CHAPTER 1

Introducing SQL Server 2005 Integration Services

SQL Server 2005 is built on solid foundations. Microsoft has repositioned the product many times over its life cycle, adding new parts and facets over the years and removing others. Young technologies have matured and become a staple of the SQL Server experience. The database engine alone has evolved beyond recognition since the original release and provides a robust, incredibly scalable database platform.

One of the major additions to the product has been an increase in business-intelligence (BI) capabilities. Data Transformation Services (DTS)—Microsoft's introduction of native Extract, Transform, Load (ETL) functionality into SQL Server—formed a major part of SQL Server BI. ETL is the practice of taking data from a source (extract), manipulating it in any of a variety of ways (transform), and finally channeling the manipulated data to a destination (load)—possibly for BI purposes. Perhaps because DTS had so many capabilities outside of the BI space, though, many SQL Server customers tended to overlook DTS in favor of third-party ETL tools such as Essbase, Datastage, and Cognos Transformer.

Once DTS launched, I stopped using third-party ETL tools, because DTS worked well and was simple to use. At the most basic level it provided an IDE as a part of SQL Server Enterprise Manager in which to use extraction, transformation, and destination components. DTS could be employed to manifest an ETL scenario such as populating an Online Analytical Processing (OLAP) database, based on calculations performed against an Online Transactional Processing (OLTP) source. DTS could also use components for modifying SQL Server settings and tables, and it had scripting components that took the basic idea of ETL far beyond comparable third-party software.

SQL Server 2005 Integration Services (SSIS) is essentially the latest iteration of DTS, enhanced and evolved into a highly versatile platform in the BI space and beyond. Leveraging new features of SQL Server 2005 such as .NET Common Language Runtime (CLR) scripting and built on a brand new processing engine, SSIS is at the pinnacle of ETL evolution. SSIS is DTS pulled apart and put back together in a more logical and mature way with new and improved features that overshadow DTS in many respects.

This chapter gives you a snapshot of the whole of SSIS to prepare you for diving into much greater detail in the following chapters. You'll find that SSIS offers a tremendous amount to work with, including its visible features, the features under the hood, and its capabilities for integrating with and extending .NET technology.

A Little History

My relationship with SQL Server began in 1995 with version 6.0 (SQL95). I remember being extremely impressed with how much potential it had and yet how easy it was to use. Now even my smallest projects could use a robust relational database management system (RDBMS), instead of Microsoft Access 2.0, at low cost. As a developer, I felt completely at home with SQL Server, in stark contrast to products like Oracle Database. I could install, configure, and implement database designs without having to turn to a database administrator (DBA) just to add a new table to my schema. SQL Server demystified the black art of RDBMS management: for many professionals it highlighted the possibility that Oracle Database was too obfuscated and abstracted power away from all but a few. SQL Server gave instantaneous and transparent power to the developer, and at a much lower cost.

I didn't hear much about ETL tools in 1995. I'm sure they existed but were considered much too expensive for my small company. We performed all data manipulation through either SQL scripting or Visual Basic 3 components based on the Windows 3.11 platform. These VB3 tools didn't have a user interface (UI), of course—certainly nothing approaching SSIS or even DTS—but they were fairly powerful and a great demonstration of the possibilities that lay ahead.

ETL capabilities became an intrinsic part of the SQL Server family in version 7.0, a release that saw a surge forward in features, capabilities, and possibilities in the OLAP space. I was working on a remote-data–capture project that involved processing and transforming a staggering amount of data—coin usage in video-game machines. Although the company concerned ran SQL Server 6.5 and 7.0 concurrently, the business value I could offer by implementing DTS in a fairly standard way earned me DBA status. I felt this title was inaccurate; I would say I was becoming a data artisan!

The effectiveness of DTS and SQL Server Analysis Services (SSAS) helped me induce an investigative group to repeal the company's decision to use Business Objects. I demonstrated that the benefits offered as part of the SQL Server package could equal or even surpass facilities within Business Objects. In pure business terms, it made sense to use a tool that had already been paid for and yet was more than powerful and capable of doing the job right, vs. a tool that cost many hundreds of thousands of dollars and required specialist consultants offered at $1,500 per day.

