1

Getting Started with Reporting Services

SQL Server Reporting Services is an amazing offering from Microsoft that will change the way you create and deploy reporting solutions. It's difficult to fully appreciate the revolutionary nature of this product until you understand its architecture. The look and feel of the Report Designer environment and the functionality of a particular report view window have little to do with its full capabilities. Like your favorite media player program, you can always bolt-on another skin or façade, but it's what's inside that really matters. The product group has done a stellar job by providing a design environment and a nice web-based report management and viewing application. The impressive part is the underlying architecture that makes SQL Reporting Services a fully scalable and extensible solution that is also surprisingly easy to work with.

If you are impressed by the capabilities of the .NET Framework, web services, SQL Server, and ASP.NET, you should know that by using these technologies Reporting Services takes data accessibility to the next level. Microsoft is making good on their promise of making information available "*any time, any place, and on any device*." Reports may be designed using specific rendering formats and page sizes to support mobile devices. There are many other reporting tools with impressive capabilities but none of them are quite like this one.

This chapter will introduce several topics that will be covered in greater detail later in the book. This will be a high-level view of the need for, purpose, capabilities, and mechanics of SQL Server Reporting Services.

In short, this chapter includes a discussion on the following main topics:

- The history of reporting
- *Business Intelligence (BI)* and decision support in current and past reporting solutions
- Reporting solutions and application types used to deliver reports

- Installing Reporting Services, setup options, resources, and tools
- *Report Definition Language (RDL)*

Who Is This Book for?

Since you've picked up this book, you may be in need of a reporting solution. You may be an application developer, solution architect, project manager, database administrator, or business owner. Maybe you're not a technical professional and you just need reports for your business. Perhaps you are the executive sponsor of a project and you need to know what kinds of capabilities are available for IT professionals to build a solution for you. We assume that you have made (or are considering making) an investment in Microsoft products to manage a business process of some kind. You may need SQL Server or Visual Studio .NET.

We have made it a point to address several aspects of reporting from the perspectives of executives and business managers who need to have solutions developed for them; project managers, business analysts, and software developers who will design and create solutions; and for database and system administrators who will configure, deploy, and maintain databases and business reporting infrastructure.

After spending a couple of months with the early beta release versions of Reporting Services and building solutions with them, I had the opportunity to conduct some early adopter classes for BI and report solution professionals before the product was released. For most, this was their first look at SQL Reporting Services. Everyone was impressed and excited about putting it into practice. I was taken back by a handful of non-developers who complained that they wanted to use Visual Studio to create reports. *"Why should we have to buy this product and learn to use it?"* they asked.

In Chapter 11 you will see how designing reports isn't restricted to the Visual Studio .NET design environment. There will likely be other design tools for building reports in the market soon. The fact is that designing reports is easy. If you have used other report design tools, I'm sure you will agree. One nice thing about using the Visual Studio report designer is that it feels like the other Microsoft products you already know how to use. If you are a Microsoft developer, you'll love it. If you're not a developer, you'll love it when you realize how easy it is to design, deploy, and manage very powerful reporting solutions with it.

Agility

Imagine that you are sitting in a presentation meeting at the corporate office of a key customer. You are a senior sales representative for a company that sells high volume data backup systems, and the solution they decide on will be implemented in several regional data centers around the world. Your team has been preparing for this meeting for months. Your success depends on your ability to demonstrate your competence to the customer and a clear understanding of their needs. Your team has done their homework, and you know the customer has a history of scanning printed medical records and storing them as image files. Based on this information, you are certain that a particular product will adequately provide the file backup facilities for their moderate volume of image files. You have made it a point to familiarize yourself with the capabilities of the system that appears to be the best fit.

During your customer's opening presentation, they tell you that they have recently made a huge investment into full motion video imaging equipment. Now they need a backup system that can handle large file capacities. They are prepared to make an investment that is substantially larger than what you had anticipated for a capable backup solution. Your company began to offer a large-scale solution just a couple of weeks ago but you aren't very familiar with its capabilities. You've spent so much time preparing to sell the smaller system that you haven't had time to learn more about this new product. Your associate is doing introductions, and it will be your turn in about 15 minutes.

Discretely, you open your Pocket PC Phone and access the World Wide Web. You login to your company's secure report server, select the product catalog report; choose the product category and then *drill-down* to the new product. The report has a *drill-through* option that lets you quickly view a detailed specification report for the new, high-volume backup system. After noting the pertinent specifications, you save this report to a PDF file and then choose the customer sales inquiry history report. Looking up this customer, you learn that someone named Julie made an inquiry about two months ago regarding video media backups from this very company.

Looking around the room, you find a name card with her name on it. You explore the details of this call, and you find that she had asked if you offer a solution comparable to a very expensive product from a competitor. Checking the competition's web site, you discover that the competing product Julie had mentioned uses older technology, has a smaller capacity than the new system, and it costs considerably more. You save a report with all of the pertinent specifications to your memory card, hand the card to the administrative assistant sitting next to you, and ask that he make printed copies of the PDF file it contains.

Your colleague finishes her presentation and then introduces you. Taking another quick glance at the new product specs, you begin your introduction. You explain that one of your team's greatest strengths is your real experience and understanding of how business can change day-to-day. In order to be responsive and competitive, it's necessary to adapt to these changes. You show the brochure for the mid-scale product and explain that this product would be an excellent solution for a company that just scans documents. But for digital video, a more capable solution is required. You share the product specification and qualify the product to your customer's needs. During your presentation, the administrative assistant returns with the printed specification report. Not missing a beat, you distribute these to everyone and conclude. Making brief eye contact with your colleague, he raises an eyebrow just before your customer's chief decision maker, Julie, aggressively shakes your hand, and thanks you profusely for your time and effort.

The Way We Were

In many business applications, reports were an afterthought. This lack of planning often forced developers to build ad-hoc reports with little opportunity for significant planning and design. Queries became complicated and difficult to support. Reports ran slowly and were prone to errors. To avoid these difficulties, you really need a plan. In a perfect world, you would architect the database and application around your reporting needs, and would completely understand your users' requirements before designing the system. In the real world, you may understand some of the users' needs ahead of time but chances are that new reports will be requested long after the other features are in place.

According to Frederick P. Brooks' *The Mythical Man-Month*, it's usually a good idea to learn from and throw away your first few attempts at almost any design. I typically try to develop reports in stages

realizing that the first attempt will be a prototype. My experience has been that when you gather the initial requirements, users will ask for a handful of different reports based on some specific criteria. After the solution is deployed and people begin to use it, others will almost inevitably realize that they too would like reports to help make their jobs as easy as their associates. As users realize what kinds of information they can get, they will find new and exciting ways to sort, filter, group, pivot, and slice and dice their data – in ways they never thought possible. That is, until you show them the possibilities.

That Was Then, This Is Now

Static, printed reports may be an acceptable format for a list of products and prices or for a company, but not for the majority of the information people use to make important decisions today. Business decision makers need pertinent information, and they need to view it in a manner that applies to that person's role or responsibility. Since most users deal with information in a slightly different manner, you can create hundreds of reports, each designed for a specific need. Alternatively, you can create flexible reports that serve a broader range of user needs. For example, a sales summary report could be grouped or filtered by the sales person's region, by customer type, and include information for the week, month, quarter or year, or for a specific product category. To produce individual reports for each of these needs would be time-consuming and cost prohibitive. Besides, computer users are savvier than they were a few years ago and need to have tools that help them take informed decisions, not just look at the numbers.

