SYLLABUS

- Overview of SSIS Development
  - Packages
  - Control Flow vs. Data Flow
  - Package Functionality Extension Objects
- Deploying SSIS Projects
  - Deploying Packages to SQL Server Integration Services Catalog
  - Deployment Models In SSIS
- Planning SSIS Package Execution
- Introduction to Business Intelligence
- Introduction to Reporting
  - Basic Reports in SQL Reporting Services
  - Parameters in Reporting Services
  - Charts in SSRS
- SQL Server Reporting Services Configuration Manager (For Deploy Project in Report Server)
- Introduction to Data Analysis
- Create Cube and Dimensions Using Data Analysis

Table of Contents

1. Overview of SSIS Development

The ability to modify (transform) data before it can be stored at the destination, and the ability to merge the new or modified data appropriately with existing data are not available in the SQL Server Import and Export Wizard; therefore, the wizard is not really suitable for complex data movement scenarios. Nevertheless, one could, for instance, use the wizard to design an SSIS package in minutes, test it, and deploy it so that the data warehouse could be deployed as soon as possible, and then later use the **SQL Server Data Tools (SSDT)** to add any missing functionalities in order to improve the reusability and manageability of the solution.

On the other hand, data warehousing scenarios do not generally count among projects to which rapid solution development is paramount; in the majority of cases, data warehousing projects require a lot of research and planning before it would even be reasonable to do any actual development, with the goal of providing a complete, production-ready data movement solution. By the time the planning phase is completed, an early and quickly developed data movement process would probably already have become obsolete and would have to be modified significantly for production. It is therefore unlikely that the benefits of early deployment would outweigh the need to revise and possibly redesign the data movement process after the design of the data warehouse has actually matured enough for production.

In other words, SSIS development would usually be done "from scratch"—using SSDT instead of the Import and Export Wizard; but this prospect alone does not render the wizard useless.

SSIS uses a special declarative programming language—or rather, a special programming interface—to define the order and conditions of operations' execution. With a strong emphasis on automation, DW maintenance applications differ from the majority of applications in that they do not support user interfaces. Monitoring, inspection, and troubleshooting are provided through auditing and logging capabilities.

Techniques used in SSIS design and development may differ significantly from other programming techniques, appearing especially different to database administrators or developers who are more accustomed to other programming languages (such as Transact-SQL or Microsoft .NET). Most of SSIS development is done graphically—using the mouse, rather than by typing in the commands using the keyboard. A visual approach to design not only allows you to configure the operations, define their order, and determines under what conditions they will be executed, but it also provides a WYSIWYG programming experience. After they have been deployed, SSIS solutions are usually executed automatically—for example, when using SQL Server Agent—but they can also be invoked on demand, by using utilities provided by the platform or through an application programming interface (API).

2. SQL Server Data Tools

Working in SQL Server Data Tools (SSDT), you can perform the following tasks:

- Run the SQL Server Import and Export Wizard to create basic packages that copy data from a source to a destination.
- Create packages that include complex control flow, data flow, event-driven logic, and logging.
- Test and debug packages by using the troubleshooting and monitoring features in SSIS Designer, and the debugging features in SQL Server Data Tools (SSDT).
- Create configurations that update the properties of packages and package objects at run time.
- Create a deployment utility that can install packages and their dependencies on other computers.
- Save copies of packages to the SQL Server msdb database, the SSIS Package Store, and the file system.

3. Deploying SSIS Projects

SSIS enhancements in SQL Server 2012 bring a brand new deployment model for SSIS project deployment. This new deployment model is called Project Deployment Model and unlike the Legacy Deployment Model where each package was a single unit of deployment, this new model creates a deployment packet containing everything (packages and parameters) needed for deployment in a single file with an ispac extension and hence streamlines the deployment process.

Apart from handling the deployment issues, managing the configuration file for each environment for each package was also a pain in the Legacy Deployment model. The new Project Deployment Model includes Project/Package Parameters, Environments, Environment variables and Environment references.

These are the steps we will cover in this tip series:

1. Create an Integration Services Catalog

2. Create a SSIS project with Project Deployment Model

3. Deploy the project to Integration Services Catalog

4. Create Environments, Environment variables

5. Set up environment reference in the deployed project

6. Execute deployed project/package using the environment for example either for TEST or PROD 7. Analyze the operations performed on the Integration Services Catalog

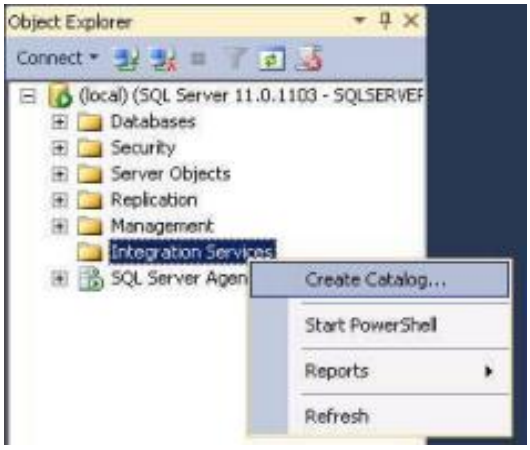
8. Validate the deployed project or package

9. Redeploy the project to Integration Services Catalog

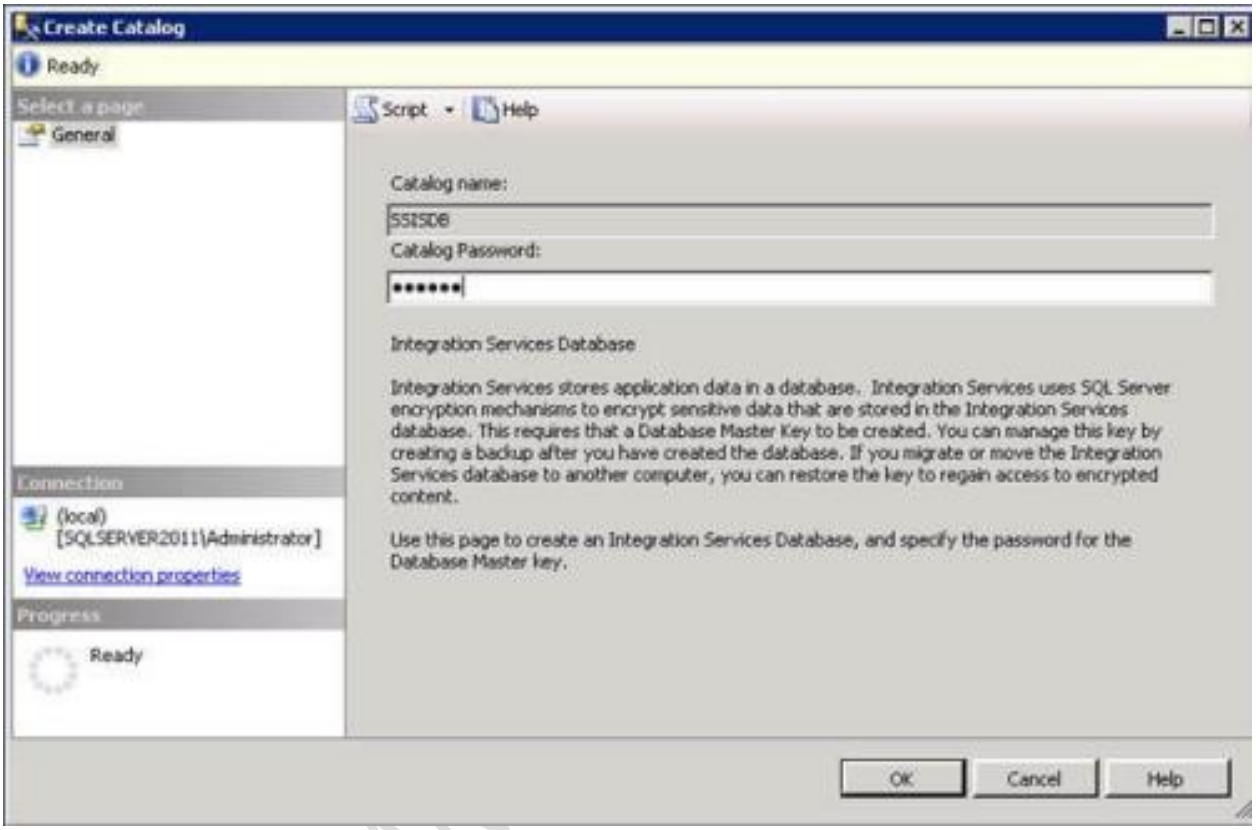
10. Analyze deployed project versions and restored to desired one

4. Creating Integration Services Catalog

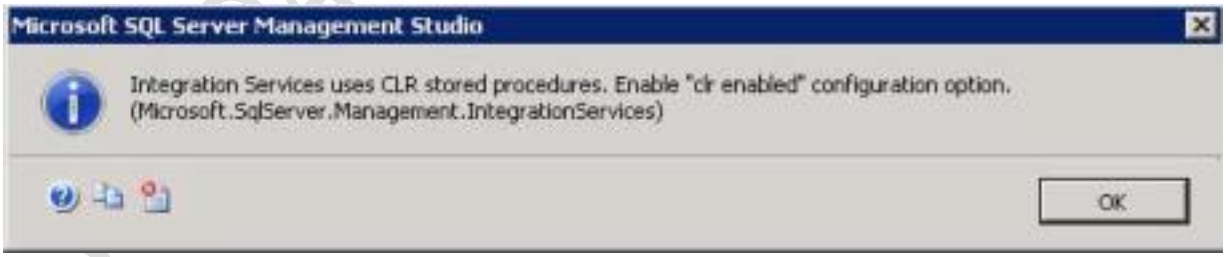
First of all we need to create an Integration Services catalog on the SQL Server instance (note: we can create only one Integration Service catalog on the SQL Server instance, though a catalog may contain many folders and inside each folder many projects). To create an Integration Services catalog connect to a SQL Server instance in SSMS (SQL Server Management Studio) and right click on the Integration Services node under the connected server in the Object Explorer and click on **Create Catalog** as shown below:



By default the catalog name appears as SSISDB and cannot be changed. You need to provide a strong password for the SSISDB Integration Services catalog you are creating which is used for the database master key. Let me tell you why you need to provide this; actually the Integration Services catalog stores application data in a SQL Server database and uses SQL Server encryption to encrypt sensitive data. For encryption, it needs to have a database master key and for that only you need to provide a password here. It's recommended to back up the database master key after the Integration Services catalog creation.



The Integration Services catalog uses CLR based stored procedures and by default CLR is not enabled for a SQL Server instance, so you need to enable it before creating an Integration Services catalog.



To enable CLR on a SQL Server instance you can execute the script below. Once you have enabled CLR you can go ahead and create the Integration Service catalog as discussed above.