I expect that this is a similar scenario to one that many developers around the globe are seeing today with SSIS. The difference is that back then I wasn't 100% sure that I could back up my claims with a solution built entirely with DTS. If I'd had SSIS at the time, I probably wouldn't have allowed Business Objects to set foot in the door!

Looking back on all this reminds me how far Microsoft's ETL strategy has come since its first tentative steps with DTS. Even with DTS straight out of the box, however, I was able to match and surpass existing products' functionality. As you might imagine, I was like a kid at Christmas the day I got hold of SSIS. Any concerns about whether I could use existing DTS packages were forgotten for the moment. If DTS was great—if a little shaky at times—what would SSIS bring to the table? As I unwrapped SSIS I realized it was a gift that just kept on giving. SSIS has it all: high performance, scalable ETL, a brand new architecture, and custom capabilities.

What Can SSIS Do?

If there is *anything* you want to do with your data—or even the database itself—SSIS is the tool for you. You'll start developing SSIS packages in either Business Intelligence Development Studio (BIDS) or the full version of Visual Studio 2005, as shown in Figure 1-1. If a feature you want isn't available by default, you can either write it yourself or download it. In fact, with a little thought and a variable degree of effort on your part, there isn't anything SSIS *can't* do.

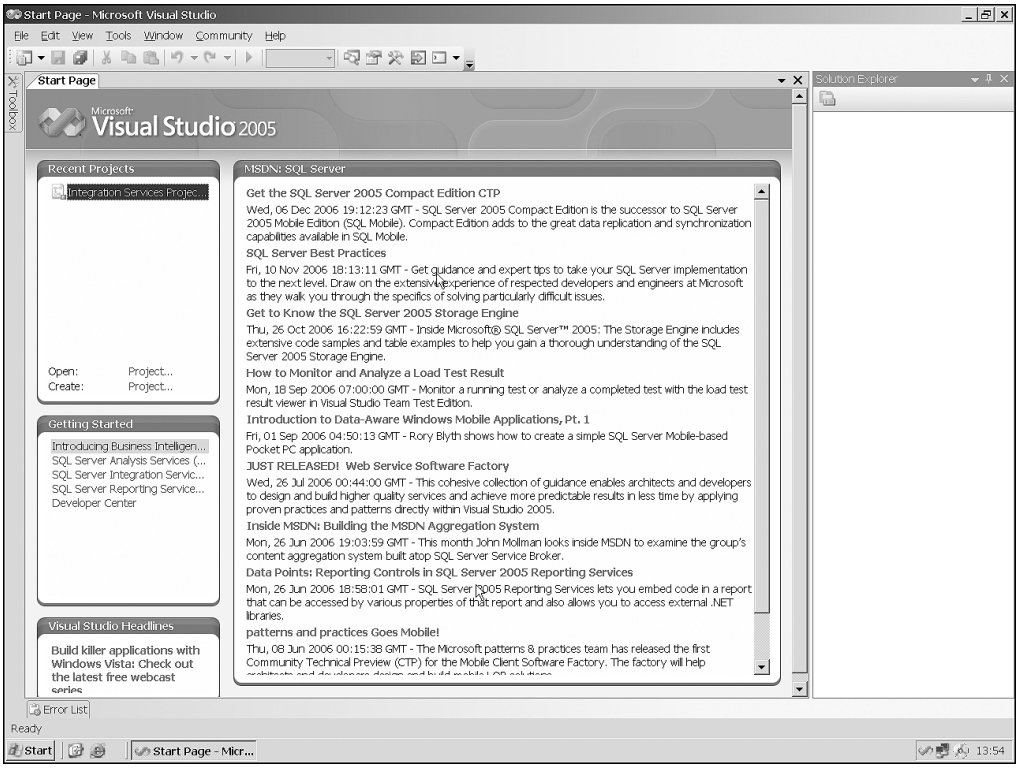

Figure 1-1. *Does this look familiar? It's where you'll be developing SSIS packages.*

Here's a brief look at some things SSIS can do straight out of the box:

- **Import/export/transfer**: By specifying the data source and destination, you can import and export content in whichever format is required without even using a transformation. Any defined imports or exports are probably the simplest elements of a package.

  Also, *Control Flow* items provide a range of capabilities for transferring database objects such as table structure and contents, or programmability items, between servers.