I recall working at Hewlett-Packard several years ago in a manufacturing site IS group. Every Thursday a report cart would come around. There were several regularly scheduled reports that the mainframe system produced on a weekly and monthly basis. Users, typically department managers, would subscribe to these reports that were then printed in another building and delivered by hand to each subscriber. Many of these reports were little more than a huge list of numbers and text printed on continuous, fan-fed paper – some as large as 500 pages. I watched inquisitively as managers would meticulously scan through the pages, highlighting and circling figures of interest. Some would bind them into large books and give them to their administrative assistants to go through with a ten-key calculator and add up all of the figures they had highlighted.

At the end of the month dumpsters full of these reports were hauled off to landfills and recycling centers as their usefulness quickly came to an end. I spent nearly two years developing a reporting application for this group using Microsoft Access. We originally planned for eight to ten reports in this application. But as time went on, and users began to rely on the reports to perform their jobs, they would ask for the same reports with different sorting, grouping, and selection criteria. In the end, we deployed some 25-30 reports, most of which were variations on the few original reports.

Business Intelligence and Decision Support

Dennis Higgins is a BI Consultant with Strafford Technology, Inc. in Windham, NH. He says, *"I think one of the greatest challenges to providing BI Solutions is to educate the customer as to the extent of the long-range problems (and the associated business costs) caused by disjointed attempts to derive information from corporate data. Closely related to that is to correct the normal tendency to apply band aids. Foresight and planning with a BI Strategy is the most effective means of halting the creation of stove-pipe data analysis systems. Once management perceives the benefits and buys into the process, a 'master plan' strategy can be formulated, that will guide the*

process of developing the solution. Integration of existing systems, new tools, or BI Platform migration can then be tackled based on priority and available resources."

Business executives understand that it's important to have good data. They reason that good data should lead to good decisions, and good decisions mean good business. This makes sense, right? A very common scenario today is that businesses trying to get that edge will invest in expensive ERP systems that effectively gather and store mountains of customer, product, and sales information. Mission accomplished? Wrong! These days, the time between data entry and consumption is very short, almost instant. More effective data-gathering mechanisms result in data silos and data warehouses populated to the gills with all kinds of facts.

The new generation of business workers are informed and empowered to make decisions. They need tools to get useful information and respond to changes. Having data available is useless unless it has business value and can be used to effectively take informed decisions.

A fundamental fact in business is that the people who gather and collect data are often not the people who use that data or need access to the information that the data represents. Business executives, managers, and analysts make strategic decisions everyday that may affect many people, the direction of their organizations, and ultimately, the way people and organizations will go about conducting business in the industry. These decisions are largely driven by the relative height of a bar displayed in a chart or a few numbers printed on a piece of paper. Having capable reporting tools doesn't necessarily solve this problem. Most businesses don't know how to effectively use the products they own. A reporting tool is of little value if it's complicated and difficult to use.

This presents some fundamental challenges such as collecting comprehensive, accurate and meaningful information, storing it in a form so it continues to represent the facts, and presenting the information in a concise and unbiased form. On the surface, it seems like a simple task.

Automation to the Rescue – A Scenario

I'll share an example of this kind of challenge. Several years ago, I spent a few months developing a reporting system for the operations group at a paper mill in the Pacific Northwest. The old mill is located in a small, remote town and many of the people operating the mill have been working there all of their lives. As is common in the pulp and paper industry, the mill has changed ownership a few times and is currently operated by a very large paper and office supply company.

As time went by and technology changed, several different computer systems were incorporated into the operation of this mill; an IBM 360 and an AS400 system were used to manage customer orders and production history records. The original inventory management system is still in place. It's a very old, special-purpose computer that stores most of its data in a single, flat text file. All of its components are redundant and it hardly ever needs significant maintenance. Shortly before I arrived, a Windows server box was installed with a SQL Server database and an application that would replicate production and inventory data from the existing database systems. Management within the parent company believed that they didn't have a handle on the rates of material consumption and product quality. They wanted a reporting system that would give them the figures they needed to make adjustments to their ordering and pulp production processes.

Over a period of time, orders would be placed for certain grades of pulp. The system would calculate quantities of ingredients to produce a batch – typically to fulfill an order for a customer. The order would be sent to the production floor where workers had newly installed controls used to assure the accurate delivery of pulp ingredients. Different batches of product continued to be produced with varying degrees of quality and their ability to track the consumption of these materials didn't significantly improve. Management continued to invest in reporting solutions. They bought and developed software to look for trends and perform statistical analysis but to no avail.

After several months and hundreds of thousands of dollars invested, the product quality didn't really improve much. Finally, one of the IT managers put on a hard hat and walked down to the production floor to observe the process. What he learned was a simple lesson: when the orders arrived on their computer workstations, workers were printing the orders and then putting them aside. They had overridden the automated controls and were using the same manual techniques to make paper that earlier generations had been using for decades. It was a matter of tradition and pride, and they weren't about to let some computer do their job for them.

The initial reporting solution was elegant and technically capable. The calculations were accurate and the report presentation was appropriate. However, the solution didn't fully support the process. This cultural hurdle was eventually overcome (workers were instructed to use the automated systems if they wanted to keep their jobs) and the product and process improved. A report is only as good as the data it presents, and the data is only as good as the information used for collection. The information is only as good as the process that it represents.

Challenges of Existing Reporting Solutions

For over ten years, Microsoft has offered only one product with substantial reporting capabilities. Designed to run as a single-user or a small workgroup, desktop application, Microsoft Access is a capable database and reporting solution. In Access 2000, Access Data Projects were added. This extension of the product works well against a SQL Server back-end, in a LAN environment. In Visual Studio 6, an integrated reporting tool was offered for Visual Basic 6 but its capabilities were meager at best. Developers at that time thought this was a glimpse of things to come in subsequent versions of Visual Studio.

Due to the lack of a unified, consistent approach for reporting, many developers have had to revert to creating their own custom solutions. One case in point is the reports starter kit project available on the ASP.NET development support site (www.asp.com). The developers did a bang-up job creating a web-based reporting solution using ASP.NET datagrids and datalist controls. They even made their own pie charts using line drawing objects. This effectively proves that .NET is a powerful arsenal of programming tools. However, it also makes the point that we have lacked a strong reporting solution to round out Microsoft's front-line development and database suite.

When Visual Studio .NET was released in 2002, I was a little disappointed because the only integrated reporting component was a limited-use version of *Crystal Reports*. Now, before I get myself into too much trouble with folks who may be loyal to this product, I'll say that Crystal Reports is a capable reporting tool. However, it's neither a part of Microsoft's strategic direction nor does it behave like, or integrate tightly with other Microsoft products. The version of Crystal Reports that installs with Visual

Studio is limited to five concurrent users (and the term *concurrent* is subject to some serious interpretation). Now that Crystal Reports has changed hands once again (recently acquired by Business Objects), it will be interesting to see how this affects the direction of this well-known product.

Notably, the most remarkable change in the industry over the past few years has been the opportunity and need to exchange information over the Internet. Previous technologies simply don't provide the means to access application components across the Internet. Component architectures such as COM, DCOM, and CORBA were designed to communicate across secure LAN and WAN systems, which required a substantial infrastructure investment. Connecting business trading partners and even regional sites was often cost prohibitive and logistically infeasible. Few options existed for reporting over the web. At best, a list or table filled with data could be viewed in custom-built, server-side web page solutions using ASP or CGI. Each page had to be carefully designed and scripted at the cost of dozens, or sometimes hundreds of programming hours.

With the recent maturity of the web, a new generation of mobile devices is evolving that can connect users to company resources, email, documents, and databases. These laptop, hand-held, palm-top, and wrist-worn devices open new doors of opportunity and present new challenges for data presentation. Perhaps, it will soon be common for people to stagger around the streets, talking to themselves and staring blindly into space in a zombie-like trance as they are connected to the world through web-enabled cerebral implants! We can only hope!

