolyBase

solution, in this case Microsoft Azure (cloud) HDInsight Hadoop and/or Hortonworks/Cloudera appliance/commodity Hadoop data platforms.

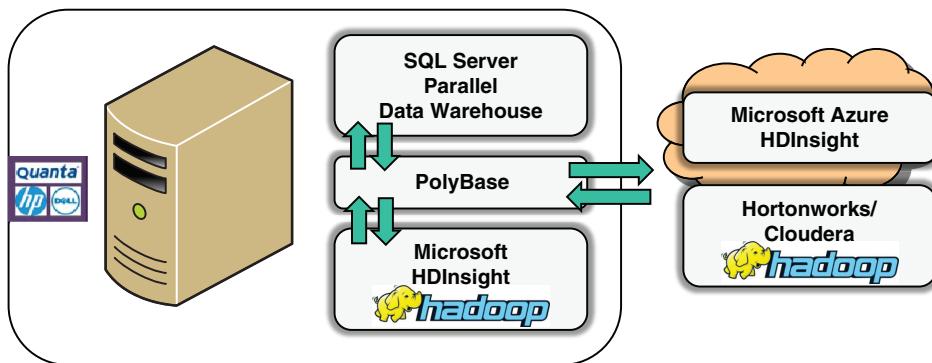


FIGURE 51.4 PDW, PolyBase with Big Data architecture.

Figure 51.5 gives you a more detailed view of how PolyBase and Hadoop combine with PDW to unleash complex data analytics that were never possible before (basically providing you super analytic mashups).

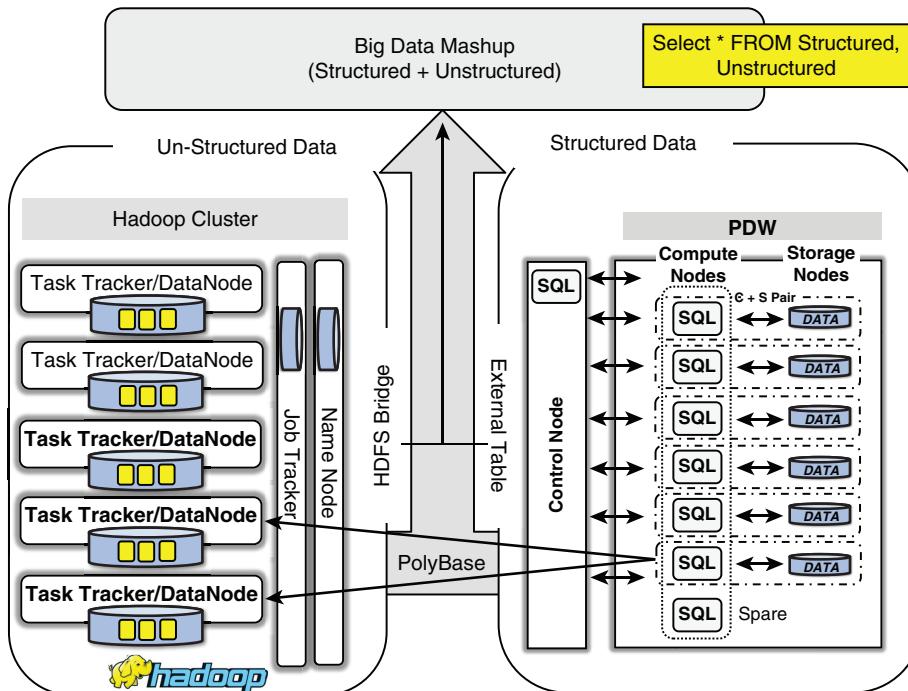


FIGURE 51.5 Structured PDW data with unstructured Hadoop Big Data via PolyBase.