- **Web services**: Your SSIS package can access an XML web service as part of your Service Oriented Architecture.

- **Maintenance tasks**: You can choose from all kinds of maintenance tasks, from backups to rebuilding indexes to shrinking the database.
- **Fuzzy matching and grouping**: Microsoft Research Labs has supplied a commercial-quality implementation of fuzzy matching and grouping that is fully integrated into SSIS. Fuzzy matching is the practice of discovering inexact matches between textual values based on the algorithmic processing of input. The implementation in SSIS is fully configurable and allows many options for tweaking how the matches are made.
- **Data and text mining**: Crossing over from SSAS, mining functionality is an important tool when it comes to "discovery" of information within data. These components are an excellent example of how data mining (and now text mining) should be done.

Of course these types of tasks are great, but seeing them as stand-alone components is really missing the point. The real power comes from combining different components of SSIS and building them into a useful package.

For example, with SSIS you can

1. Wait for a file to arrive via FTP
2. Conditionally split the data into two in-memory streams where the data types are converted or a derived column is created
3. Recombine the data and insert it into a waiting table

And it's fast. You want to make sure it's performing as expected? Drop a row counter in there to check. Or use a visualizer. In SSIS, putting together a package that does all this is not only straightforward but takes only a few moments.

Figure 1-2 shows just how simple this is to perform. I simply created an empty SSIS project, dropped an `FTP Task` from the toolbar on the Control Flow tab, and configured the FTP server details (specifying that a file was being received, not sent). I then added a `Data Flow Task` and connected the two tasks together to form a flow so that once the FTP task completes successfully it executes the `Data Flow task`. Figure 1-3 shows the Data Flow.

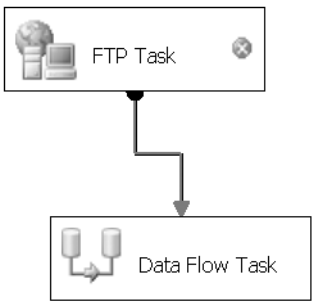

Figure 1-2. *A simple Control Flow*

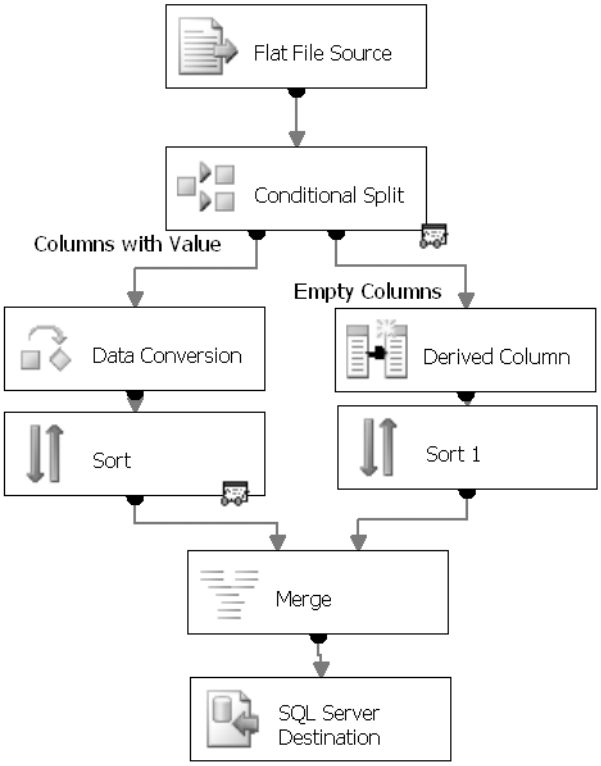

Figure 1-3. *Inside the Data Flow*

Adding the `Flat File Source` pointing to the file that will be delivered via FTP lets me easily define the necessary tasks:

`Conditional Split`: I configured this component to split the data based on whether or not the first column holds data. This gives two outputs from the component.

`Data Conversion`: This component was configured to convert the data from the `Columns with value` output to a `CHAR(2)` type.

`Derived Column`: I used this component to derive a `CHAR(2)` value from subsequent columns in the source file.

`Merge`: This merges the two datasets back together ready for output.

`SQL Server Destination`: This writes the dataset into a waiting table.