To gain access to useful and readable information, data must be accessible over available communication channels (such as corporate networks and the Internet), easy to access, secure, and available in a variety or formats so that it may be viewed using available document readers or browsers – all compatible with different devices. Did I mention the need to support different *Operating Systems (OS)*, applications, and perhaps, without the installation of any custom software on the client device? This is the challenge.

How Does SQL Server Reporting Services Meet This Challenge?

SQL Server Reporting Services is a server-side reporting solution that meets all of these requirements and more. It can obtain its data from a variety of data sources that you can access using modern programming tools. That data may be grouped, sorted, aggregated, and presented in dynamic and meaningful ways. The structure of the data and the presentation elements may be transmitted across practically any communication medium, using an industry standard format, to just about any type of client or server computer or device. The resulting content may then be displayed in many standard formats using browsers and document readers. Further, the data itself may be consumed by standard and custom applications to be further parsed, imported, manipulated, and consumed. It's a truly remarkable innovation with incredible possibilities.

Since Reporting Services is based on .NET, it offers the advantage of integrating tightly with the Windows platform and benefits from the performance, scalability, and security inherent to the .NET Framework. When used in concert with BackOffice products like Share Point Portal, it can provide a comprehensive enterprise solution with little programming effort. Reporting Services can be used with ASP.NET and other .NET programming tools to produce highly customized, special-purpose solutions.

In Chapter 2, we will discuss the specific Reporting Services architecture that is used to perform all of this magic. In brief, functionality is exposed through an XML web service that may be accessed across a LAN or across the web. Reports may be rendered in program code or they may be accessed through a simple web address – like any other web page. Reports may be rendered in several formats. These include different flavors of HTML to provide compatibility with different browsers and devices, the Adobe Acrobat *Portable Document Format* (*PDF*) for uniform presentation and printing, as a graphic file, and in Microsoft Excel so users can slice, dice, pivot, and re-analyze the data. Content may also be rendered in XML and CSV formats to import and exchange data with a variety of applications.

Business Intelligence Solutions

Traditionally, BI solutions have been very costly and only accessible to large businesses that could afford them. *Customer Relations Management (CRM)* systems, *Online Analytical Processing (OLAP)* systems (or data warehouses), and analysis solutions have been available for many years from specialized vendors. However, they require costly deployment, training, and maintenance. By contrast, (this is the part I like the best) Reporting Services is available at no additional cost if you install it on a computer with a licensed instance of SQL Server. Reporting Services is an add-on to SQL Server rather than a stand-alone product. In a single server installation, you don't need an additional license and you can use it royalty free – so long as your database, and server products are appropriately licensed. For additional information regarding licensing and deployment options, please refer to Chapter 13.

Comparatively speaking, collecting data is the easy part. Most companies have been doing this for decades, but how they utilize all of this data is often another story. There is no doubt that effectively collecting data may not be so easy but it's something businesses have been doing for quite some time. Most companies have untold mega-, giga-, or even peta-bytes of "important" archived data residing in documents, spreadsheets, and various databases on backup tapes, disks and folders throughout their enterprise – with no hope of fully utilizing and gaining significant value from it all.

According to Tommy Joseph of Disney Interactive Group, "*BI is about more than just tracking product sales. It's about measuring performance, discovering patterns and trends; and measurable forecasting through statistical analysis.*"

An effective BI solution provides visibility to important facts at all levels of an organization, and gives people access to uniform data from different sources using familiar and easy to use applications. It ties together applications, documents, and data sources in a manner that lets people collaborate and communicate effectively.

BI systems are no longer a luxury but a necessity in many business environments. Today, having access to timely information can make the difference between having a competitive edge and being left in the dust behind competitors.

Who Uses Reports and Why?

In almost any organization, there is a universal condition that people in different roles and at different levels have different perspectives on information. This is typically most apparent in large corporations, where executive leaders who make financial and market-direction decisions have less exposure to the daily processes of the company than the line-level workers. Ask any executive and they will tell you that

the line-level worker doesn't have a broad perspective regarding the challenges and direction of the organization at a high level. Conversely, ask most of the line-level workers in the organization and they will tell you that the upper management and executives don't share their perspective of "real problems" and the daily pulse of the company. To a point, this is the natural condition of a healthy organization.

Bill Gates has spoken extensively about the *information worker* of the twentieth century. At all levels within an organization, people who have convenient access to accurate and appropriate information are empowered to take informed decisions that benefit the organization and the individual. This is rapidly becoming the case throughout many industries today and continues to change the way people work and are managed. Although this paradigm shift may be occurring for many people, organizations often struggle to provide the resources necessary to support workers who are eager to use information to make a difference in their environments.

Executive Leadership

Leaders simply must make informed decisions. They must fully understand their business environment and the competitive climate in which they operate. Access to market conditions, customer needs, and financial information can often make the difference between decisions that produce success or jeopardize the organization.

Decision support systems provide interfaces for executive leadership through dashboards called *Executive Information Services (EIS)*. Reporting Services installs them with a simple web interface and enhances integration with executive consoles through SharePoint Portal services and third party solution integration.

Managers

Inefficient business processes can no longer remain the status quo. Customers demand results and simply will not tolerate services or products that don't meet their expectations. Customers have choices and will quickly switch to a competitor if their needs are not met. Managers need the information necessary to drive customer satisfaction and make corrections, directing business processes and the effective use of people and other important resources.

Information Workers

In businesses today, workers are educated and given more freedom to solve problems and effect change. This category could be applied to workers at various levels within an organization, including the managers and higher level workers. Often, the customer service representative or service provider will be the only human interface a customer has with an organization. That person must be empowered to collect and retrieve information quickly and accurately. They must also be empowered to make corrections to - and to work with, not against - unyielding business processes. In the past, workers simply had to accept the way information was presented to them, as well as the inefficiencies of most automated systems. With greater demands on businesses, workers simply must have the means to acquire accurate and concise information that meets their needs – in order to work efficiently.

Customers

Many businesses can't afford to put people in front of their customers on a routine basis. Customers who can get the information, services and assistance they need may not demand that someone help them when it's not warranted. By making regular services available through customer-friendly automation and information portals, you can afford to offer assistance to customers who really need special attention. Customers often need to look up account and transaction histories, order status, and shipping information. Making these services available through a web browser, email, or a mobile device can provide a greater degree of customer satisfaction.

Vendors and Partners

Like customers, business vendors may need to interface with an organization to place orders, schedule service calls, and obtain status information. Making this information available in the most appropriate form will improve efficiency and ultimately business-vendor partnerships. Business vendors are often more accepting of special procedures and automated systems. Vendors can be trained to use more sophisticated systems to obtain product information, service orders, invoices, and other business-related information. Systems may be designed to interface and automate the download or exchange of information that enable a partnering business to work cooperatively.

Reporting with Relational Data

Transactional databases are designed to capture and manage real data as it is generated; for example, as products are purchased and as services are rendered. Relational databases are designed according to the rules of normal form and typically have many tables, each containing fragments of data rather than comprehensive information or business facts. This helps preserve the integrity and accuracy of data at the detail level, but it presents challenges for deriving useful information from a large volume of transactional data. In order to obtain information with meaningful context, tables must be joined and values must be aggregated. Although relational database systems may support complex queries, reporting against these queries routinely could prove to be slow and inefficient.