PolyBase is part of an overall Microsoft Big Data solution that already includes HDInsights from Hortonworks (a 100% Apache Hadoop compatible distribution for Windows Server). As you can also see in Figure 51.5, the Hadoop Distributed File System (HDFS) Bridge abstraction layer provides the HDFS structured access, and this HDFS Bridge has been added to the Data Movement Service (DMS) that runs on each PDW node. Essentially, the PDW data is matched to the Hadoop data at the node level, and all data accesses occur at the compute node level (still in parallel). The External Table option is used on the PDW side to identify what Hadoop nodes to point to.

xVelocity Columnstore Indexes

SQL Server 2014 PDW uses the updateable version of xVelocity columnstore capabilities as a part of PDW for added linear scale out.

This is a memory-optimized columnstore index, which is both updateable and clustered. In SQL Server 2012 Database Engine, columnstore indexes didn't allow table updates to tables with columnstore indexes. In SQL Server 2014, these are now updateable.

By reorienting data in a column format rather than a traditional row store, you can get up to 50 times the performance gains. This is due to the grouping of data into a columnar format, which contains values from multiple rows. The net effect is much higher efficiency in both storage and the dataset processing of column-oriented SQL queries.

Columnstore Indexes

As mentioned before, for some types of queries, SQL Server can take advantage of a columnstore (columnar) layout to significantly improve query execution times (up to 50x performance in many cases). In particular, if you have a lot of filtering, aggregations, grouping, and star-join queries that touch few columns, but many rows, you are likely to want to look seriously at columnstore indexes.

The primary columnstore index concepts and characteristics you need to be aware of are as follows:

- ▶ **Columnar data format**—Data is grouped and stored one column at a time, not in traditional row format.
- ▶ **Only the columns needed are read**—Therefore, less data is read from disk to memory and later moved from memory to processor cache.
- ▶ **Columns are heavily compressed**—This reduces the number of bytes that must be read and moved.
- ▶ **Most queries do not touch all columns of the table**—Therefore, many columns will never be brought into memory. This, combined with excellent compression, improves buffer pool usage, which reduces total I/O.
- ▶ **Advanced query execution processing approach**—Touches chunks of columns called batches in a streamlined manner, reducing CPU usage.

- ▶ **Key columns**—There is no concept of key columns in a columnstore index, so the limitation on the number of key columns in an index (16) does not apply to columnstore indexes.
- ▶ **Clustered index key**—If a base table is a clustered index, all columns in the clustering key must be present in the nonclustered columnstore index. If a column in the clustering key is not listed in the `create index` statement, it will be added to the columnstore index automatically.

Columnstore indexes

- ▶ Cannot have more than 1,024 columns.
- ▶ Can now be clustered.
- ▶ Cannot be a unique index.
- ▶ Cannot be created on a view or indexed view.

Columnstore indexes cannot be combined with the following features:

- ▶ Page and row compression and the vardecimal storage format
- ▶ Replication
- ▶ Change tracking
- ▶ Change data capture
- ▶ FILESTREAM

When you create a columnstore index, you are providing a vertical slice of column-oriented data to SQL Server (called *columnar*) so that SQL Server can use it as a more efficient data access plan. Figure 51.6 shows the basic columnstore index concept. This narrow column-wise representation (store) will be created with only the targeted data from the columns. In this example, this is the `Salary` column that is needed to satisfy the query that can be used effectively to determine the average salary for employees.

When you extend this capability to tables that have millions or billions of rows, the magnitude of performance gains is substantial.

A great feature with columnstores is the extensive compression that is done on the column's data. The data is compressed, stored, and managed as a collection of partial columns, called *column segments*. Figure 51.7 shows the data life cycle of how data rows are transformed into column segments and then compressed and added to the columnstore.