In all, this took only a few moments to put together. I was able to add two visualizers to the flow to ensure that the correct columns are being derived and that the conversion takes place as expected. This allows me to see a grid (or graph) of the data contained in memory at that point of the flow.

Don't worry too much at this point about how this example works and what tasks are actually being performed. I am including it only to demonstrate how even limited exposure to SSIS can yield enough insight to produce a worthwhile, useful package quickly. In fact, my initial experience with SSIS in the UK allowed me to identify the potential of the technology for use within a current project. Budgets were tight and time was limited, but through just a brief introduction to SSIS I could see that to succeed, I had to migrate the majority of data processing to this new platform.

The First Time Ever I Saw SSIS

It was the height of summer 2005, and I was meeting at Microsoft's UK offices with two Microsoft architecture evangelists for my first discussion with Microsoft about the new Visual Studio and SQL Server 2005 products. I clearly remember the look of excitement on one evangelist's face as he showed me SQL Server 2005's capabilities and repeated breathlessly, "Look how fast it is!" I was there to discuss how the project I was working on could best meet the criteria for the Development and Platform Evangelism (DP&E) team to promote the project through Microsoft. We'd already looked at addressing the web-enabled area through implementing ASP.NET 2.0, so the next logical area of this incredibly data-processing–intensive project to address was the database. Having looked at various new features of SQL Server that, while interesting and useful, weren't a great fit, we started looking at SSIS. The doors of perception were flung open to the wall!

Transformation processes that I'd written for the project in an object-oriented fashion in C# meant that data transformation was an overnight task that, with more data being collected on a daily basis, was only going to take longer and longer. With SSIS I was able to reengineer the solution using, initially, only the standard components. Not only was I able to implement asynchronous processing easily, but I could use the data viewer and row counter in SSIS for rapid tracing of problems in complex transformations that had taken hours to debug in C#. Processing time dropped exponentially, and development time was cut in half.

As a side note, something incredibly interesting happened during this project when I invited Google to tender a bid for the use of its Search Appliance. We didn't get as far as discussing costing; the European sales director took a long, considered look at how much data was being handled and the terabytes of output from SSIS . . . and stopped returning my calls! This speaks volumes about the capabilities of SSIS in handling massive sets of data.

The redevelopment of the ETL process was a rapid and pleasantly coherent one. The Microsoft SSIS team has worked hard on including features most used and most useful to those interested in getting unequalled value from their data.

I Have Such Delights to Show You

I have had a fantastic time learning the intricacies of SSIS. This book presents everything I've learned about SSIS, ready for assimilation. I will begin by taking a peek under the hood of SSIS and then open up the IDE to impart knowledge on

- **The architecture of SSIS**: How data is handled behind the scenes in SSIS and how the architecture can be leveraged to provide support for the most demanding of applications.

- **Migrating from DTS to SSIS**: The how, when, and why of moving from DTS to its big daddy, SSIS, observing potential pitfalls and ways to keep legacy DTS packages running as normal.
- **Visual Studio**: Information for readers unfamiliar with Visual Studio, such as DBAs and non-.NET developers, on Visual Studio as a mature and complete IDE for all kinds of CLR languages and SSIS projects.
- **Package hierarchy**: The king is dead, long live the king! Packages as you know them have changed dramatically and are now made up of a number of different components and logical concepts. Containers, Control, and Data Flow—oh my!
- **Control Flow and Data Flow components**: Content on Control Flow and Data Flow presented as both a tutorial and a source of reference on every one of the components you can use within a package.
- **Optimization**: Information not only on optimization of SSIS components but also championing the idea that forethought in database design can yield dramatic performance benefits.
- **.NET languages and object-oriented programming (OOP)**: Again aimed at those less familiar with the concepts of OOP and .NET in general, a detailed and concise body of work containing everything you need to know about leveraging the integration between SSIS and .NET.
- **Custom components**: Almost everything in SSIS is customizable. You'll learn how to create custom components that can be used in a variety of ways within a package.

By the end of this book you will be able to take advantage of the best features of SSIS, avoid potential problems, and create the most complex packages imaginable as if you had been using SSIS for years. This is SSIS from the front lines.

Where would SSIS be, however, if it weren't for the people involved in the project at Microsoft? Their gifts of creativity and insight in understanding the needs of database developers have propelled the possibilities of SSIS beyond the stars. They give us the tools and we make things happen.

