Data Warehouse Wiz

Tutorial:
Introduction to Data Warehousing Using Data Warehouse Wiz

Version 2.1.1.15

Magenta LLC
October 2018
Copyright and Version

This document is copyright © 2018 by Magentic LLC and covers Data Warehouse Wiz software version 2.x. This document may be reproduced, copied and distributed for non-commercial purposes, provided the document remains in its complete original form.

The Data Warehouse Wiz software is copyright © 2018 by Magentic LLC, all rights reserved. To download the latest version of the software, go to: www.DataWarehouseWiz.com.

.NET™ and Visual Studio™ are trademarks of Microsoft, Inc. All other trademarks mentioned herein are property of their respective owners.

THIS PUBLICATION IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS PUBLICATION COULD CONTAIN TYPOGRAPHIC ERRORS AND/OR TECHNICAL INACCURACIES. UPDATES AND MODIFICATIONS MAY BE MADE TO THIS DOCUMENT AND/OR SUPPORTING SOFTWARE AT ANY TIME.

www.datawarehousewiz.com

Comments and suggestion are welcome! Please send them by email to support@datawarehousewiz.com.
Table of Contents

Copyright and Version..............................................................................................................2

1. What is Data Warehouse Wiz?..............................................................................................5

2. Tutorial Overview..................................................................................................................6

3. Tutorial Setup........................................................................................................................7
   Create Databases..................................................................................................................8
   Start Data Warehouse Wiz and Connect to the Databases....................................................9

4. Create an ETL Group..............................................................................................................12

5. Dimension Tables...................................................................................................................14
   Design DownloadCustomer..................................................................................................14
   Design DownloadInvoice......................................................................................................16
   Design FlatCustomer............................................................................................................17
   Design FlatInvoice...............................................................................................................18
   Design DimCustomer..........................................................................................................19
   Design DimInvoice..............................................................................................................22

6. Fact Tables.............................................................................................................................25

7. ETL Stored Procedures..........................................................................................................27
   Procedure for FlatCustomer..................................................................................................27
   Procedure for FlatInvoice....................................................................................................31
   Procedure for DimCustomer..................................................................................................32
   Procedure for DimInvoice.....................................................................................................32
   Procedure for FactInvoice....................................................................................................32
   ETL Group Procedure............................................................................................................35
   ETL SSIS Package..................................................................................................................37

8. Junk Dimension Tables.........................................................................................................48
   Design DownloadInvoiceStatus and DownloadPaymentStatus...........................................48
   Design FlatInvoiceStatus and FlatPaymentStatus...............................................................48
   Add New Columns to FlatInvoice........................................................................................49
   Define Junk Table JDimInvoice............................................................................................49
   Procedure for FlatInvoiceStatus..........................................................................................50
   Procedure for FlatPaymentStatus........................................................................................51
   Add New Column ETL to FlatInvoice_upd...........................................................................52
   Procedure for JDimInvoice....................................................................................................54
   Add Foreign Key JDimInvoiceKey to FactInvoice.................................................................54
   Add New Column ETL to FactInvoice_upd...........................................................................55
   Recompile ETL Package........................................................................................................56