Reporting for Decision Support

Optimized data storage systems are for analysis and don't use normalized tables, and don't contain details at the transactional level. Tables typically have more columns and fewer rows and often contain descriptive values that would otherwise exist only in lookup tables. The purpose of a decision-support data store is to drive meaningful reports and analysis tools with a sampling of read-only, historical, data and not for keeping up-to-the minute details.

Data Warehouses

The usual approach for maintaining a decision-support system is to copy only necessary data from relational, transaction-support databases to a separate data store at regular intervals. Depending on an organization's reporting needs, this data dump (or import) is performed daily, weekly, or even monthly.

The transactional tables are joined and pre-aggregated to eliminate unnecessary detail. Surrogate key values and codes have little use since the transformed data is readable in a more concise form.

Data warehouses (and smaller subsets of data analysis called *data silos*), can simply be implemented as relational data stores that have been designed for analysis. There can also be special purpose data structures that store data in hierarchal or multi-dimensional structures. These specialized data storage structures are optimized for performing pivots and extensive calculations and aggregations against a large volume of decision-support data.

A data warehouse is typically a large, central store of decision-support data whereas smaller, more specialized *data marts*, effectively divide analysis data into business unit and divisional data warehouses. *SQL Server Analysis Services* (see http://www.microsoft.com/sql/evaluation/bi/bianalysis.asp) is capable of storing and analyzing data in both relational and multi-dimensional structures. Data warehouse systems often use special query expressions that have capabilities beyond that of SQL. The *Multidimensional Expressions (MDX)* language supports pivoting and slicing data cubes to derive informational facts for comparative analysis.

Eventually most businesses that generate reports from live data will experience a common anomaly. As the data in a transactional database is ever growing and changing, reports will reflect these subtle changes and show up on the bottom line. Different users may produce similar reports in a short period of relative time and will notice that totals and summary values slightly vary. With no other explanation, users question the accuracy of report data and assume that there is a problem with their data. Believe it or not, the technical term for this condition is known as "twinkling data". As this phrase suggests, the totals and data points aren't typically way out of line. They just seem to fluctuate slightly. Reporting Services helps this situation by "freezing" report data through *caching* and report *snapshots*.

You might not perceive this to be a problem but consider what might happen if a user had made data entry mistakes and in an effort to correct the errors, entered duplicate, corrected data and then deleted the old records. Or, they deleted the erroneous records and then re-entered corrected information. If a report were produced at the wrong time, summary values could be skewed significantly. Statistically speaking, the chances of this condition causing a crisis may not be significant, but over time, they may increase. Data analysis should therefore be performed on unchanging values that are updated at regular, predictable intervals. This problem is often addressed by building separate OLAP data stores used only for analysis and reporting against snapshot data imported from transactional databases at regular intervals.

The Reporting Lifecycle

Chapter 2 will discuss the reporting lifecycle in greater detail with the architecture that supports this process. Briefly, creating a functional reporting solution requires an understanding of user and business requirements. Existing data sources must be considered and new data stores must be designed to meet reporting needs. From this perspective, the process of creating useful reports consists of three activities:

- **Authoring**: With the available tools, reports are authored using the *Report Designer* in Visual Studio .NET. This interface is used to create data sources, queries and datasets, and the report definition.

- **Management**: Report management is performed using the *Report Manager*, a web browser interface used to manage and deploy report definition files, shared data sources, and configuration settings; it can also be used to view and export report data.
- **Delivery**: Reports may be delivered to a user *on-demand* through the Report Manager or a custom application; it can also be scheduled for delivery through s*ubscriptions*. Reports can be delivered in the form of a web page, document, file, or even via email.

Report Delivery Application Types

In the past, reporting solutions were typically delivered through a desktop application of some kind. Data was queried in real time, and of course the application had to be connected to the data source. Users also had limited opportunity to save reports for later viewing and usually printed them on paper.

Now we have many opportunities to view and interact with reports in environments where it may not be possible (or feasible) to connect to data stores. Reports may also be presented in different forms that offer multiple capabilities and compatibility with various devices and software.

Web Browser

Web browser-based solutions have become popular for a number of reasons. User accessibility takes on a whole new definition when special software isn't required on the client computer. Of course, a web browser makes information available for viewing over the World Wide Web, but browser-based solutions are also a compelling means to deliver information in a controlled business enterprise environment. Whether users access resources within their corporate intranet environment or over the web, the browser paradigm has significantly changed the approach to application delivery.

Some of the traditional challenges with browser-solutions are the lack of consistent support for client-side script and components. These issues have largely been resolved with server-side rendering mechanisms that output product-independent HTML content. For viewing offline content, HTML documents require links to external files, such as images, sounds, and video. These issues have also been resolved by using a MIME-encoded format call *MHTML* or *Web Archive* to encapsulate binary content within the page definition. Although not supported in all browsers, this format is a viable means to deliver extensible report content for live and off-line viewing. HTML 4.0 works on different types of computers across the Internet, within a LAN on newer web browsers and HTML 3.2 works with older browsers and on a portable or hand-held device.

Office Applications

Microsoft Office brings together a tremendous assortment of capabilities to assist report users at all levels. Microsoft Excel has been the mainstay tool for data collection and analysis. By rendering a report into Excel, the data may easily be reformatted, modified, or analyzed using formulas and calculations. This capability has been around for several years but it required writing custom code to use the Excel object model from Access or Visual Basic to produce report data in Excel; in addition, this process was tedious at best. Now, pushing complex report data into a useful and well-formatted Excel document is simple.

Microsoft Access continues to be the office worker's database of choice. Data tracking and management solutions can be created with minimal cost and effort. Report Services may be used to exchange and import data into an Access database using XML or CSV formats. Access and Excel both provide the Office Web Components that may be used to view pivot tables and charts. These components duplicate the functionality of the Matrix and Report Services *chart* items but might give users a more convenient option for analyzing data.

Programmability

The possibilities for incorporating report features in your own applications are impressive. All of the features of the Report Manager can be duplicated in many cases and can be extended through program code. Reports may be viewed in place within an application by using an external web browser window, integrated browser control, or a custom report viewer component. Report content may be rendered to a file for persistent storage to directly into a viewer or browser.

Subscriptions

Subscriptions allow users to receive or gain access to reports on a regular schedule. Reports are delivered by email or saved to files where they may be viewed offline at the users' convenience. Report subscriptions may be setup for an individual user or large groups of users using data-driven subscriptions. To put this into perspective, effectively reports may be delivered to any individual or size group of users in practically any readable format at any place and any time.

Report Formats

In addition to the three HTML rendering formats, you can use document types to control formatting elements, printing layout and adding other capabilities. The PDF document format remains the most popular means for assuring that documents are formatted exactly as they were intended. Rendering a report to a Microsoft Excel workbook gives users the ability to continue to message data and perform calculations.

Importing Data/Exchanging Data

Not all "reports" may be intended to be read or printed. Reporting Services provides two report rendering formats that can be used for export/import and data exchange. Using either the *Comma Separated Values (CSV)* or XML formats, Reporting Services provides a very convenient mechanism for inter system data exchange or pushing data out to a trading partner. Imagine your system automatically sending invoices and shipping manifests to your order fulfillment vendor at end of the day via XML file attachments to email.

System Requirements

The hardware system requirements for Reporting Services are very similar to those for SQL Server. The default installation will place the Report Manager, Reporting Services, and the Report Server database on the same physical server but this configuration is not a requirement. These components may be installed on three separate servers.

The Report Server and the Report Manager servers must be running *Internet Information Services (IIS)* 5.0 or higher with ASP.NET, and the .Net Framework 1.1 or higher. The Report Server Database requires any edition of SQL Server 2000 with Service Pack 3. The SQL *Agent* and the *Distributed Transaction Coordination (DTC)* services must be running.