An Interview with Donald Farmer

In late 2006 I was lucky enough to secure an interview with a veritable SSIS VIP, Donald Farmer, Microsoft Group Manager for SQL Server Business Intelligence. Here are some important excerpts from that interview.

James Wightman (JW): *I'd be interested to know about the perspective from which you approached the implementation and feature set of SSIS. In terms of function, did you start by examining DTS with a look to reimagining it? Or did you (as I suspect) take a clean slate and work from there?*

Donald Farmer (DF): The team approached the implementation of SSIS from two perspectives: what capabilities were needed to enable SQL Server to succeed in the data warehousing market, and what architecture best enabled those capabilities. There was a balancing act required as to what extent we would take dependencies on SQL Server features, and to what extent we would build a fully stand-alone product.

JW: *Was the feature set (both internal and external) provided by the new SQL Server engine tailored in particular ways so as to better integrate (and give a more rich feature set to) SSIS? I ask this because SSIS certainly feels like a more integrated and natural product than DTS, which by comparison (using it now) almost feels like a third-party add-on.*

DF: This required us to work closely with the relational engine team—on bulk-load interfaces for example. However, there is also deep integration with the Analysis Services and Data Mining engines, and with the management toolset, all of which provides a more organic data-management experience for SQL Server users.

JW: *One big advantage over existing ETL tools is how SSIS enables developers to create their own custom components. I can imagine this presented a particular set of challenges and obstacles during both design and implementation. Was including this feature a major part of the SSIS project?*

DF: It is worth pointing out that the supported interfaces for extending SSIS with custom components are managed code, while most of the components we wrote ourselves are in native code. There are a number of reasons for this, perhaps the most important being the ease with which we can support managed-code APIs, samples, etc., and the growing popularity of managed-code languages. As for problems, the most interesting have been simply explaining to third parties just how radical the capability is.

JW: *In terms of function SSIS surpasses commercially available ETL tools such as SAS and Datastage. In terms of flexibility SSIS goes way beyond these products. Would you say that SSIS is also aimed at a different type of user than these other tools? Perhaps both a more technically savvy but not necessarily BI-focused developer as well as the more traditional BI/Management Information (MI) specialist?*

DF: I tend to agree with you on our flexibility—which is partly a result of a great architecture and partly a requirement that drives, as our user base is so broad. DTS in its day was a handy utility for DBAs, a low-end ETL tool for the data-warehouse architect, and an interesting environment for developers to extend. SSIS still needs to be a utility, is still a favorite with developers, and covers a much wider range of data-integration scenarios. I often talk about a particular kind of SSIS (and SQL Server) user that I call the "data artisan." They are database-centric developers who use tools when they can, write code when they need to, and also have a responsibility for architecture and administration. They do about four jobs—and SSIS is a major component in their toolkit. So yes, SSIS has created or extended a new role.

JW: *Is SQL Server used in the back end of Xbox Live to support the service?*

DF: At Microsoft, we're famous for eating our own dog food and yes SQL Server is used throughout the business for its high-end operations, including Live services, Xbox, and AdCenter. In some cases, these teams extend the application themselves with custom components into areas where there is not yet enough commercial demand for us to put those features into the retail product. Without giving away too many secrets of the internal teams, some things they have done to squeeze extra from the engines include writing highly specialized schedulers tuned for their precise needs, or data-source components that can parse their specific file types at near-optimal speeds.

JW: *Could you please sum up what you think SSIS offers over other tools meant for the same purpose?*

DF: Value for money for one thing! If that is not a critical factor, I love the interactivity of the visual debugger. I was recently visiting with a customer who uses another very capable data-integration tool. As I helped them with some warehousing problems, I wanted to tweak their ETL processes. In SSIS I could have dived in there, made some changes, run through the debugger with visualizers and breakpoints, and validated what I was doing using a row counter as a dummy destination. Instead I had to create temporary tables, reconfigure the server and the metadata repository, run the modified jobs, and then trawl through logs and reports. Coming back to SSIS I hugely appreciated the ability to roll up my sleeves and get on with the work. I guess I'm a "data artisan" at heart myself!