9. Bridge Dimension Tables......................................................................................................57
   Design DownloadSalesTeam and DownloadEmployee.........................................................57
   Design FlatSalesTeam and FlatEmployee............................................................................58
   Design DimEmployee............................................................................................................58
   Define Bridge Table BridgeSalesTeam.................................................................................58
   Add New Column BridgeSalesTeamKey to FactInvoice.......................................................60
   Procedure for FlatSalesTeam................................................................................................60
   Procedure for FlatEmployee..................................................................................................60
   Procedure for DimEmployee..................................................................................................60
   Procedure for BridgeSalesTeam.............................................................................................61
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add New Column BridgeSalesTeamKey to FactInvoice_upd</td>
<td>62</td>
</tr>
<tr>
<td>Recompile ETL Package</td>
<td>64</td>
</tr>
<tr>
<td><strong>10. Outrigger Dimension Tables</strong></td>
<td>65</td>
</tr>
<tr>
<td>Design DownloadDemography</td>
<td>65</td>
</tr>
<tr>
<td>Design FlatDemography</td>
<td>65</td>
</tr>
<tr>
<td>Design DimDemography</td>
<td>65</td>
</tr>
<tr>
<td>Add New Column DimDemographyKey to DimCustomer</td>
<td>65</td>
</tr>
<tr>
<td>Procedure for FlatDemography</td>
<td>66</td>
</tr>
<tr>
<td>Add New Outrigger Relationship to DimCustomer_upd</td>
<td>67</td>
</tr>
<tr>
<td>Procedure for DimDemography</td>
<td>68</td>
</tr>
<tr>
<td>Recompile ETL Package</td>
<td>68</td>
</tr>
<tr>
<td><strong>11. Pivot Tables</strong></td>
<td>70</td>
</tr>
<tr>
<td>Design DownloadPhone</td>
<td>70</td>
</tr>
<tr>
<td>Design FlatPhone</td>
<td>70</td>
</tr>
<tr>
<td>Design Pivot Table PivotPhoneWithExt</td>
<td>70</td>
</tr>
<tr>
<td>Add New Columns to FlatCustomer and DimCustomer</td>
<td>71</td>
</tr>
<tr>
<td>Procedure for FlatPhone</td>
<td>71</td>
</tr>
<tr>
<td>Procedure for PivotPhoneWithExt</td>
<td>72</td>
</tr>
<tr>
<td>Update Procedure for FlatCustomer_upd</td>
<td>73</td>
</tr>
<tr>
<td>Update the Procedure for DimCustomer_upd</td>
<td>75</td>
</tr>
<tr>
<td>Recompile ETL Package</td>
<td>76</td>
</tr>
<tr>
<td><strong>Alphabetical Index</strong></td>
<td>77</td>
</tr>
<tr>
<td><strong>Illustration Index</strong></td>
<td>79</td>
</tr>
<tr>
<td><strong>Online Video Index</strong></td>
<td>80</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td>81</td>
</tr>
</tbody>
</table>
1. **What is Data Warehouse Wiz?**

Data Warehouse Wiz (DWiz) is a software tool for the creation, loading, maintenance, and modular augmentation of data warehouses and data marts. By using this tool, you can dramatically reduce the development time needed for these tasks—especially in the ETL (extract-transform-load) processes required for both your end-user data warehouse database and the intermediate staging database.

This tutorial demonstrates the use of Data Warehouse Wiz in quickly creating a data warehouse “from scratch”, starting only with the tutorial source database that simulates a company's main operational database.

The latest downloads for Data Warehouse Wiz—software and manuals—are available from DataWarehouseWiz.com.

*Illustration 1: DataWarehouseWiz Control Areas*
2. Tutorial Overview

In this tutorial, we will create a star-schema data warehouse that is compliant to the Kimball principles of dimensional modeling [1]. We will start with a small tutorial Source database, provided with the DWiz app, that simulates a company's operational database. We will create both an intermediate Staging database to house the ETL processes, and an end-use Data Warehouse database. We will create tables in the Staging database to accept data, and maintain history, from the Source database and to facilitate ETL. We will create ETL processes to load and maintain these tables. We will create our star-schema tables in the Data Warehouse database, and will create ETL processes to load and maintain the warehouse with data from the Staging database.

In Chapter 3, “Tutorial Setup”, we will set up the software for the tutorial. This includes installing a small tutorial Source Database which poses as our starting "operational" DB. The rest of the tutorial will then demonstrate the creation of a data warehouse derived from this source db.

In Chapters 4 -7, we design and deploy our first iteration of a basic data warehouse: two dimension tables and a fact table.

In Chapter 8, “Junk Dimension Tables”, we do a modular augmentation of our warehouse, adding a Junk Dim table.

In Chapter 9, “Bridge Dimension Tables”, we add a Bridge table to our growing production data warehouse.