Editions of Reporting Services correspond to editions of SQL Server and include Evaluation, Standard, Development, and Enterprise editions. Like SQL Server, Standard Edition is a good solution for a single-server environment with a moderate number of users.

The Enterprise Edition has additional capabilities to support many users in a scalable environment using web farms and clustering technology. Subscriptions may be managed for users and recipients whose names and email addresses are stored in any accessible database.

The following table shows the requirements and features for each edition of Reporting Services:

SRS Edition	Operating System	Requirements
Enterprise	Windows 2003 Server	Must be configured as an Application Server SQL Server 2000 Standard or Enterprise with SP3
	Windows 2000 Server	Windows 2000 SP4 SQL Server 2000 Standard or Enterprise with SP3
Standard	Windows 2003 Server	SQL Server 2000 Standard or Enterprise with SP3
	Windows 2000 Server	Windows 2000 SP4 SQL Server 2000 Standard or Enterprise with SP3
Developer	Windows XP Professional	Windows XP SP1 SQL Server 2000 Developer, Standard or Enterprise with SP3 limited to 10 database connections
	Windows 2000 Professional	Windows 2000 SP4 SQL Server 2000 Developer, Standard or Enterprise limited to 10 database connections
Evaluation	Windows XP Professional	Windows XP SP1 SQL Server 2000 Evaluation, Developer, Standard or Enterprise with SP3 limited to 10 database connections
	Windows 2000 Professional	Windows 2000 SP4 SQL Server 2000 Evaluation, Developer, Standard or Enterprise with SP3 limited to 10 database connections

Installing Reporting Services

This section describes the steps for installing Reporting Services for SQL Server 2000. Reporting Services may be installed using one of two methods. A standard, Windows setup package provides a wizard dialog to guide the user through the entire process. It will prompt for file paths, server and other resource names and allow for optional component selection. The other installation method is a command-line utility and may be used, typically by system administrators, to script or automate the installation process. This method is not quite as user-friendly and requires more planning and a better understanding of the product and its components.

Setup Options

The following options are offered during setup as shown in Figure 1-1:

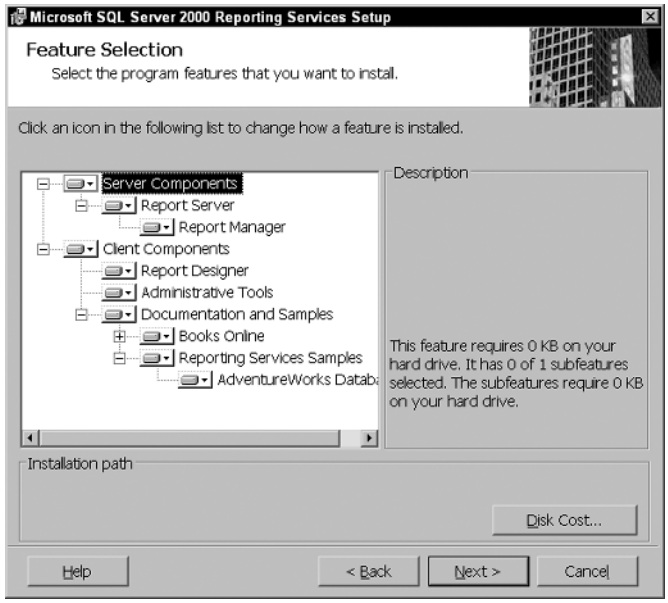

Figure 1-1

You can select the feature options that you want to install. Some features are related to others and appear as expandable branches of the feature tree. A plus sign next to a feature indicates that the branch includes related features. You can also specify that all related features under a branch should be installed by choosing the option labeled Entire feature will be installed on local hard drive. Likewise, selecting Entire feature will be unavailable will set this mode for all features under this branch. See Figure 1-2:

Figure 1-2

A few options can be installed so the content is located on a central network fileshare. Choosing this option for Books Online, for example, will not copy these files to the local disk but will refer to files at a central location. In order to use this option, it is necessary for the setup disk image to be copied to a folder that will be accessible whenever this feature is used. Some features include the option to install and run from the network displaying the drop-down list shown in Figure 1-3:

Figure 1-3

Adding and Removing Options

After the initial product installation, features may be added or removed by running setup again from the Control Panel | Add or Remove Programs applet. The currently installed start is displayed for each feature. Any changes made in the setup dialog will cause features to installed or removed.

Server Components

Server components include the Report Server and the Report Manager. It consists of a Windows service that runs continually on the server computer, a .NET web service hosted in IIS and a SQL Server database. The Report Server database can be installed on only one instance of SQL Server per physical database sever computer. The database need not reside on the local Report Server computer but the server must be a member of the Windows domain or a server trusted by the domain.

Report Manager is an ASP.NET application that exposes reports, configuration, and administrative features through a web browser interface.

The Report Manager requires IIS 5.0 or greater to be running on the Report Server computer. The .NET Framework version 1.1 also must be installed on the server. This is an included feature of Windows Server 2003. On a Windows XP Professional system, SP1 is required. Windows 2000 Professional, Server, and Advanced Server require SP4. Windows XP Home Edition is not supported.

Client Components

This includes the Report Designer. This option installs the *Report Project* and *Report Wizard Project* templates and *BI Projects* project group into Visual Studio .NET 2003 or greater. The Report Designer enables developers to create report projects in Visual Studio to create reports and related data sources. A report project generates definition files in Report RDL, an XML grammar that defines report content, layouts, and data sources. These reports may be deployed directly from Visual Studio into the Report Server. Reports may also be deployed manually using the Report Manager.

The Report Designer and Report Server can reside on different computers to support a separate development environment. The installation requirements for the Report Designer are the same as those for Visual Studio .NET 2003.

Let's look at the documentation and samples for Reporting Services.

Books Online

The documentation for Reporting Services is the Reporting Services Books Online. Its format and behavior are similar to that of SQL Server Books Online. All Reporting Services - related documentation is contained in only one source. If you plan to install the server and client tools on different computers, you should consider including the Books Online with both installations.

Reporting Services Samples

The samples include example projects for Visual Studio .NET, SQL Server databases, import data files, report definitions, and scripts. The projects include Visual Basic and C# code samples to manage and render reports using program code. The samples also include several reports exemplifying various design techniques.

AdventureWorks Database

The AdventureWorks2000 sample database was created for demonstration and instructional purposes in future versions of SQL Server and related products. Like the Northwind and Pubs databases, it will include design elements and sample objects for SQL Server developers and administrators. Unlike Northwind and Pubs, the AdventureWorks database is not a simplified design. It has been created to model the design of a highly normalized, large-scale business database containing over 60 tables. We will use this database for many of our sample reports later in the book.

Administrative Tools

Command-line utilities provide scripting and command level access to server management, deployment, and configuration features. These capabilities are thoroughly discussed in Chapter 8.

The following command-line utilities are installed with the Administrative Tools option:

Utilities	Description
`rs`	Used to process script files for deploying reports, managing the Report Server database, making and restoring backups, and other administrative tasks.
`rsactivate`	Used to manage the Report Server Windows service on a local or remote Report Server computer.
`rsconfig`	Used to set or modify configuration settings for a Report Server. This includes security authentication setting and database connections.
`rskeymgmt`	Used to allow administrators to backup and restore private keys to enable access to this secure information.

Command Line and Unattended Installation

The setup may be run using command line switch to automate the installation process. This capability is provided by the standard Windows Installer 2.0. Although there is no command-line interface, the setup process may be scripted and settings can be specified.