In speaking with Farmer at length I found my own feelings about SSIS and the genuine potential it holds to be justified and consolidated. Such conversations often make me question the wisdom of not using a data-integration tool such as SSIS with current "large" IT projects around the world, such as the proposed plan to connect all UK physicians' surgeries together and hold a central database of patient records. At last count this was estimated as a GBP £19 billion project that would take many, many years to develop. Depending on the desired implementation, there could be a clear case here for the use of SSIS and, by extension, web services.

Here's perhaps a smaller-scale example of a similar kind of system. During the latter half of 2006, I was involved in discussions around the data-integration policy for a large American vehicle manufacturer that has a parent company and many subsidiary brands. The discussions centered around the vast global "approved dealership network," in particular how best to implement interfaces between the many different systems that had been built for each brand.

Some of these systems were accessible only via MQ Series, some of them were Oracle-based, some used SQL Server, and some used a number of legacy databases that were available for only a few minutes each day because of firewall restrictions. Some of the systems accepted only comma-separated value (CSV) files that had to be parachuted in via FTP.

Historically, data integration was performed via individual—completely stand-alone—applications that connected on a system-by-system basis to a target system and transformed the source data to fit requirements before transmitting the data. Each interface was built from the ground up with no capacity for reuse of any kind. Seemingly, cost was the driving issue, in that project funding was available for only one interface at a time. This meant that by the time another interface was paid for, the original development team had moved on, and it was left to someone else to come in completely cold. When one of the integration interfaces stopped working . . . well, no one could remember who wrote it or where the source code was, so maybe it would have to be written all over again . . .?

My idea was simple and courted considerable derision. What if a single data-artisan–type developer could replace all of these singular interfaces with an all-encompassing solution that could be modified where necessary at any time, reused and added to when needed, and all within the time and cost of creating a one-off, one-shot integration application?

Once the incredulity had subsided I invited the senior decision makers to watch on a projector screen while from my workstation I created a prototype SSIS package that connected to each of the target systems via the protocol required, and performed some basic transformations to demonstrate how the data could be restructured for the intended target to accept.

"And look," I said. "Look how clean and efficient it is! No staging of data! No temporary tables! And look how quick it is!"

I had, in a few short moments, demonstrated the power of SSIS to an audience of senior and incredibly IT-literate individuals, including BI professionals, and left them agog. "What else can it do?" they asked. "And what are the limits? Could SSIS be the answer to consolidating and transforming all of our disparate data?" I tentatively suggested a concept I had been working on for a while: creating a *knowledge brokerage* within the company.

Become a Knowledge Broker

It has been a long-held dream of mine to create a company offering the service of what I call a knowledge brokerage. It's a relatively simple concept: harvest data from as many discrete and diverse sources as possible, whatever the format; reduce the data to its component parts; and then reengineer it so it offers knowledge rather than simply data and allows the asking of "questions" that reveal accurate and insightful "answers."

In some ways I suppose the idea is similar to what credit-reference agencies such as Experian and Equifax do, but on a much larger (and less structured) scale.

I can envisage a time when knowledge brokers compete with one another to offer the most complete and reliable (i.e., accurate and comprehensive) knowledge. It might sound a little Cyberpunk or even outright Orwellian, but central repositories held in secure data silos accessible only to authorized parties would be not only more efficient, but also more accountable and manageable by the subject of the information.

On a technical level, how difficult would they be to create? Before SSIS, before the .NET Framework, and before web services, I would probably have answered in terms of decades and many billions of dollars.

A breakdown of what would be needed to make a knowledge brokerage possible consists of

- **Multi/any format data-source connections**: I would be amazed if the majority of the data being harvested weren't held in one of the MS Office products—probably Excel—but Access is also a possibility, as are legacy systems and more-modern databases such as Oracle Database and SQL Server.

- **Rapid processing turnaround**: This would be required under the many data-protection laws around the world to ensure the information supplied is accurate, valid, and pertinent to the individual concerned.

- **Knowledge-consumption services**: Using web services over Secure Sockets Layers (SSL) to provide entry to authentication and to provide for the consumption of new knowledge by other authorized parties.

Would it look, perhaps, something like Figure 1-4?

I believe today, with SSIS as the technology at the heart of such a system, development costs and ultimately total costs would be reduced by at least two orders of magnitude.