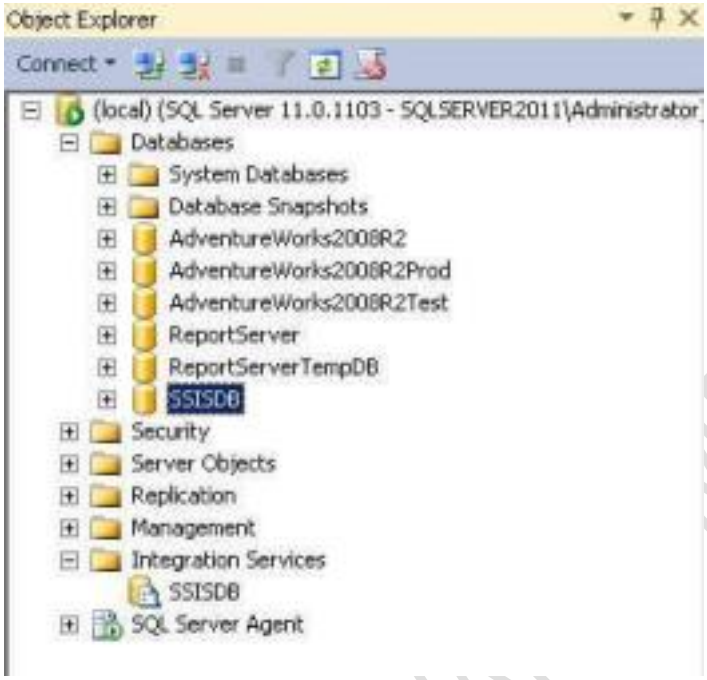
```
--Script #1 - Enabling CLR on the SQL Server Instance
sp_configure 'show advanced options', 1;

GO
RECONFIGURE;
```

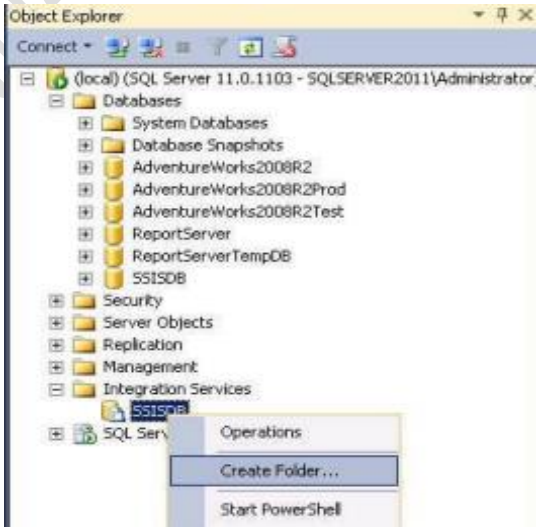
```
GO
sp_configure 'clr enabled', 1;

GO
RECONFIGURE;
GO
```

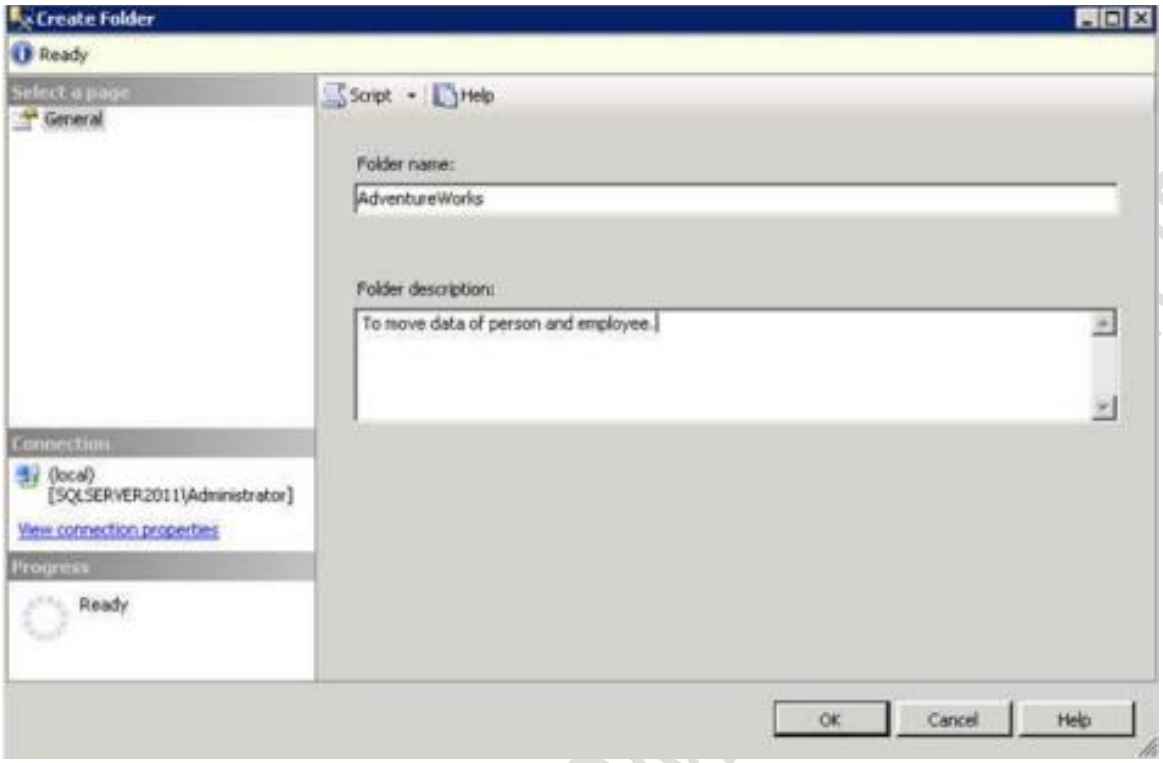
Once an Integration Services Catalog gets created you can verify it in SSMS as shown below. As I mentioned before, an Integration Services Catalog stores application data in a SQL Server database and hence a database has been created with the same name as the Integration Services Catalog. If you browse through the database you will notice there are several tables which store the data, several views built on top of these tables and several stored procedure to access and manage this data.



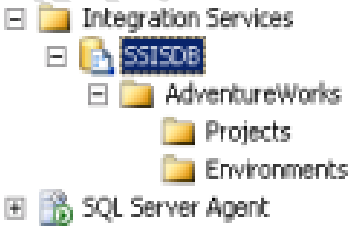
Even though you can have a single Integration Services Catalog on each instance, each Integration Services Catalog can have several projects deployed to it. Let me first create a folder (AdventureWorks) for my project which I will be deploying next.



To create a folder, right click on the Integration Services Catalog name under Integration Services node in SSMS as shown above and click on the **Create Folder** menu item. This will launch the Create Folder dialog box as shown below; specify a name for the folder to create and a folder description:



Each folder that you create inside an Integration Services catalog will have two subfolders inside it by default as shown below. The **Projects** folder is a place where you will be deploying your SSIS project and the **Environments** folder is a place where you will be creating multiple environments like development, test, PPE (Pre-Production environment), Production, etc... I will discuss these folders more in a later tip in this series.

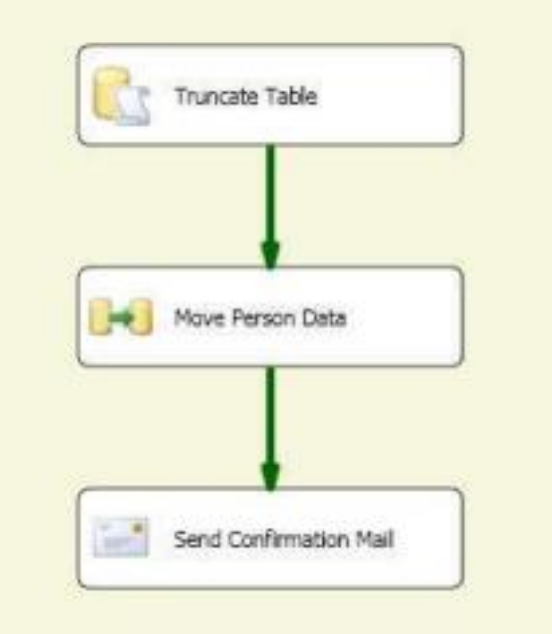


5. Creating a SSIS project with Project Deployment Model

The way we develop SSIS project/package in SQL Server 2012 remains the same as what we have been doing, so I am not going to talk in detail about SSIS project/package creation (to learn basics of SSIS you can refer to this tutorial) but rather directly jump into the development for the example. One noticeable difference in the SSIS project created in SQL Server 2012 is that by default the SSIS project will be created in Project Deployment mode, but if you need to you can change it by right clicking on the project in Solution Explorer and clicking on "Convert to Legacy Deployment Model".

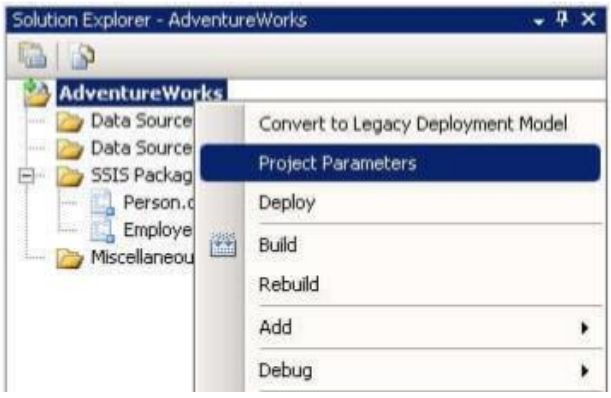
A SSIS project in Project Deployment model creates a deployment packet (with *.ispac extension) that contains everything (all packages/parameters) needed for deployment unlike the Legacy Deployment mode in which each SSIS package is separate unit of deployment.

As per our requirement we need to have one SSIS project with a couple of packages. I will keep the design of the package very simple, it first truncates the target table, moves the data from the source to the target table and sends the confirmation email at the end.