Log Files

Reporting Services records event information in the standard Windows Application Log and in specific log files. Report execution logging is enabled by default and may be configured in the Report Manager. Specific settings for the Report Server are stored in the `RSReportServer.config` file. More granular tracing information may be captured in log files for a variety of application and server events and system errors. These logs may be helpful in analyzing usage and debugging specific problems. The log files are auto-generated using time-stamped names. The file names, categories, and locations correspond to the configuration files mentioned in the *Configuration Files* section of this chapter.

Email Delivery

In order for Report Services subscriptions to utilize email delivery, an available server must be configured for the *Simple Mail Transport Protocol (SMTP)*. Internally, Report Services uses *Collaborative Data Objects (CDO)* to send mail. These required components are part of the standard configuration of Windows 2000 servers and Windows Server 2003 operating systems but you should verify that they are installed and working. The best way to do this is to create a test subscription in the Report Manager and verify that it has executed and the message is received in the recipient's inbox. Subscriptions are covered in detail in Chapter 10.

The SMTP service is not required on the actual report server but an accessible SMTP server must be available to the server in order for this feature to work. If you don't have an SMTP server configured, or if you don't intend to use the email delivery option, simply leave these fields blank in the setup wizard.

Designing Reports

Chapter 3 will deal with the essentials of report design and followed by Chapter 4 will take more specific design elements to the next level. Reports fall into a few design categories which will be covered next.

Form Reports

A report can display a single record on a page with data from a table, calculations and just static text. Form reports can be used to print or display a letter, invoice, contract or informational sheet.

Tabular Reports

This is a fundamental style for reports that have repeated rows of data called data regions. Tabular data is repeated in free-forum bands or table rows with rows and columns. Either the list or table items may be used to produce a tabular report in various layouts. Column headers can be displayed for each column in a table and subtotals and summary information may be displayed in table or group section footers.

Groupings and Drill-Down

Records in a report may be sorted and grouped. Each group can be collapsed and expanded to drill-down into more detail. This capability gives users the ability to explore large sets of data without the need to scroll though long, multi-page reports. The report may also be printed in its expanded form.

Drill-Through Reports

A drill-through report can be any standard form, tabular or pivot table report that contains links to a separate report. Any textbox item may used as a link to provide drill-through capability. Key values are hidden with the link and passed as a parameter to the target report for filtering.

Multi-Column Reports

A report may contain multiple columns. List or tabular rows are repeated vertically within a column and then *snake* from one column to the next, filling the page. This type of format is ideal for optimizing page space for labels and contact information.

Matrix

A matrix is like a cross-tab or a pivot table in which the rows and columns roll-up summary values and may be expanded or collapsed to expose more detail. It is a simple and easy-to-use control, much like the datagrid control in ASP.NET.

Charts

Charts are used to display a graphical representation of data, typically aggregated along at least two axes. Common types of charts are bar and column, pie and donut, line, area and scatter charts. More specific types of charts like stock and bubble charts are more specialized.

Data Sources

Reports can obtain data from standard data providers supported by the .NET Framework. In addition to SQL Server versions 7.0 and 2000, this list includes Oracle, Access, Excel, Informix, DB2, and any other databases and data sources accessible via an OLE DB provider or ODBC driver. Non-relational sources such as Active Directory Services, Exchange Server, and OLAP sources like Analysis Services can be queried. Developers can create custom data provider extensions— when an OLEDB provider or ODBC driver does not exist—to make practically any type of data accessible to a report.

Queries

Each report contains a query expression within its definition. A standard Transact-SQL query builder tool is incorporated into the report designer, capable of producing complex query expressions to be stored in the report. Although this is the defacto behavior of the designer, keeping queries in the report may not always be the best practice. Using a view or stored procedure from a SQL Server database can be a far more efficient method to query enterprise data. Parameters passed to a stored procedure cause the precompiled query to be processed on the database server before data is transferred across network connections.

OLAP Reporting

Decision-support databases come in many sizes and shapes. In its simplest form, a reporting data source can be a relational database with a few tables that can be queried more easily with some joins to other tables. Unlike transactional databases (often called online transaction processing or OLTP systems), OLAP databases are designed for efficient read-only access and reporting.

Large-scale OLAP databases require special storage and retrieval engines. Data may be managed in a cube structure, which enables values to be summarized and aggregated into slices and pivots. To query these structures effectively, MDX is used—much like SQL is used for relational structures. Reporting services supports the use of MDX queries although the reporting engine was not specifically built around this capability. When an OLAP data provider is used as a data source, the Report Designer displays the Generic Query Designer, which supports MDX expressions. The dataset resulting from this type of query is flattened into rows and columns like any other dataset. For this reason, all report items are supported for MDX queries.

Using Visual Studio .NET

Visual Studio .NET is Microsoft's integrated development environment tools for developing all types of applications. It replaces several different development tools in previous versions. Visual Studio now supports many different programming languages and file types. Whether you are an application developer or not, Visual Studio is the only tool currently available to design and build reports for SQL Server Reporting Services. Because of the extensibility of Reporting Services and the RDL XML grammar, other design tools will likely become available soon.

Report Wizard

The Report Wizard is a simple way to get started creating reports. It leads a designer though all of the steps necessary to build a simple report interface. New designers will find it an easy, uncomplicated tool for creating or choosing a data source, creating a query, selecting fields for header, grouping and display values, and choosing report styles and format options. Experienced designers will likely not find the wizard helpful as they become more familiar with the design process and prefer to have more control of these options. After completing the wizard, the report design may be extended and tuned to provide more functionality.

The .NET Framework

The Microsoft .NET Framework is a completely new direction for Microsoft, and replaces the *Application Programming Interfaces (API)* and object technology of the past. It's far more than a marketing strategy or a product. It gives application developers the objects and building blocks to create powerful applications of all kinds. Design and debugging features are also available in it to help developers through the tedious application development process. Utilities and compilers enable applications to be configured, compiled, and deployed. A runtime environment manages execution, resource allocation, security, and interoperability with other services, servers, and operating systems.

The main thing to understand about .NET is that it is a core component of Windows and it supports applications at many levels. The runtime and the development support tools are free. Visual Studio .NET

is a development tool that gives developers convenient access to these design and development capabilities.

Reporting Services is built on the .NET platform. The Report Server runs as a Windows service and is a .NET-managed assembly. Rendering and management features are exposed as an ASP.NET web service. The Report Manager is an ASP.NET web forms application. Finally, the report metadata, subscriptions and configuration information is managed in a SQL Server 2000 database access through the SQL Server ADO.NET data provider. As you can see, Reporting Services is purely a .NET solution.

Custom Reporting Extensions

On the advanced end of the opportunity scale, reports can be extended and enhanced in a variety of ways. At the core of the Reporting Services architecture is a set of extendable programming interfaces that enable the use of custom components written with .NET programming tools.

Data Processing Extensions

The .NET Framework includes native support for connecting to standard data sources using the SQL Server, OLE DB, ODBC, and Oracle .NET data providers.. However, to report on non-traditional types of data, developers can create custom data processing extensions to expose practically any type of data as a data provider, for instance. If a cache of data could be held in memory rather than written to disk. Another example would be data stored in files using a proprietary format.

Delivery Extensions

Reporting Services supports subscription delivery via email or file output with no additional programming work. Additional delivery options can be added by creating a custom delivery extension. Using a custom solution, reports could be sent to a message queue, FTP site or practically any other destination.