In Chapter 10, “Outrigger Dimension Tables”, we add an Outrigger Table to our warehouse.

In Chapter 11, “Pivot Tables”, we show an example of how DWiz can be used to easily create pivot tables, which are frequently useful in the transforms for a data mart.

In Chapter 12, “Transformations in the Flat Definition Page”, we look at examples which use the advanced features available in the DWiz Flat Definition Page.
3.  Tutorial Setup

Please refer to the Data Warehouse Wiz Installation Manual for detailed instructions concerning installation. The installation manual is available from DataWarehouseWiz.com, where you will also find the latest downloads for Data Warehouse Wiz. Normally installation involves the following steps:

• Create a free online account at https://www.datawarehousewiz.com/Account/Register
• Order a Free Trial or Paid License activation token
• Download the app installer from https://www.datawarehousewiz.com/Download
• Run the installer
• Run the newly-installed DWiz app, click on the menu item Help->Registration, and enter your activation token

In addition to the Data Warehouse Wiz app, you will need access to Microsoft Sql Server, where you will load the sample tutorial source database as well as create new staging and warehouse databases during the course of the tutorial.
Create Databases

When you begin any new warehouse, start by creating the necessary empty staging and warehouse databases. In addition, for this tutorial, we also will need to create the tutorial source database, which will pose as the “operational source db” from which we want to derive a data warehouse. Open Microsoft® Sql Server Management Studio (SSMS) and create empty databases DWizStaging, DWizWarehouse, and DWizSource. You may substitute other names for these databases; if so, substitute your names consistently for DWizStaging and DWizWarehouse and DWizSource throughout this reference manual. SSMS allows the easy creation of databases by right-clicking “Databases” in the Object Explorer and then selecting “New Database…”.

![Illustration 2: Creating Databases in Microsoft Sql Server Management Studio](image-url)
**Start Data Warehouse Wiz and Connect to the Databases**

After opening Data Warehouse Wiz, connect to the Staging/Warehouse databases by clicking on the “Key” icon or by selecting File-->Connect to DW:

From the popup window, enter connections to DWizStaging and DWizWarehouse, then press Connect:

*Illustration 3: Connecting to Staging & Warehouse DBs*

*Illustration 4: Connect-To-Database Pop-Up*
The first time that you connect to the Staging DB, the app will ask to write metadata tables to it:

*Illustration 5: OneTime Initialization of DWiz Metadata*

Please select “Yes” to create these metadata tables in the staging database. These tables will hold your project data for ETL processes that you will create through Data Warehouse Wiz.

Next, from the top menu bar, select Help->Tutorial->Install Tutorial tables:

*Illustration 6: Installing the Tutorial Tables*
The app will ask for a connection to the Source database, where it will write the tables, so enter a connection to the empty DWizSource. Then press Connect:

![Illustration 7: Connect-To-Source-DB Pop-Up for the Tutorial Source DB, DWizSource](image)

After connecting to DWizSource, click through the app's top-level menu to Help->Tutorial->Install Tutorial Tables. The app writes the tutorial tables to the DWizSource database. This database then becomes the “source” DB for the tutorial exercises, posing as the “operational database” of a small company that wants a data warehouse.

This concludes the set-up, and we are ready to begin designing a warehouse.
4. Create an ETL Group

First we will form an ETL Group, which may be thought of as a “project”. This group will begin empty, and grow to contain tables and processes related to one manageable portion of the overall ETL needed to feed our warehouse. In many cases, you will only need one ETL group. However, for warehouses with multiple sources or a large number of tables, it is helpful to be able to divide the action into smaller manageable chunks— and thus multiple ETL Groups.

In the Object Explorer, expand DWizStaging-->ETL-->Groups, and right-click “Groups”, then select “New Group”:

Illustration 8: Creating an ETL Group
Next enter a name ("TutorialGroup" in the example) in the ETL Group Page, and save by using the Floppy-Disk icon on the page. After a successful save, you will see a "Save successful!" message at the bottom:

Now that we have our ETL Group, we can define tables and processes to go into it.
5. Dimension Tables

In the first iteration of our star-schema warehouse, we plan to make a fact table of invoice item sales, a Customer dimension table, and an Invoice dimension table. We will create the tables and processes for this simple warehouse, and later can come back and add to it. Since the fact table will need to include foreign keys from the dimension-table primary keys, we will create the dimension tables first.