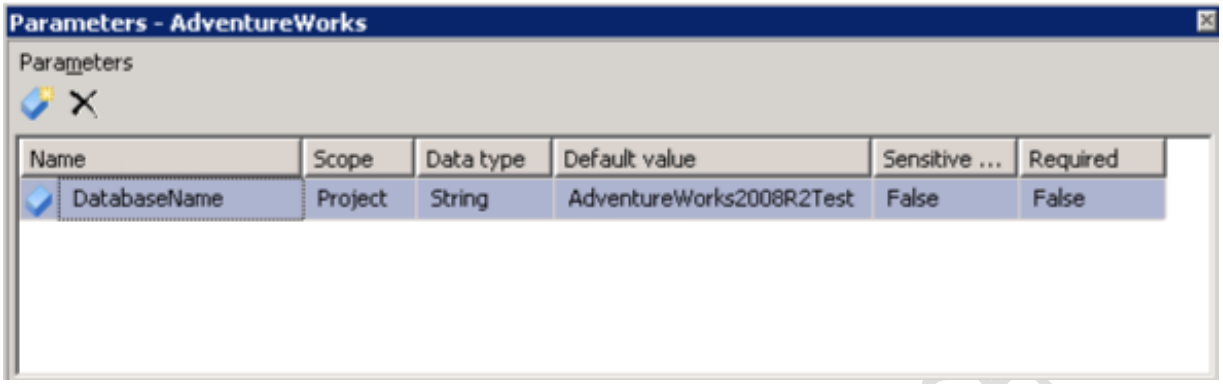


Now what I want is to move data to a table in a different database depending on the environment. For example, if the package is executed with the TEST environment reference then data should be moved to AdventuresWorks2008R2Test database and if the package is executed with the PROD environment reference then data should be moved to AdventuresWorks2008R2Prod database. So let me create one

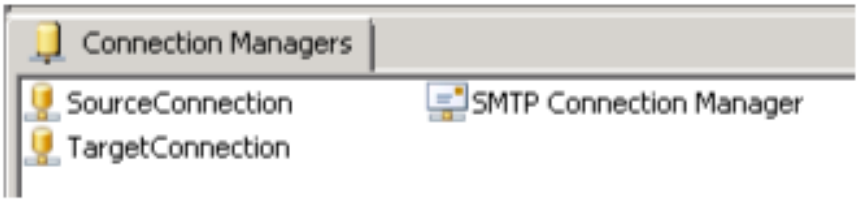
Project Parameter to hold the name of the database which will be passed from an environment variable and used in the Target Connection's connection string to connect to the target database. To create a project parameter, right click on the project in the Solution Explorer and click on the Project Parameters menu item as shown below:



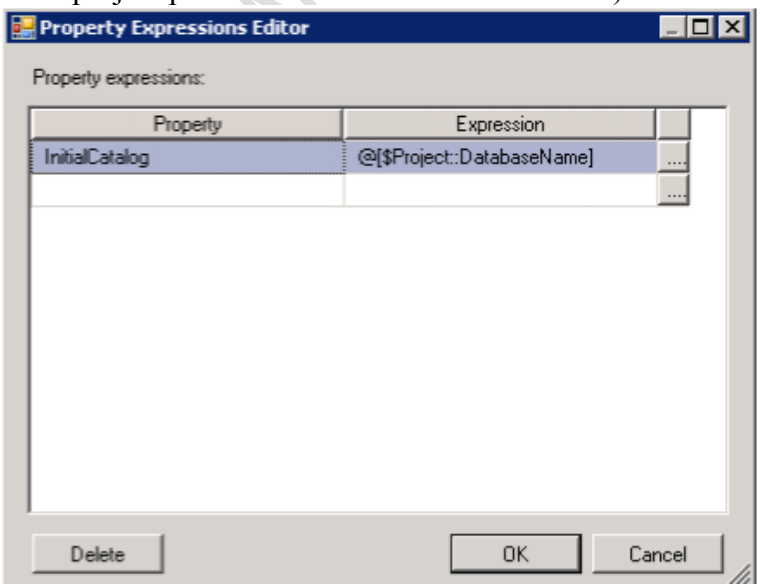
On the Project Parameters window, click on the New Parameter icon in the upper left and specify the name of the project parameter. In this case I want the parameter to hold the database name hence I have specified DatabaseName name for the parameter and the default value specified as "AdventureWorks2008R2Test". This means if the package is executed in the designer, data will move to AdventureWorks2008R2Test database.



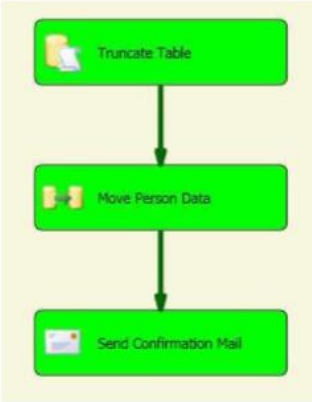
The package also contains two connection managers to connect to the databases. Source Connection manager connects to the source to pull data from and will remain constant whereas Target Connection connection manager gets the database name from a project parameter and varies from environment to environment. Apart from these two connection managers to connect to the databases, I have one SMTP connection manager which is used to send confirmation email as shown in the package design.



To make a connection manager to dynamically use the database name, you need to configure the Expression property of the Target Connection manager and specify the value of Initial Catalog property to come from an expression (in this case project parameter which we created above) as shown below:

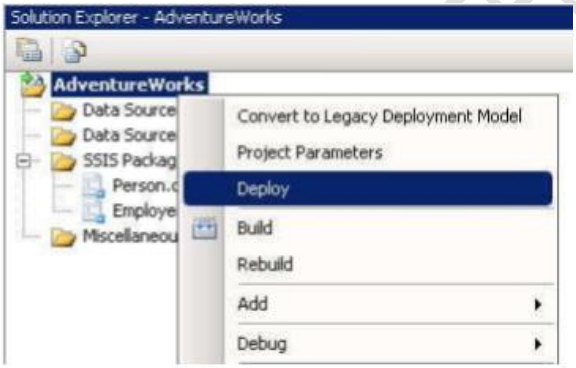


As we have specified a default value for the project parameter, if you execute the package it will execute successfully and will move the data to AdventureWorks2008R2Test database (the default value specified for the project parameter).

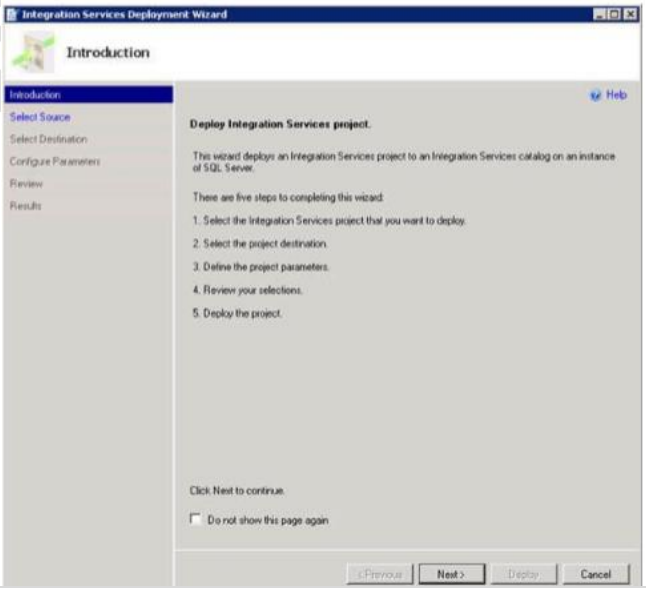


Deploying the project to Integration Services Catalog...

Now that we are done with project and package development, we need to deploy the project to the Integration Services catalog we created above. To deploy the project, right click on the project in the Solution Explorer and click on the Deploy menu as shown below:

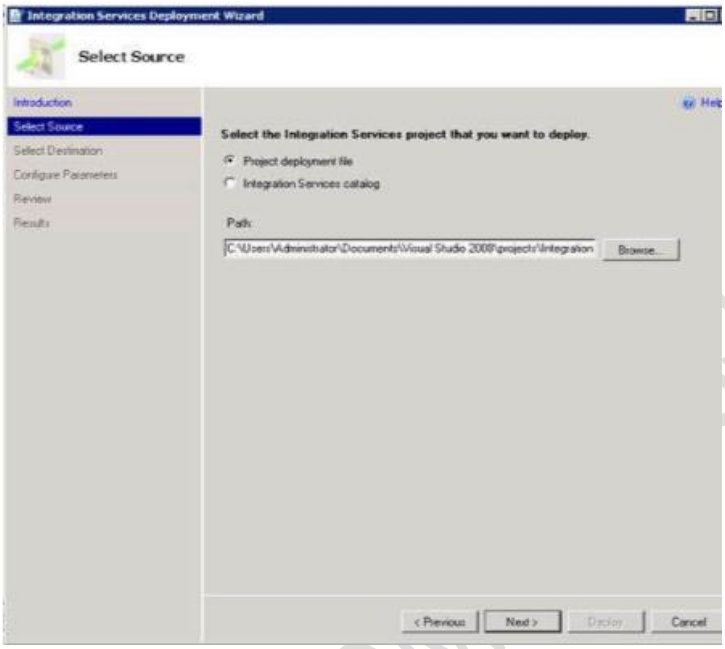


Clicking on the Deploy menu item will launch the Integration Services Deployment wizard and the first screen of the wizard is a welcome screen as shown below. You can choose not to show this screen next time by

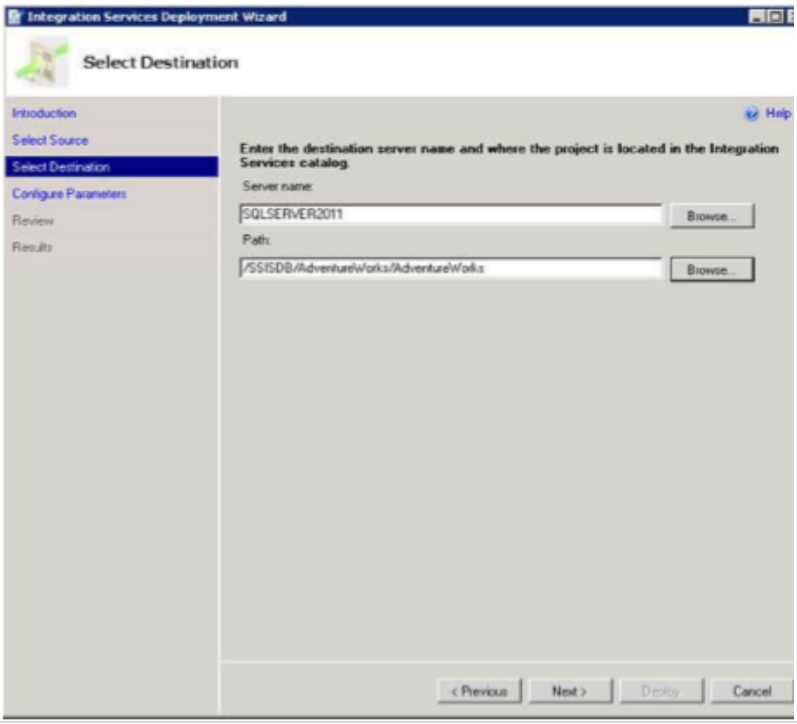


clicking on the checkbox on bottom. Click on Next button to move ahead:

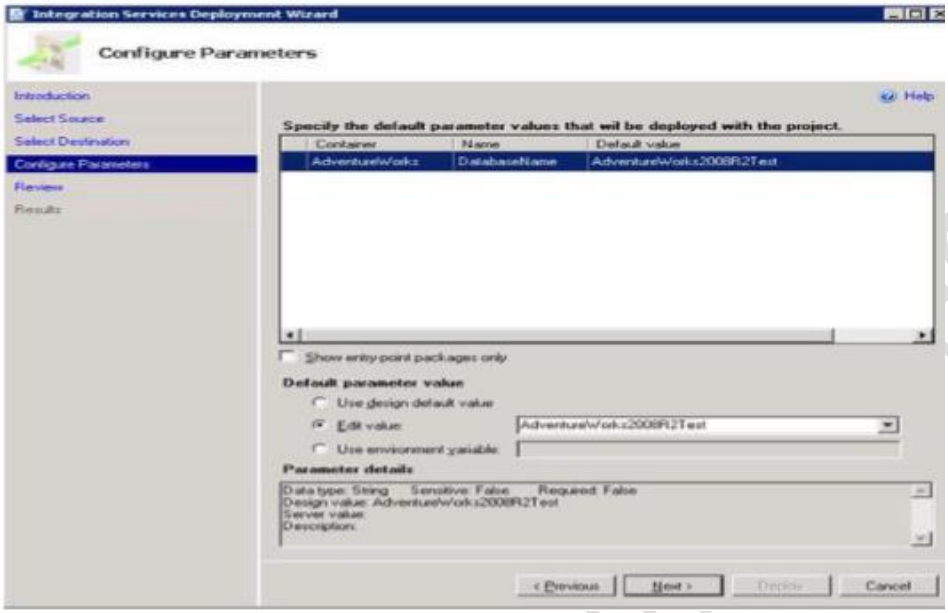
The second screen of the wizard is the place where you actually specify the location to deploy from; you can either choose the deployment packet (*.ispac) file or choose an already deployed package from the Integration Services catalog as the source for the deployment. Since I want to deploy from the SSIS project, I have specified the *.ispac deployment packet name:



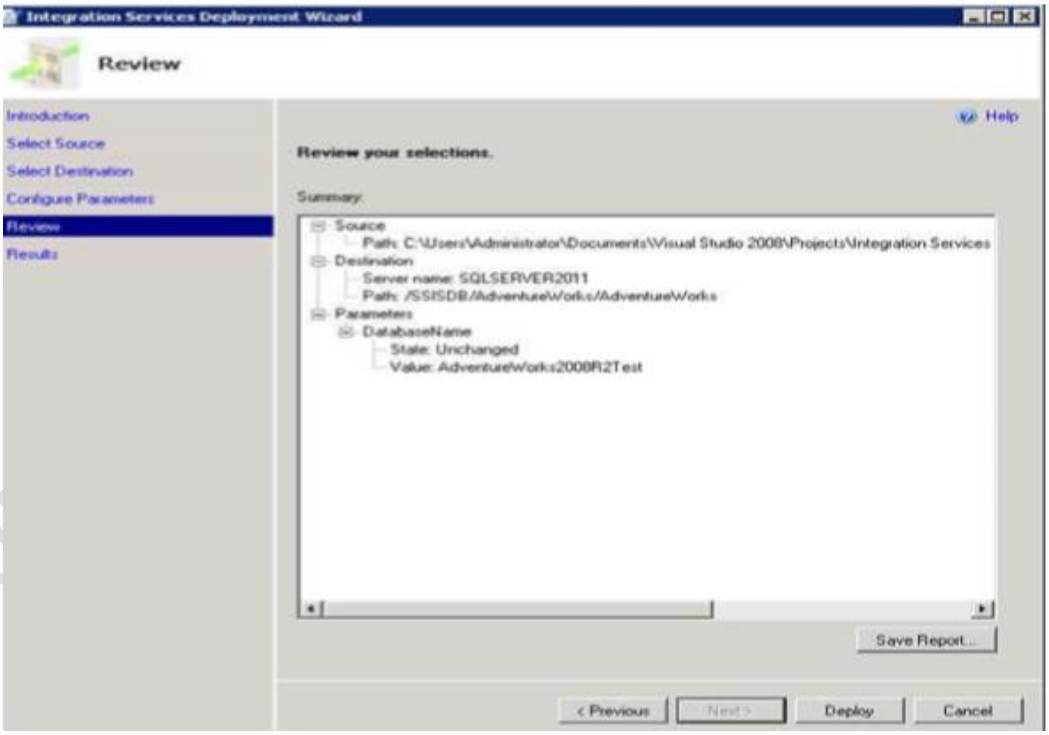
The third screen of the wizard lets you specify the destination where you want to deploy the project. You need to choose the name of the server where you have created the Integration Services catalog and the folder in the catalog where you want to deploy the project. I will use the folder I created above to deploy the project as shown below:



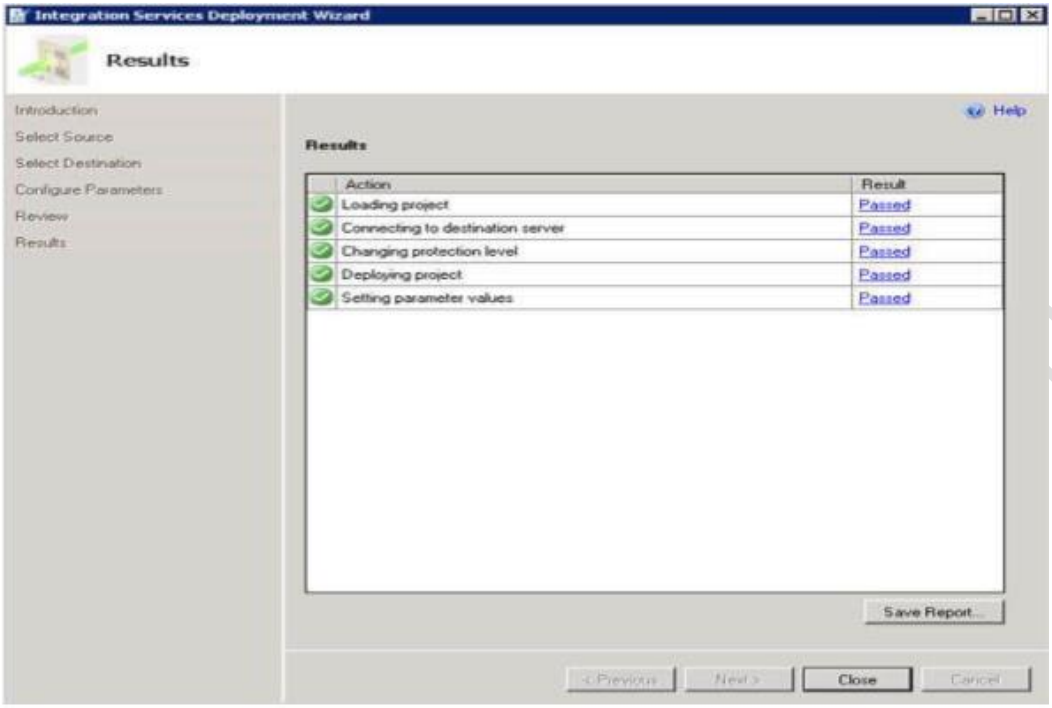
As I mentioned before, a parameter can have a server default value, the next screen is the place where you define the server default values for all the parameters of the project. Here you can either use the same design default value, specify a new value or choose the value to come from a variable as shown below:



The next screen of the wizard is a review screen where you review your selections, you can click on the Deploy button to start the deployment of the project as shown below:



The final screen of the wizard shows the deployment progress and deployment status as you can see below. If there are any failures they will be marked red and you can click on the Result column to see the reason for the failure.



Since our deployed was successful as shown above, we can connect to the Integration Services Catalog using SSMS and verify the deployment as shown below:

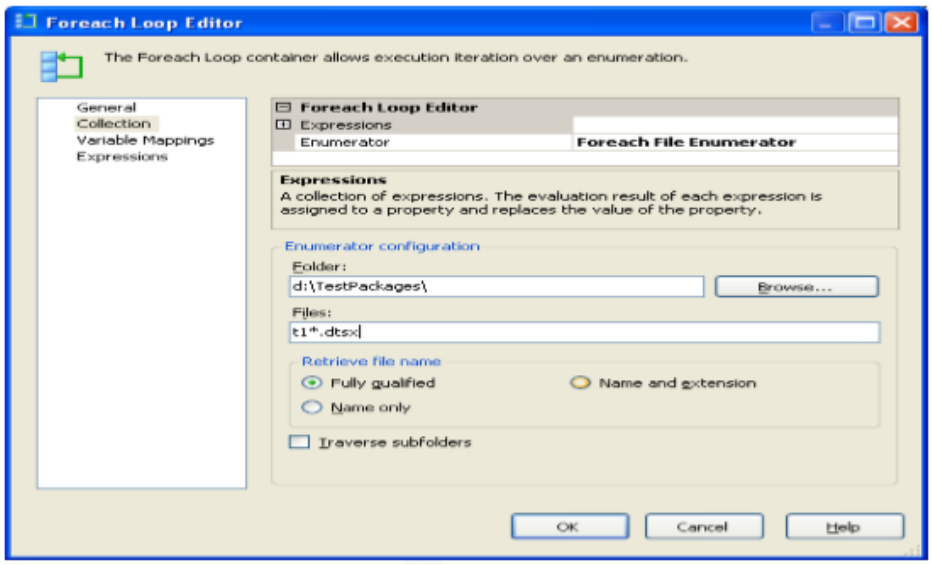
Planning SSIS Package Execution

For this we create a package that uses a for each loop task to call an execute package task for all packages in the folder.

The first task is to create an SSIS package with a Foreach Loop container that will loop round the packages in the folder, setting a variable called "PackageToRun" to the file name for each package.