Configuration Files

Options and settings are stored in XML-based text files that are easy to edit and maintain. The purpose, location and content of these files are detailed in Appendix E. Configuration files include the following:

File Name	Purpose
`RSReportServer.config`	All settings that apply to the Report Server including connections and security, caching, and subscription delivery options.
`RSWebApplication.config`	Settings that apply to the Report Manager web application. Most settings correspond to options in the Report Manager application configuration pages.

Table Continued on the following page

File Name	Purpose
`ReportingServicesService.exe.config`	Enables tracing and logging of certain server events that include restarts, exceptions, warnings and status messages. Some settings are used to manage the trace and log files and tracing output options.
`RSReportDesigner.config`	Used to manage and configure custom data processing and rendering extensions. Also sets designer preview and rendering options.

Scripting

Most report management and delivery features may be automated through a simple scripting interface. A single utility executable, `rs.exe`, is used to obtain access to the vast capabilities of the Report Services web service. You can create scripts to manage batch processing of reports or programmatically manipulate any exposed functionality of reporting service. Capabilities are similar to that of the web service proxy used in .NET programming code but a scripting solution is a simpler approach that doesn't require complex programming or a compiled project. Scripting is an ideal approach for system administrators to create simple maintenance, deployment and ad-hoc delivery solutions. Chapter 8 will detail report scripting options.

Subscriptions

Subscriptions allow users to request reports to be delivered to them automatically. Based on a schedule (single-instance or recurring) reports may be delivered using any available deliver extension (email, file or custom) in any available rendering format. Subscriptions can be either standard, where a user requests the scheduled delivery of a specific report, or data-driven, where a group of users can request the scheduled delivery of one or more reports. This is an extremely powerful tool that can be used to provide report content in an efficient manner to users in practically any location or work schedule. Chapter 10 will lead you through this compelling feature.

Securing Reports

Reporting Services uses a role-based security model that is installed and configured by default. This model is highly extensible and may be changed after installation to use a custom authentication component.

In order for sensitive data to be protected from intrusion, it should be encrypted both at the Report Server and in the web browser or client application. The preferred method to do this is to use Reporting Services' built-in support for certification-based encryption over the *Secure Sockets Layer (SSL)*. Implementing SSL will automatically redirect web requests to an address at the same location using the

http:// prefix. This enables bidirectional encrypted streaming of all data over port 443 (by default) rather than the standard HTTP port 80. Reporting Services supports levels of automatic encryption which are detailed in the section that follows. There is currently no maintenance interface for this setting through the Report Manager or any other provided utility.

You will need to obtain a digital certificate from a certificate authority such as Verisign, AuthentiCode or Thawte. These companies will sell or lease the certificate for a specified period of time for a few hundred dollars per year. The authority will do a background check on your business to verify you are legitimate. Configuring the certificate is actually quite easy. This is performed using the IIS management console and setting the properties for the `ReportServer` web folder.

To enable encryption in Report Services, edit the `RSReportServer.Config` file using Visual Studio or a text editor and set the `SecureConnectionLevel` element to a value from 0 to 3.

In the `RSReportServer.Config` file, you will find this setting near the top of the file after the encrypted login and connection settings:

```
<Add Key="SecureConnectionLevel" Value="0"/>
```

Modify only the number value between the double quotes and save and close the file:

The values are as follows:

Value	Description
`0` (default)	Disables encryption support
`1`	Enables minimal support but may expose credentials if a certificate isn't installed or configured properly
`2`	Provides an appropriate level of encryption for rendering and authentication calls
`3`	Encrypts all data

The Report Manager

The Report Manager (shown in Figure 1-4) is a web-based interface that provides both user-level access to reports and administrative features to configure security, subscriptions, report caching, and data access.

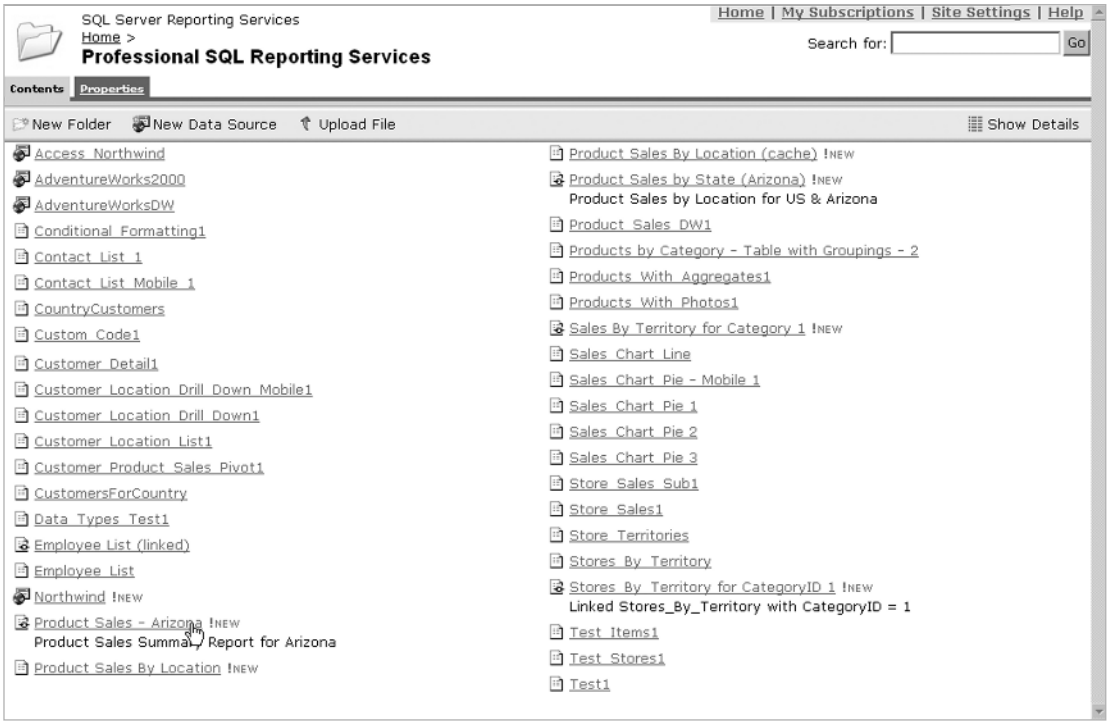

Figure 1-4

This web application is used to perform report and server administration as well as report delivery. Users may use it simply navigate to reports, provide parameter values and view them. The Report Manager will be discussed in detail in Chapter 6.

Designing Reports

In this first release of Reporting Services, reports are designed and created in Visual Studio .NET using a special type of project especially for report design. Simple reports can be built with little effort using the report wizard. The wizard (shown in Figure 1-5) leads the user through all of the steps necessary to produce a variety of useful but simple report designs.