Design DownloadCustomer

We will create a download table, called DownloadCustomer, in WizStaging to facilitate the downloading and processing of the “Customer” table data from the Source DB. In addition to capturing the initial data from Customer, the DownloadCustomer table will also capture changes from Customer over time, perhaps in nightly updates. In this step, however, we are only copying the schema of the source table; we are not copying any data records yet. The actual data will not be copied until our ETL Process is designed and run in a later step. In Object Explorer, expand DWizSource-->Tables to view the source tables, then right-click the Customer table and select Copy Table:

Illustration 10: Copying a Table's Schema

A popup window allows you to choose a name for the new table in Staging (default is "Download" + source table name). Press OK:
This brings us to the DownloadCustomer table definition page:

Most of the parameters of this page are conveniently filled with default values by Data Warehouse Wiz, yet you can override any parameter for your own purposes. The first line of the page details the location and type of table DownloadCustomer:

Database = DWizStaging
Schema = dbo
Table Type = download
Table = DownloadCustomer

The second line details the source:
Source Schema = dbo
Source Table = Customer
Download ETL Group = TutorialGroup (be certain to select this group)
Connection = <select your connection to a Source DB>

The columns of DownloadCustomer default to copies of the source Customer, but may be modified as needed. Also, columns may be added or deleted. Ensure that the source primary key is indicated properly by a “1”. If the source has a multi-part primary key, each part should be indicated sequentially (1,2,...). The app also quietly adds several columns (prefixed by “_$” or “__$”) for maintenance purposes. When you have completed the page, save it with the Floppy-Disk icon. Afterward, it is possible to view the new table within the Staging DB in SSMS.

Design DownloadInvoice

Now repeat the above step to create a download table called DownloadInvoice, based upon the source DB's Invoice table. Right-click the Invoice table under DWizSource and select Copy Table. Fill in the definition page to create DownloadInvoice, being certain to select the ETL group “TutorialGroup” and to specify the DWizSource Connection. After you press the save icon (Floppy-Disk), it should look like this:
We will create a flat table, called FlatCustomer, in DWizStaging. We will copy the schema of DownloadCustomer to create FlatCustomer, but we will not copy any data yet. In the Object Explorer, expand DWizStaging-->Tables-->Download and right-click on DownloadCustomer, then select Copy Table. In the pop-up window, set the table type to "Flat", check the name (FlatCustomer), and press OK. Ensure that the database is set to DWizStaging and the table type is Flat!:
This brings us to the Flat Table Definition page:

Illustration 15: "Flat Table" Definition Page

Most of the parameters of this page are conveniently filled with default values by Data Warehouse Wiz; yet you can override any parameter for your own purposes. The first line of the page details the location and type of table FlatCustomer:

- **Database** = DWizStaging
- **Schema** = dbo
- **Table Type** = flat
- **Table** = FlatCustomer

The columns of FlatCustomer default to copies of DownloadCustomer, but may be modified as needed. Also, columns may be added or deleted. Ensure that the primary key is indicated properly by a “1”. If the table has a multi-part primary key, each part should be indicated sequentially (1,2,...). The app also quietly adds several columns (prefixed by “_$_” or “__$_”) for maintenance purposes. When you have completed the page, save it with the Floppy-Disk icon. Afterward, it is possible to view the new table within the Staging DB in SSMS.