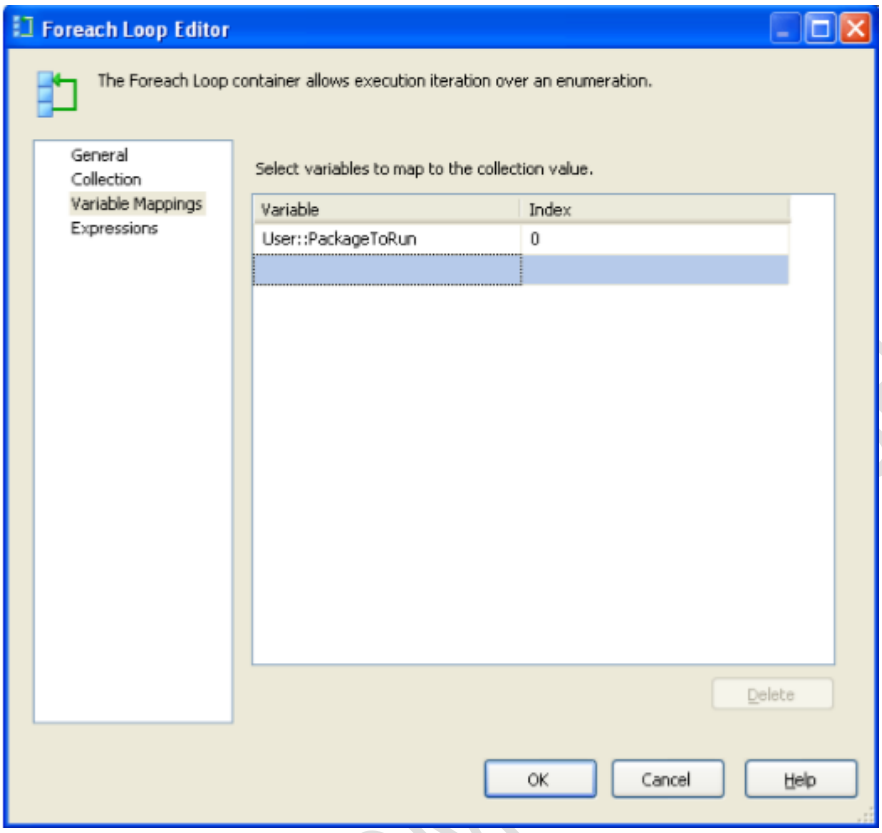
1. Load Business Intelligence Development Studio and start a SSIS project.
2. Create a new package.
3. Add a Foreach Loop container to the package.
4. Right-click on the Foreach Loop container and select Edit.
5. Click on Collection.

6. Set the Enumerator to Foreach File Enumerator.
7. In the Enumerator configuration: a. Set Folder to "d:\TestPackages\" b. Set Files to "t1*.dtsx" c. Under Retrieve file name select fully qualified.
8. Click OK.



1. Click on Variable Mappings.
2. Click on the Variable drop-down list and select New Variable.
3. Set Name to Package Torun.

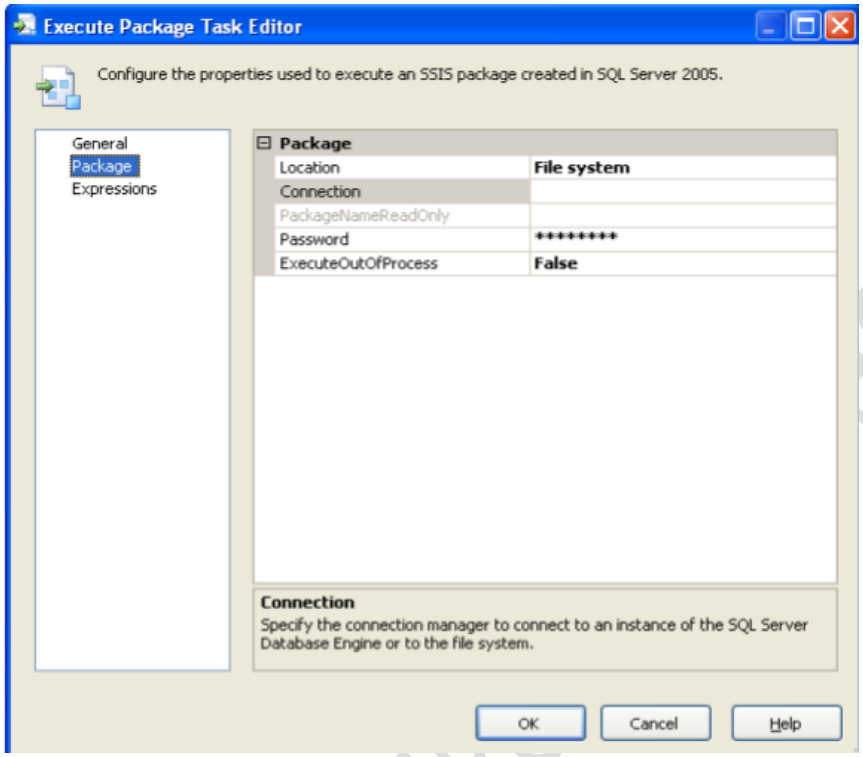
4. Click OK.



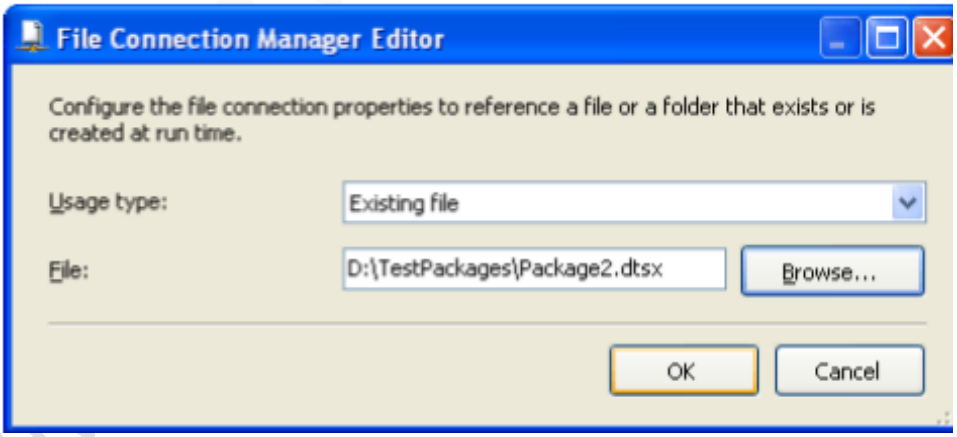
Next, we need to add the Execute Package task to the Foreach Loop container so that this task will be executed for each package that we wish to run. We then set the variable value to be the name of the package to be executed by the ExecutePackage task.

1. Drag an Execute Package task into the Foreach Loop container.
2. Right-click on the Execute Package task and select Edit.
3. Select Package.

4. Set Location to File system



1. Click on the Connection drop-down list and select <New connection…>.
2. Set the File to an existing package.



1. Click OK to save the connection.
2. Click OK to complete the Execute Package task configuration.

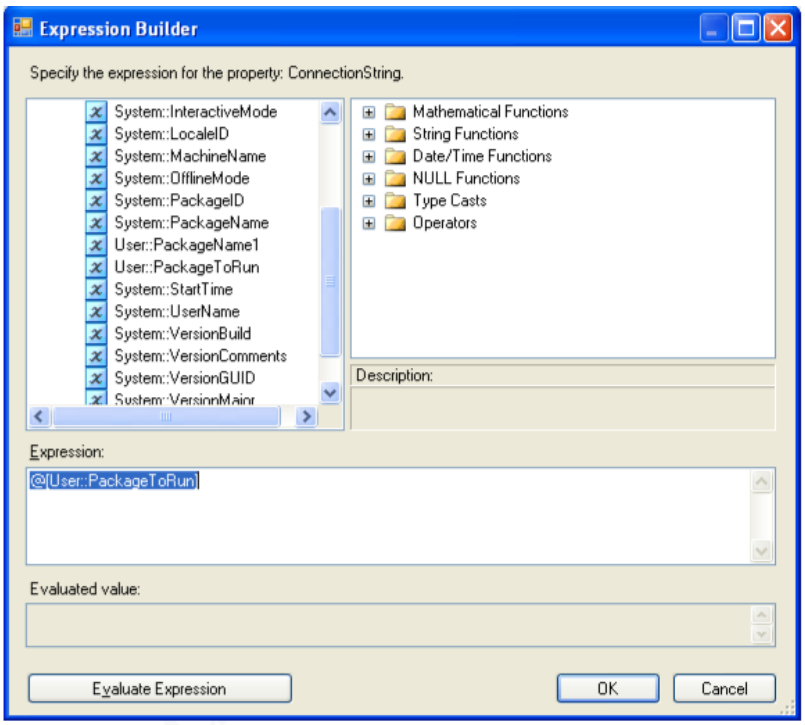
Finally, we configure the connection to use the variable package name:

1. Right-click on the connection and select Properties.
2. In the Properties window change the name to PackageToRunConnection.

Note: this will automatically update the connection name in the Execute Package Task.

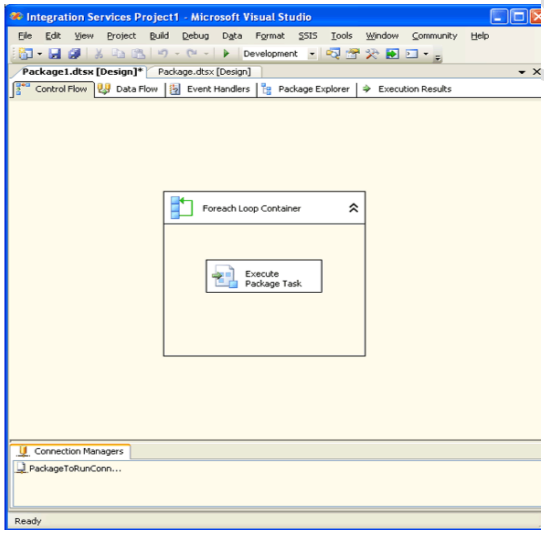
1. Select Expressions and add a new expression.

2. In the property drop-down list select ConnectionString.
3. Click on the Expression Editor Button.
4. From the Variables tree drag @ [User::PackageToRun] into the Expression window.



1. Click OK twice, to save the expression.

The package should now look like this:



If you now run the package, it will execute each package in the folder with the correct filename mask.

6. Introduction to Business Intelligence

Business Intelligence (BI) is a collection of activities to get an understanding and insights about a business by performing various types of analysis on the company data as well as on external data from third parties to help make strategic, tactical, and operational business decisions and take the necessary actions to improve business

performance. The key words here are "improve business performance." The purpose of any BI activity is to improve business performance. Some of the most popular examples of BI implementations are analyzing customer profitability, studying product profitability, evaluating sales figures across different products and regions, exploring accounts profitability, examining supplier performance, and discovering customer risk patterns.

It is important to bear in mind that business intelligence activities are not limited to the data in the data warehouse. Business intelligence applications can query the ODS or ERP or any other systems in the enterprise to perform the necessary analysis for improving business performance. But in this chapter, I'll discuss only BI applications that query the data warehouse.

You can classify the various BI applications into six categories: reporting applications, analytic applications, and data mining applications, dashboards, alerts, and portal. Reporting applications query the data warehouse and present the data in static tabular or pivot format. Analytic applications query the data warehouse repeatedly and interactively and present the data in flexible formats that users can slice and dice. Data mining applications explore the data warehouse to find patterns and relationships that describe the data. Reporting applications are usually used to perform lightweight analysis. Analytic applications are used to perform deeper analysis. Data mining applications are used for pattern finding. Dashboards are a category of business intelligence applications that give a quick high level summary of business performance in graphical gadgets, typically gauges, charts, indicators, and color-coded maps. By clicking these gadgets, we can drill down to lower-level details. Alerts are notifications to the users when certain events or conditions happen. A BI portal is an application that functions as a gateway to access and manage business intelligence reports, analytics, data mining, and dashboard applications as well as alert subscriptions.