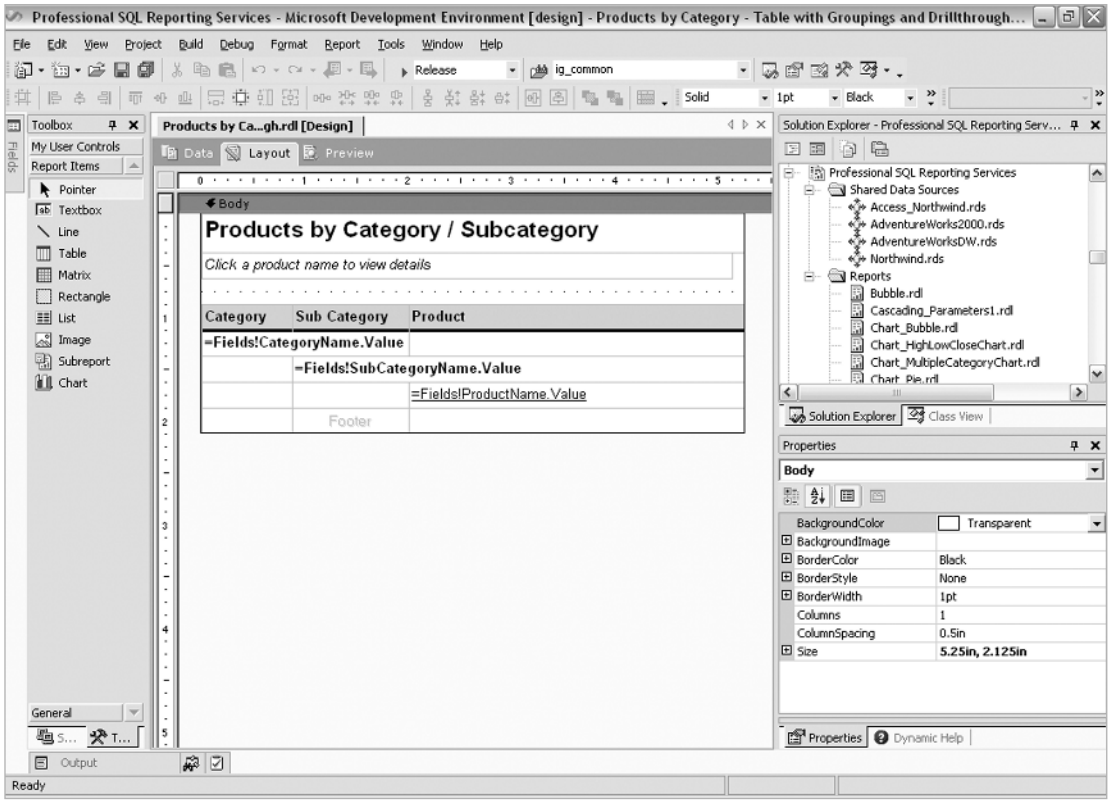

Figure 1-5

Chapter 3 will lead you through a series of exercises to get you started with simple report design and lay the foundation of the report design elements.

In Chapter 4, we will explore data sources and datasets that provide data for reports. We will discuss different query types like SQL expressions, MDX expressions, views and stored procedure. Query parameters and report parameters enable data to be filtered at the right level and to make the best use of data, server and network resources.

In Chapter 5, we cover the spectrum of report design items, data ranges and formatting tools. By using groupings, we can design multi-level, hierarchal reports. Drill-down reports let users interactively expand groupings and discover more detail without having to navigate through many pages of content. Drill-though reports let users navigate from one report to another, passing filtering parameters to obtain detail information about items in the report. Navigational links may also be used to drill-through to external resources like web pages, documents and email links.

Charts are useful for aggregating values and presenting a series of data for comparison. A number of standard charts are available including bar, column, line, area, pie and doughnut charts. Specialized charts types like scatter, bubble, and stock charts are used with multi-dimensional data and values in distinct ranges.

Report formatting and content may be enhanced by using program code in a few different ways. Custom functions may be written in a block of code that is embedded into the report. These functions may then be called in various property expressions providing conditional formatting and business rules. More complex code routines may be built into a class library and exposed to reports as custom assemblies. An assembly is deployed to the Report Server and may be shared by many reports. Finally, custom extensions may be written to replace or extend inherent data source and rendering capabilities, providing custom capabilities beyond those built-into the product.

URL Access to Reports

The Report Manager environment is the default entry point and a convenient, comprehensive interface to view reports. However, one easy method to view a report is to simply navigate to the report's web address provided by the Report Server. URL query string parameters are used to specify a variety of options including rendering formats, filtering parameter values, and display options. This is a simple method for managing and using reports right out of the box—without additional programming or configuration.

Rendering Reports in Program Code

Possibly the most unique characteristic of Reporting Services is the way it renders report content. Unlike traditional report solutions that use a proprietary, custom viewer to render the report content; at its core, Reporting Services is built on a programmatic interface (an XML web service) that outputs the entire contents of reports in several different file or rendering formats. This capability gives programmers an incredible range of options for creating custom solutions. These options may include:

- On a simple web page, users could click a link to display a custom report in their web browser using simple URL rendering.
- In a custom ASP.NET web application, users provide filtering criteria on a web page; click a button and view the resulting report in a secondary browser window without navigating off the application's web site.
- In a desktop application, users provide filtering criteria and view the report within the desktop application form.
- Custom reports are saved to an Adobe Acrobat (PDF) file that may be viewed offline on a laptop, Pocket PC or other mobile device.

An in-depth discussion of programmatic rendering may be found in Chapter 9. Even for the novice programmer, creating these kinds of solutions is relatively simple and may be accomplished with just a few lines of program code.

Report Definition Language

Rather than defining a proprietary specification for individual report definitions, our friends at Microsoft took a very different approach. They chose to publish an extensible and well documented standard. The entire set of instructions that define a report are stored in a single XML document using an RDL XML grammar. If necessary, property values for elements of a report's design could be modified

with a text editor. If someone wanted to build their own report design tool, they would simply need to output the appropriate XML tags to an RDL file. This also makes it easy and convenient to send the report definition to someone or to deploy a report to another server.

In chapter eleven, we will use a third-party tool to build reports from an RDL template. This will enable us to create new reports without using Visual Studio .NET. This exercise should open your eyes to the possibilities to design and build reports from your own custom software, extending vertical business systems and making Reporting Services part of a complete business solution.

Deploying Reports

Reports are defined in an RDL file but the report's definition is stored in the Report Server database once it has been deployed to the server. Report deployment may be performed in at least three different ways. In Visual Studio .NET, the project defines a corresponding web folder on the target Report Server. Building a report project will deploy reports to a designated target Report Server. The Report Manager web interface may be used to deploy individual reports manually by simply browsing for and selecting the RDL file. The Reporting Services web service may be used to deploy reports programmatically using methods of this multi-purpose object. Chapter 13 will explore each of these options and detail deployment techniques and related considerations.

Designing and Architecting Report Solutions

Reporting Services does offer an out-of-the-box solution. Reports can be designed in Visual Studio .NET, deployed to a server and viewed using the Report Manager Web interface quite easily. However, for custom applications or to meet specific business needs, this may not be the ideal solution. Reporting Services is an extensible service with several options for designing, managing, deploying, rendering and delivering reports to users.

In Chapter 13, we will discuss these options and consider how understanding your business requirements should lead to the most ideal solution. We will look at different business cases and how a reporting solution fits into the overall picture to meet business and users' needs now and in the future.

Summary

At this point, you should understand that Reporting Services uses a new approach for report delivery. Each report has a data source that may be shared with other reports. A data source can obtain from practically any database product or data provider.

Report definitions are stored in an XML document format called RDL. Out-of-the-box, reports may be designed in Visual Studio .NET but third-party and custom solutions may be used to create and design reports as well.

Reporting Services can be completely secured and highly customized. The Report Manager is provided to simplify server, user and report management. Solutions may be simple and easy to implement or they may be completely customized and integrated into your custom-built software.

Reports may be delivered using snapshots and subscriptions that are either pulled by the user in real-time, or pushed by the server on a schedule. Using these capabilities, valuable system resources are conserved since reports are rendered less often and can be cached in the Report Server database

Setup is performed using a standard Windows Installer package. We've explored the setup options and selections. Configuration settings are stored in XML configuration files that may be modified by an administrator using a text editor.

The next chapter will help you understand the architecture that makes Reporting Services work. You will learn about the nuts and bolts that give this impressive product the ability to provide scalable and extensible reporting solutions. Throughout the book, we will build on this foundation as you learn to design, manage and deploy your own reporting solutions.