One final note: on July 7, 2005, a small group of extremists detonated explosive devices onboard public transport in London, England. Shortly afterward I had an opportunity to be

involved with some of the more data-intensive investigations into the possible leaders of the group and other potentially dangerous individuals. I spoke to an incredibly hard-working but overstretched department within the Security Services that was performing investigations by printing out reams of information and checking it by hand. The data sources included XML documents, Excel, Access, SQL Server, and legacy databases spread throughout the IT infrastructure. These diverse and multiformat data feeds, and the fact they couldn't be integrated together without spending, potentially, millions of dollars, was the reason given for not using the very minimum of software-based assistance to analyze the data.

My first and only recommendation was SQL Server Integration Services.

SSIS leaves you no reason or excuse to hold your data purely relationally. In essence, a database's "data"-bearing capabilities are there out of necessity, almost as if in a legacy system. Only by performing intelligent operations on that data can you mold it into knowledge, conferring meaning and context.

Knowledge is available to us all through SQL Server Integration Services. Take control, make your data work for you, and create your own knowledge brokerage, providing the rock-solid backbone to your business.

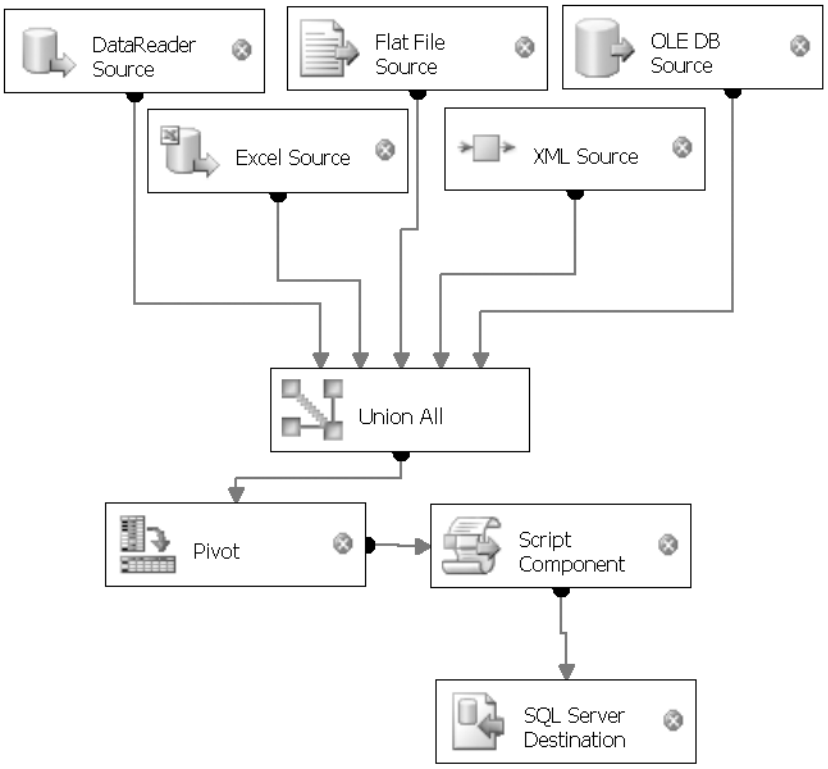

Figure 1-4. *Problem solved*

Summary

I've tried to provide a gentle introduction to SSIS in this chapter by discussing a few of its features, looking at possible applications, and talking about times I've implemented it myself.

During the time I've been developing solutions involving SSIS I have had nothing but the most amazing and complete support from Microsoft, either officially by direct contact or unofficially by reading support articles or SQL Server 2005 Books Online (`http://msdn.microsoft.com/en-us/library/ms130214.aspx`). For those of you who won't have the same level of developer support as I've had, this book should provide a viable alternative in conjunction with other readily available information. I sincerely hope your team environment is developer friendly and you feel able to ask for help, because talking your issues through with someone else, whether SSIS literate or not, gives a different perspective that might prove helpful.

SSIS is simply a great tool. Naturally it has the odd minor flaw and, in places, an occasional bug—but then I've never seen a piece of software that is completely problem free. I know you haven't either.

The business advantage you can give your company by using this technology makes SSIS much more than the sum of its parts. Because you can write your own fully integrated custom components, the advantage you'll gain by using SSIS will be far greater than you considered possible.