7. Introduction to BI Reporting

Reports query the data warehouse and present the data in tabular format or pivot format. We can also produce various types of charts. We can add parameter on the reports to make them dynamic. In a data warehousing context, a report retrieves data from the data warehouse and presents it to the users on the screen or on paper. Users also can subscribe to these reports so that the reports can be sent to the users automatically by email at certain times (daily or weekly, for example).

The main advantage of using reports in BI is their simplicity. Reports are simple to create, simple to manage, and simple to use. We usually use reports (instead of analytics or data mining applications) in BI when the presentation format requirements are fairly simple and static. The disadvantages of using reports are that they are not flexible or interactive. If users want to swap a piece of data with another piece or want to view the data at a higher or lower level, we need to redesign the report. By comparison, the other types of business intelligence applications such as analytic applications are more flexible. We don't have to redesign the application if the user wants to present another piece of information. We just "plug" the analytic application on top of some cubes, and the user can explore all the data in those cubes.

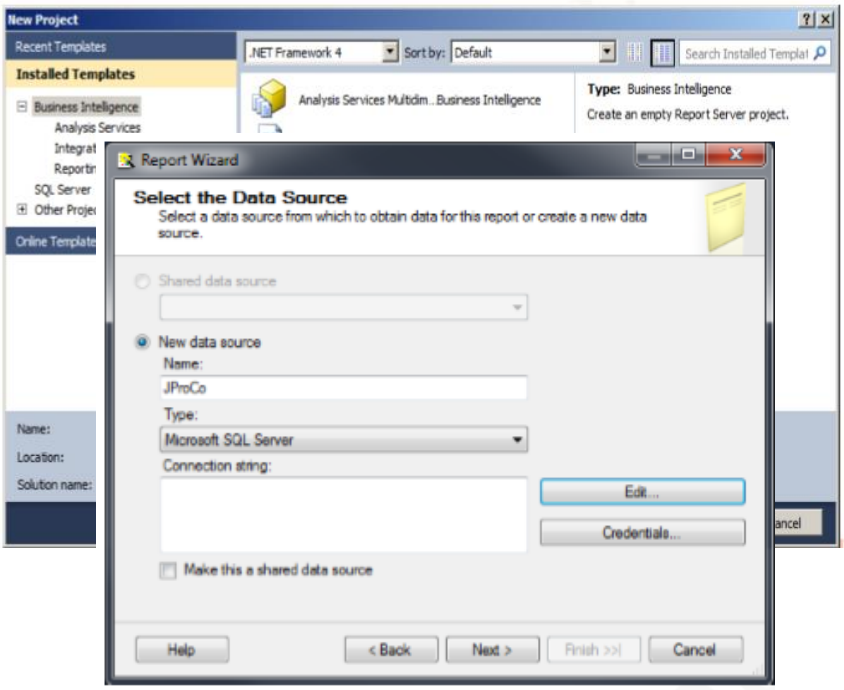
With SSRS we can do charting (line chart, bar chart, pie chart, and so on) or present the data in simple tables or pivot table format. We can group, sort, and filter the data. We can drive the content of the report using parameters and make the report a little bit dynamic. Users can also change the sort order of the report at run time. Users can subscribe to a report to receive it via e-mail automatically at certain intervals. The report can be attached to the e-mail, or the e-mail can just contain the link to the report. We can also create a subreport within a report. We can store or cache the report output for quick retrieval later. We can add a link to another report by passing certain parameters to tailor the content. BI reports are not limited to just the data warehouse.

BI reports can query other data sources within the enterprise. Most relational databases (SQL Server, Oracle, Teradata, DB/2, my SQL, Informix, Sybase, and many others) can be accessed using ADO.NET, OLE DB, Open Database Connectivity (ODBC), and Java Database Connectivity (JDBC). Instead of using ODBC to access SQL Server, we can also use the SQL Native Client driver. Most multidimensional databases can be accessed using XML for Analysis (XMLA) and ADOMD.NET. We can access, for example, data stored in the ERP systems (production, HR, sales, and purchasing information), the call center application (customer applications, complaints, and new contacts), the ODS (integrated master data, reference codes, account and policy information, up to-date prices, and currency conversion rates), or even Windows Active Directory (security information and usernames). Using reports we can combine the stored results of a saved query and the fresh output of an online query. We can combine the data from ERP systems, the ODS, and the data warehouse in a single report or in a set of reports. BI reports are usually managed and published in one central place, such as an intranet.

Using SSRS we can secure the reports, allowing only the people who should be able to access the reports and keeping everybody else out. We can set the security access so that different people or departments can administer the content of different reports. We can also integrate the report security with Windows Active Directory so the users don't have to login again to access the reports.

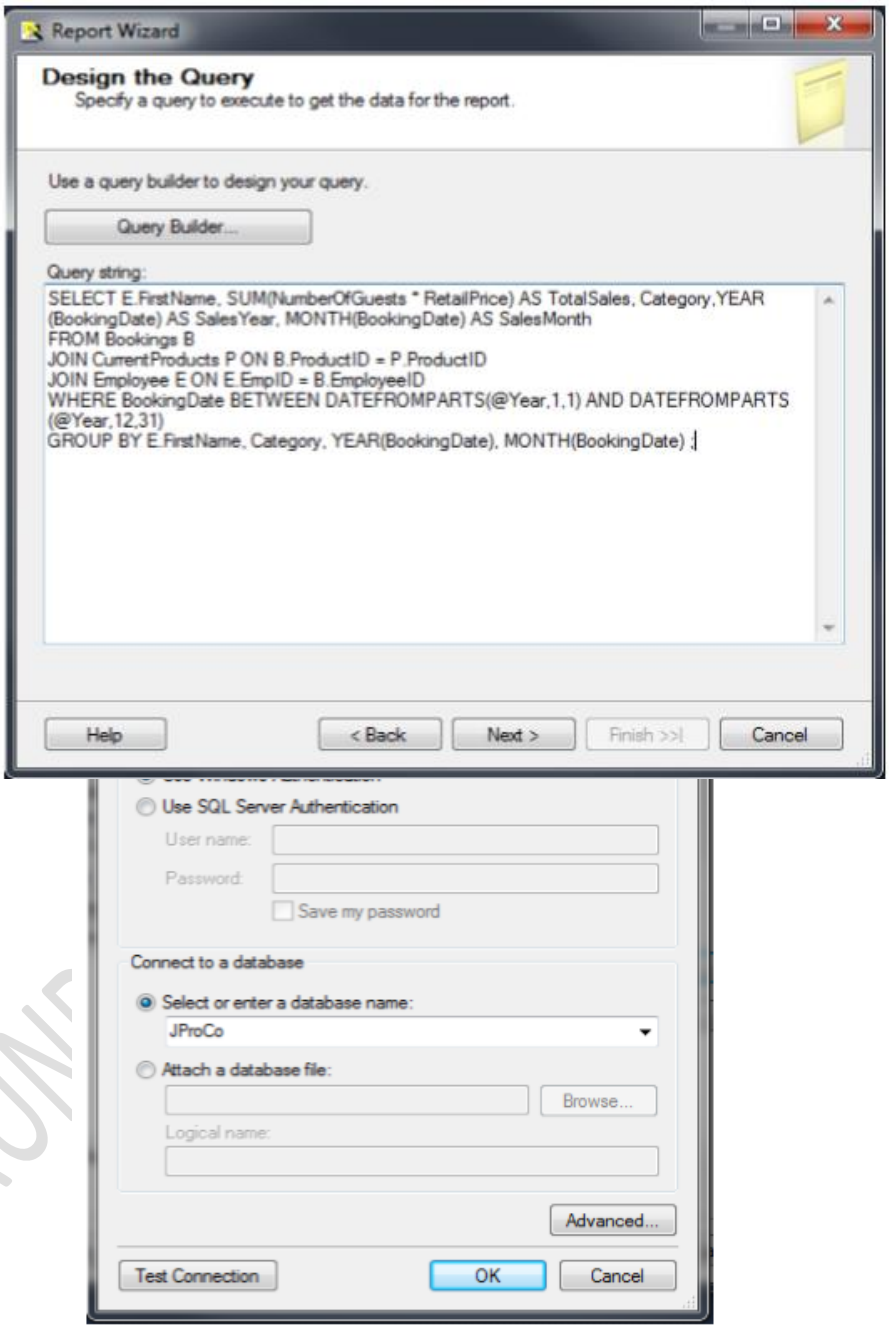
8. Create a Report with the Report Wizard

Let's get started with your first report! Launch SQL Server Data Tools (SSDT) from the Start menu under SQL Server 2012. Once SSDT is running, click New Project to launch the New Project dialog box. On the left side of the screen expand Business Intelligence and select Reporting Services. Configure the properties as shown in. Be sure to select Report Server Project Wizard as the type of report and to save the project in the Project folder.

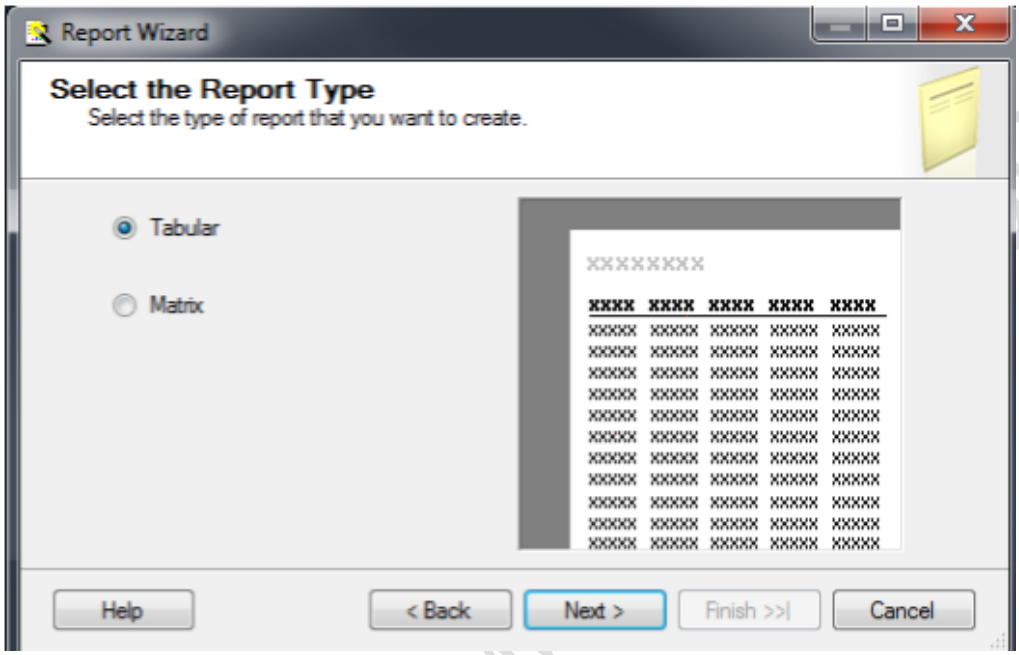


Click OK and wait for the Report Wizard to launch. Click next on the Welcome screen. On the Select the Data Source screen, make sure that "New data" source is selected. Type "Source" as the data source name. Make sure that Microsoft SQL Server is selected in the Type dropdown.