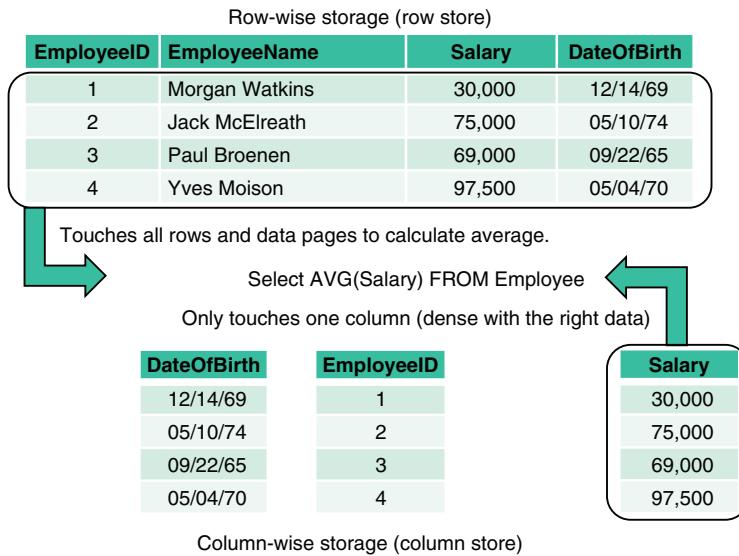


FIGURE 51.6 Row-wise (row store) versus column-wise (column store/columnar).

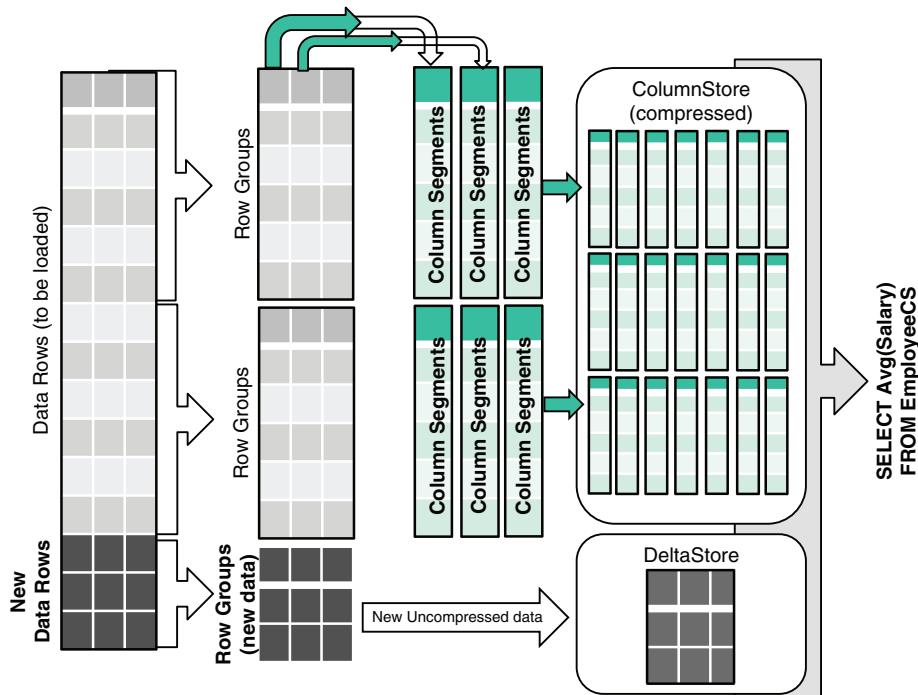


FIGURE 51.7 Data loading from rows to columnstores.

Some of the columnstore index data is stored temporarily in a rowstore table, called a *deltastore*, until it is compressed and moved into the columnstore. The columnstore index operates on both the columnstore and the deltastore, to return the correct query results.

Summary

This chapter explored the firm path Microsoft is pursuing with MPP DW appliances and some additional features such as xVelocity (in-memory option) and columnstores. There are many more emerging tools coming for the DW components in SQL Server, but the current offering is just short of incredible already. It is also now easier than ever to bring together both structured data (in PDW) and unstructured data (in Hadoop) to start conquering the Big Data opportunity that you might have.