Click Edit to configure the connection string on the Connection Properties dialog box. If your SQL Server database server is installed on your local computer, type in local host for the Server name and select the "Source" database from the Select or enter a database name dropdown.
Click OK to dismiss the Connection Properties dialog box. Check make this a shared data source and click next.

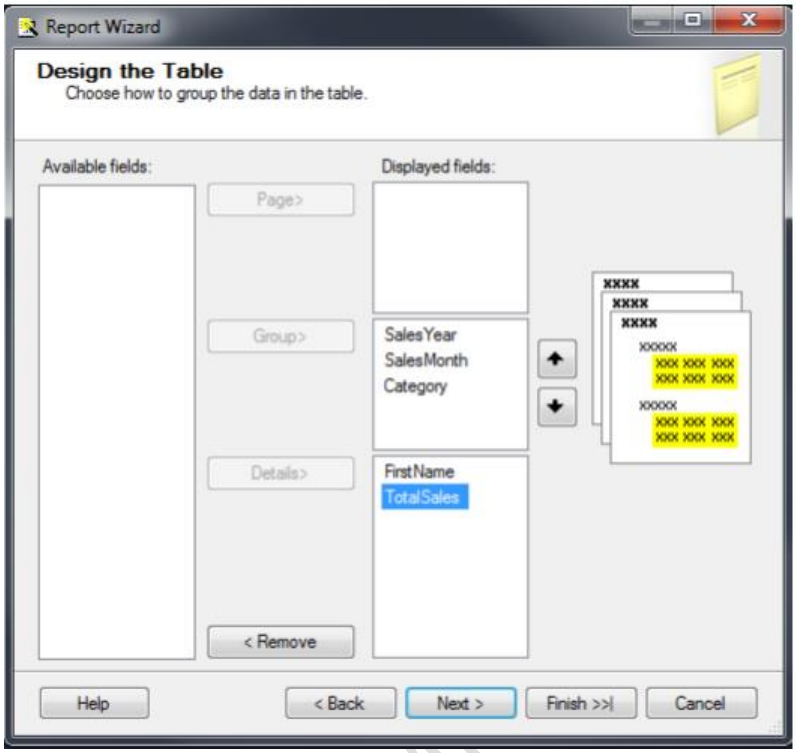


On the Design the Query screen, you can use the query builder to build a query if you wish. Since this post is not meant to teach you T-SQL queries, you will copy all queries from files that have been provided for you. In the "project" folder open the sales by employee.sql file. Copy and paste the code from the file into the Query string Text Box. Click Next.

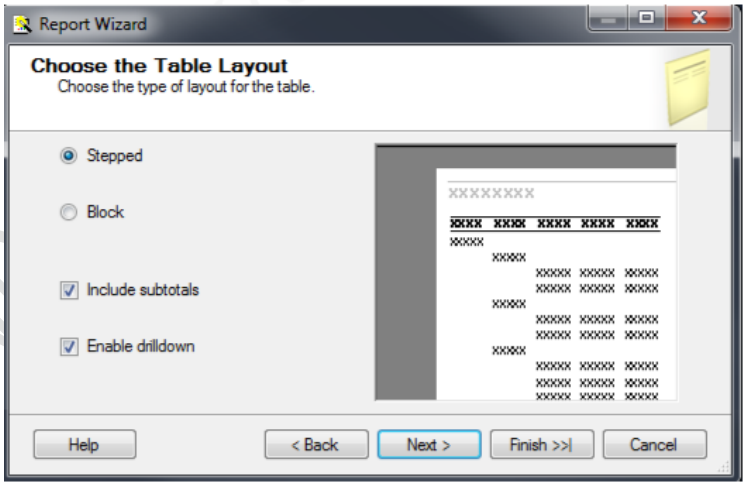


On the Select the Report Type screen, choose Tabular and click next. On the Design the Table screen, you have to figure out the groupings of the report. How do you do this? Well, you often need to know a bit about the data and report requirements. I often draw the report out on paper first to help me determine the groups.
In the case of this report, I could group the data several ways. Do I want to see the data grouped by Year and Month? Do I want to see the data grouped by Employee or Category? The only thing I know for sure about this ahead of time is that the TotalSales goes in the Details section. Let's assume that the CIO asked to see the data grouped first by Year and Month, then by Category.

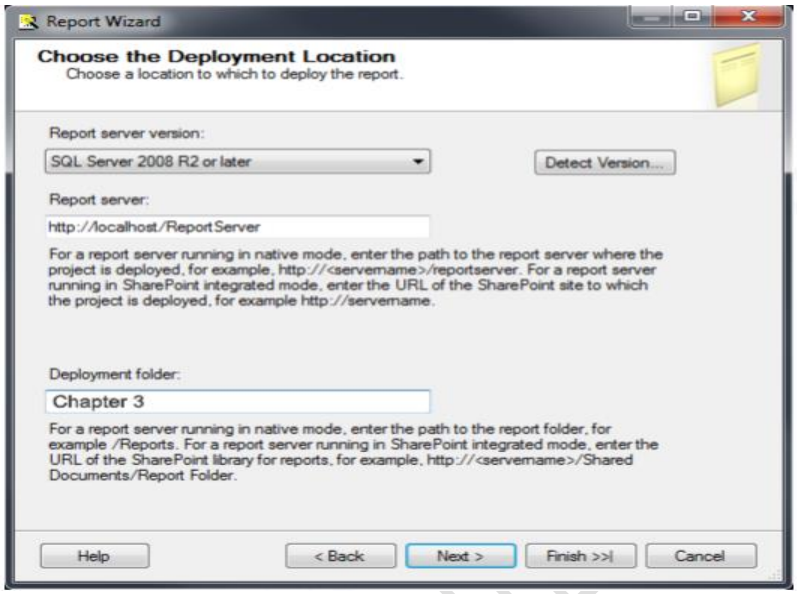
Let's move the fields to the right-hand side. This is done by selecting Page > Group or Details >, as shown in, and click Next.



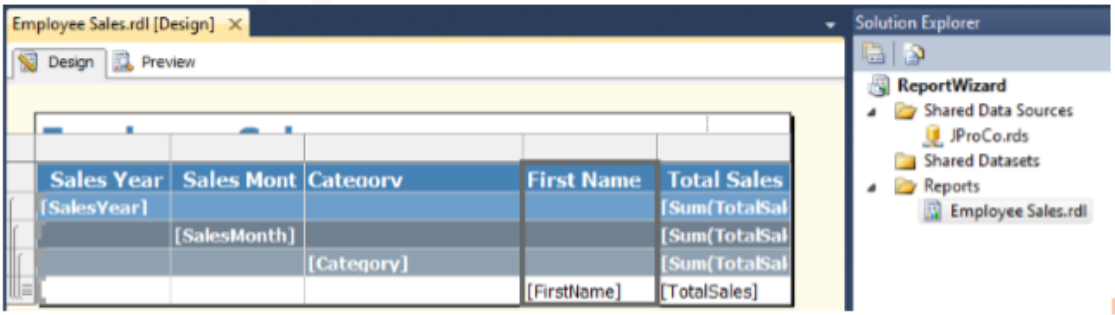
On the Choose the Table Layout screen, select Stepped and check Include subtotals and Enable drilldown, as shown in.



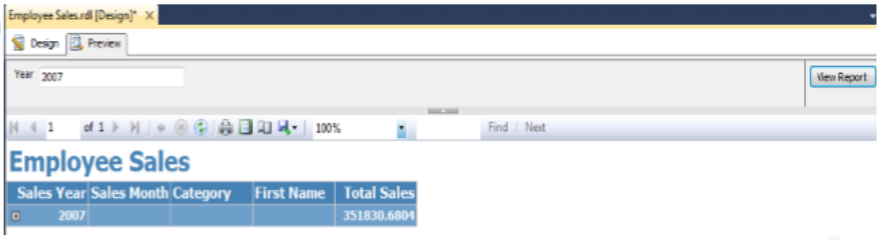
On the Choose the Style screen, choose any color scheme you wish (unlike the Model T) and click Next. I chose the default, Slate. On the Choose the Deployment Location screen, change the Deployment folder to "project" folder and click next.



At the Completing the Wizard screen, name your report Employee Sales and click Finish. After clicking Finish, the report and a shared data source will appear in the Solution Explorer and the report will also be visible in Design view.



Click the Preview tab at the top. This report expects the user to supply a year which the report will then use as a filter. Type in a year between 2006 and 2013 and click View Report.



Click the plus sign next to the Sales Year to expand the report to see the months, then expand again to see the categories and finally the details. You now have the assembly line report completed, and you probably already have some ideas on how to improve the report.

9. Introduction to BI Data Analysis

Henry Morris coined the term analytic applications in 1997. He defined analytic applications as applications that "provide guidance in the decision-making process by accessing time-based data from multiple sources."1 Users use analytic applications to access a dimensional data warehouse interactively. Analytic applications, or analytics for short, are also known as OLAP applications. Users can slice and dice the data, drilling up and drilling down.

Like to clarify the terminology that people use in analytics (such as slicing, dicing, drilling up, and drilling down) to make sure we have a common understanding. Slicing is the process of retrieving a block of data from a cube by filtering on one dimension, as shown in below given figure. Dicing is the process of retrieving a block of data from a cube by filtering on all dimensions, as shown in next figure.

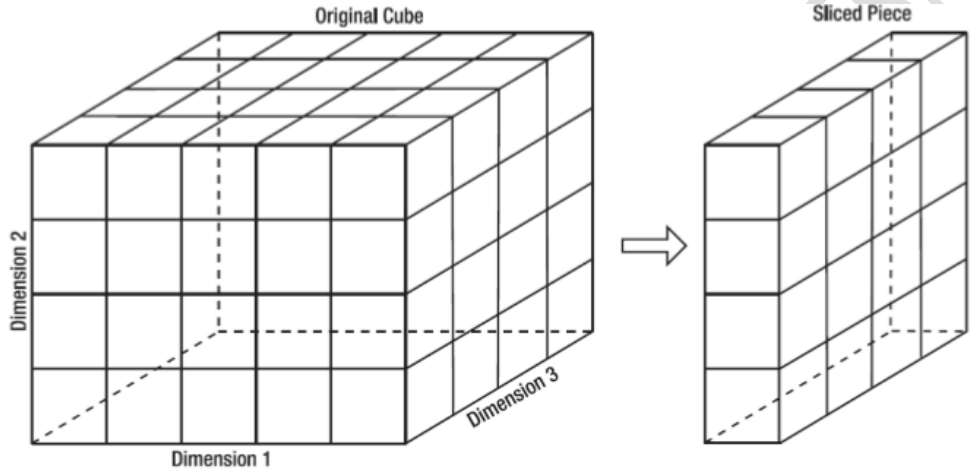


Figure: Slicing

In above figure, we are filtering only on dimension 1; in other words, dimension 1 = [values]. We do not constrain or filter dimensions 2 and 3.

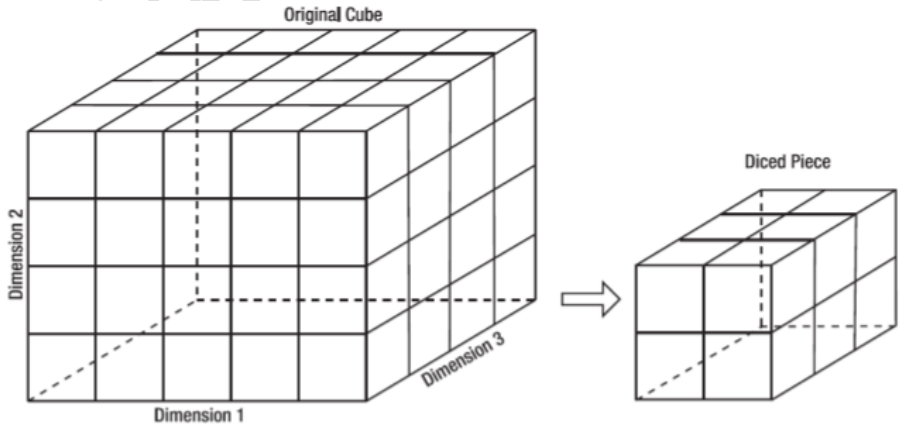
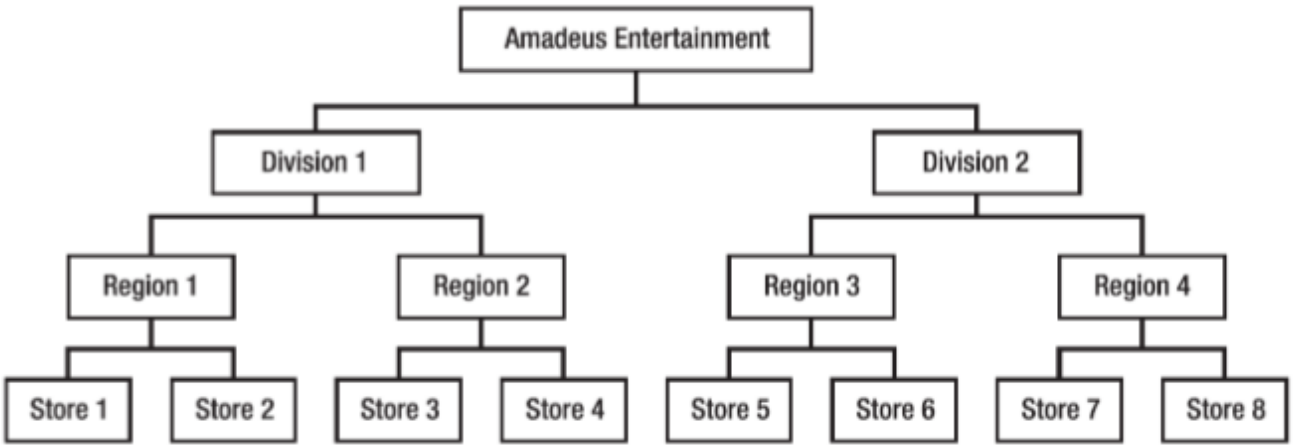


Figure : Dicing

In above figure, we filter on all three dimensions; in other words, dimension 1 = [values], dimension 2 = [values], and dimension 3 = [values].

Now let's discuss drilling up and drilling down. Figure given below displays the store hierarchy of Amadeus Entertainment. On the bottom level we have stores. Moving up the hierarchy we have regions, then divisions, and finally the whole Amadeus Entertainment group.



Drilling up means presenting the data at a higher level on the dimension hierarchy (from detail level to summary level), such as going from the store level to the region level. Drilling down means presenting the data at a lower level on the dimension hierarchy (from summary level to detail level), such as going from the region level to the store level.

To illustrate the process of drilling up, please look at the subscription revenue of Amadeus Entertainment stores for a particular week in below Table.

Store	Subscription Revenue
Store 1	72,663
Store 2	61,899
Store 3	36,409
Store 4	61,994
Store 5	75,786
Store 6	34,049
Store 7	34,937
Store 8	41,012

Table : Subscription Revenue for a Particular Week at Store Level

When we drill up from store level to region level, we will get the subscription revenue for each region, as displayed in below Table.

Region	Subscription Revenue
Region 1	134,562
Region 2	98,403
Region 3	109,835
Region 4	75,949

Table : Subscription Revenue for a Particular Week at Region Level

We can then drill up further to division level and to group level. Drilling down is the reverse, going from a higher level to a lower level. When drilling down, we get more detail information.

Analytics are what most people mean when they say BI applications. Depending on which industry you are in, you can use analytic applications to evaluate business performance, analyze product and customer profitability, and study how to lower inventory costs. Some analytic applications have their own multidimensional database management system (MDBMS) such as SQL Server Analysis Services, Cognos 8 BI Analysis, Hyperion System 9 BI Essbase Analytics, and SAP Business Warehouse. Some analytic applications can use other vendors' MDBMSs; for example, Business Objects OLAP Intelligence can read Microsoft Analysis Services, Hyperion Essbase, SAP BW, and DB2 OLAP cubes, 2 while ProClarity reads Microsoft Analysis Services cubes.

All these applications are called MOLAP applications; they're analytic applications that read from multidimensional databases. Some analytic applications do not read from multidimensional databases. They read from relational databases instead, such as Micro Strategy OLAP. These are called ROLAP applications. To be able to respond quickly to a query such as "What is the total revenue for fiscal week 4 in 2008 from all stores in region 4?", ROLAP applications store the totals (known as aggregates) in summary tables. These aggregates are calculated and stored in summary tables when the ROLAP structure is processed. This way, when end users submit their queries, the ROLAP application doesn't need to access a lot of records on the fact table. The ROLAP application can just access the summary table to get the precomputed total.

Using analytic applications, we can get an overview of the current business performance, such as sales, production, purchasing, or profitability (probably by region) and compare it with the budget or targets. We can then drill down into specific areas that have problems and try to find out the cause of the problems. That is a typical scenario of BI analysis. But there are also other scenarios; for example, in the telecommunications industry we can analyze the line usage patterns (by weekly time slots) against the bandwidth cost. Or in the waste management industry we can weigh the revenue stream (which is by volume) vs. the disposal cost (which is by weight). Using analytic applications in monetary and technical organizations, such as combustion engine laboratories, nationwide economic management, and weather measurement stations, are totally different from the "normal" business organizations (companies) mentioned earlier.

We can swap any dimension with another dimension to get a different business perspective, going up and down the dimensional hierarchy to get different levels of summary, and we can pick certain dimension members and focus our attention on them. Using analytic applications, we can performad hoc queries, drill across to another

cube, analyze data from several cubes, find and locate certain data, and perform various kinds of formatting and manipulation with regard to the results, such as adding or removing columns and performing calculations.

The main advantage of using analytic applications is the flexibility. If the users don't know what they are trying to find in advance, then this is the right tool for them. Using analytic applications, we have the whole data warehouse at our fingertips. This is especially useful for companies and organizations that make business decisions based on data rather than instincts, such as performing quantitative analysis to justify the decisions. The main disadvantage of analytic applications is the complexity. It takes some time for the users to get familiar with the tools. Also, the multidimensional databases' data sources need to be maintained properly. But once everything is set and ready, then it's easy to keep the operation going. In analytic applications, a multidimensional database (cube) is fundamental, because that's where the data is stored.

QUESTION BANK

Q-1: 1 Mark Question

1. Give the full form of BIDS.(July-2021)
2. Give the full form of SSRS. (July-2021,2019)(2023)
3. Give the full form of BI. (July-2021,2019)
4. Give the full form of BIML. (July-2021)
5. What is Package? (July-2021)
6. What is Data Analysis? (July-2021)(2023)
7. What is SQL CLR? (July-2021)
8. What is SSAS? (July-2021,2019)
9. SSAS Stand for ___. (BKNMU- april-2020)(2023)
10. Each Cube has one or more Dimensions. True/False (BKNMU - april-2020)
11. List out type of Authentication to Connect SQL Server. (BKNMU - april-2020,2019)
12. Extension of Reports is__. (BKNMU - april-2020)
13. SSRS stands for _____. (BKNMU - april-2022)
14. BIDS stands for _____. (BKNMU - april-2022)
15. RDL stands for _____. (BKNMU - april-2022)

Q-2: 3 Mark Question

1. What is the use of Aggregate in SSIS? (July-2021)
2. Give the advantage of Business Intelligence. (July-2021)
3. Explain Benefits of Report Projects. (July-2021)
4. Why we need SSIS? (July-2021)
5. Explain Event Handlers in SSIS. (April–2020)
6. What is deployment? (BKNMU - June-2021,2019)
7. What is data analysis? (BKNMU - June-2021,2019)
8. What is data mining? (BKNMU - June-2021,2019)
9. Give the content of analysis service project folder. (BKNMU - June-2021)
10. Explain Package Deployment in SSIS. (BKNMU - april-2020)
11. Differentiate project deployment model v/s package deployment model. (BKNMU - april-2022)(2023)
12. Discuss data mining. (BKNMU - april-2022)
13. What is data analysis? (BKNMU - april-2022)
14. Write steps to add a chart to report. (BKNMU - april-2022)
15. Discuss control flow v/s data flow. (2023)
16. Write benefits of business intelligence reporting. (2023)
17. Write a note on SSRS configuration manager. (2023)

Q-3: 5 Mark Question

1. Explain Business Intelligence. (July-2021,2020,2019)
2. Give the step of Deploying SSIS package Step by Step. (July-2021)(2023)
3. What is Data Analysis? Explain in detail. (April–2020)
4. Explain deployment process. (BKNMU - June-2021,2020)
5. Write a step how to create report using report wizard. (BKNMU - april-2020,2019)
6. Write a step for planning package in ssis.
7. Explain business intelligence in detail. (BKNMU - april-2022)(2023)
8. Write a note on planning of SSIS package execution. (BKNMU - april-2022)