**Design FlatInvoice**

Now repeat the above step to create a flat table called FlatInvoice, based upon the DownloadInvoice table. Right-click the DownloadInvoice table under DWizStaging and select Copy Table. Fill in the definition page to create FlatInvoice, being certain to specify the primary key(s). After you press the save icon (Floppy-Disk), it should look like this:
Now we are ready to create the dimension tables proper, based on the Flat tables. We will create a table called DimCustomer, in DWizWarehouse. Unlike the intermediate Download and Flat tables, this Dim table will reside in the end-user warehouse. Although we will copy the schema of FlatCustomer to use as a basis for DimCustomer, we will not copy any data yet...that will happen when we run our future ETL Process. In Object Explorer, expand DWizStaging-->Tables-->Flat to view the flat tables, then right-click the FlatCustomer table and select Copy Table. In the pop-up window, change the database to DWizWarehouse, Schema to "dbo", and the table type to “Dim”:
This brings us to the Dim Definition page:

Illustration 18: "Dimension Table" Definition Page

As with the other table definition pages, most of the parameters of this page are conveniently filled with default values by Data Warehouse Wiz; yet you can override any parameter for your own purposes. The first line of the page details the location and type of table DimCustomer:

- **Database** = DWizWarehouse
- **Schema** = dbo
- **Table Type** = dim
- **Table** = DimCustomer

The columns of DimCustomer default to copies of FlatCustomer, but may be modified as needed. Also, columns may be added or deleted. The “Natural Key” corresponds to the primary key(s) of FlatCustomer, but will not be used as a primary key for DimCustomer. The app will quietly add a primary key “DimCustomerKey” when the definition page is saved. In keeping with best practices, each warehouse dimension table will use a surrogate primary key (e.g., DimCustomerKey) which is unrelated to the primary key(s) of the source(s).

Ensure that the Natural key is indicated properly by a "1". If it is multi-part, then each part should be indicated sequentially (1,2,...). The definition page should look like this before you save it:
When you have completed the page, save it with the Floppy-Disk icon. Afterward, the DimCustomer definition page will show the DimCustomerKey that was quietly added:

You can review the DimCustomer definition page at any time by expanding DWizWarehouse-->Tables-->Dim and clicking on DimCustomer. Of course, it is also possible to view the new table within the DWizWarehouse DB in SSMS. Although the data has not been copied yet, DWiz has put one special record in the new dim table: the Not Applicable (or “Unknown”) record, with DimCustomerKey = -1. This value (-1) of the
dimension table primary key will be referenced in the future fact table where-ever the customer table would otherwise not have an entry (hence, Not Applicable or Unknown). This satisfies the Foreign Key-Primary Key relationship between the tables. To see what values DWiz uses for columns in the Not Applicable records, go to the Object Explorer and expand Staging-->Settings-->NA Record. Also expand Settings-->IsNull to see the values substituted for nulls where necessary. You can change any of these settings for your own purposes.

**Design DimInvoice**

We create the DimInvoice dimension table in much the same way, but will delete some columns. This Dim table will reside in the end-user warehouse. In Object Explorer, expand DWizStaging-->Tables-->Flat to view the flat tables, then right-click the FlatInvoice table and select Copy Table. In the pop-up window, change the database to DWizWarehouse and the table type to “Dim”:

![Illustration 21: "Copy Table" Pop-up for DimInvoice](image)

This brings us to the DimInvoice definition page--but don't press Save just yet:
We will delete most of the columns. Why? Because most of them do not belong in a dimension table. We will eventually make use of these columns--just not here. Some are facts (which belong in a fact table), some are appropriate for other dimensions (and will be used there), some are not slowly-changing attributes of the Invoice. In the DimInvoice dimension, we want only true attributes of the Invoice, and only ones that change at most very rarely. So in this example, we will delete all the columns except InvoiceID, InvoiceNumber, and DeliveryInstructions. You can delete a column by right-clicking it, then selecting Delete Column. After deleting all the columns except for the above three, put a "1" in the Natural Key column of InvoiceID. This indicates that InvoiceID is the natural key of the source table. As with DimCustomer, DWiz will quietly create a new primary key for this DimInvoice table, called DimInvoiceKey. After you make these changes, but before Save, the page looks like this:
Now press Save (Floppy-Disk icon). Now we have two dimension tables, so we can move on to creating a fact table.
Now we are ready to create a fact table, called FactInvoice. This Fact table will reside in the end-user warehouse. As before, we will not copy any data yet...that will happen when we run our future ETL Process. In Object Explorer, expand DWizStaging-->Tables-->Flat to view the flat tables, then right-click the FlatInvoice table and select Copy Table. In the pop-up window, change the database to DWizWarehouse, Schema to "dbo", and the the table type to “Fact”:

This brings us to the FactInvoice definition page. Delete the columns InvoiceNumber and DeliveryInstruction (we used these in the Dim table and do not want them in the Fact table). Append two additional columns, DimCustomerKey and DimInvoiceKey, by typing into the blank row at the bottom. Columns may be deleted by doing a right-click and selecting "Delete". Columns may be added by typing into the blank row at the bottom, or by doing a right-click and selecting "Insert".
You may add, delete, or modify columns as necessary. In this example, we have added two foreign keys, DimCustomerKey & DimInvoiceKey, which we will use to link in the dimension tables. (The actual link mechanism will be coded in the Fact Update Procedure definition page.)

Now press the Save button (Floppy-Disk icon). Notice that DWiz quietly adds a primary key, FactInvoiceID.
7. **ETL Stored Procedures**

In this chapter, we will deploy our first version of a working data warehouse. In later chapters, we will demonstrate modular upgrades to our working warehouse by adding bridge tables, junk tables, outrigger tables, pivot tables, and accumulation (snapshot) tables.

In this first version of our warehouse, we have designed one fact table (InvoiceFact) and two dimensions (DimCustomer and DimInvoice). Now we need to design the ETL Process that will load and maintain our warehouse.

The ETL data flows in stages: Download tables will be loaded from their corresponding Source tables/files, directly one-to-one. Each Flat table will be loaded with processed data from one or more Download tables—not necessarily one-to-one. Then the warehouse tables--Dims, Facts, Bridges, Junks, Outriggers, etc--will be loaded with data from Flat tables. In several places, Stored Procedures will be used to handle the ETL steps. Stored Procedures are small programs of SQL code that are embedded in the database itself. DWiz will write the Stored Procedures for you, but give you an opportunity to view and modify the code if you wish.

**Procedure for FlatCustomer**

Next we will create a stored procedure for the ETL into FlatCustomer. We will call this a "Flat" stored procedure because it is loading a Flat table. A "Flat" stored procedure resides in the Staging DB, where its code controls the ETL into a Flat table from one or more Download tables. So in the Object Explorer, we expand DWizStaging-->Stored Procedures-->Flat, right-click on the Flat folder, then select New Flat:
This gives us a popup window, where we will give the stored procedure a name and indicate the primary source and destination tables:

We are creating this procedure to load & update FlatCustomer, so a suitable name is “FlatCustomer_upd”. The Source Table is DownloadCustomer (note that "Source" here does not refer to the Source DB: All Flat stored procedures draw from tables sourced by the Staging DB). The Flat Table destination is FlatCustomer. When you have completed the form, press OK.

This brings us to the Flat Definition Page:
On this page, we need to set the ETL Group to TutorialGroup, indicate column CustomerID as the Natural Key with a "1", and press the Save button (Floppy-Disk icon). Note that this page provides many advanced capabilities for transforming the data: Source Filter, Additional Joins, Update options, aggregation options. We will describe these optional capabilities in later chapters.

After completing the form and pressing Save, we will need to create the stored procedure. We do this by pressing the Create/Alter Procedure button (Lightning/+ icon as shown) below:
When the above button is pressed, DWiz designs the stored procedure for you! The actual code is displayed, as shown below, and is complete; however, DWiz gives you this opportunity to modify the code if desired. After looking at the code (or not!), press the Save Procedure button as indicated (Lightning over Floppy-Disk icon):
Illustration 30: Reviewing DWiz's Stored Procedure Code Prior to Compiling It

The FlatCustomer_upd stored procedure is now saved into the DWizStaging DB.

**Procedure for FlatInvoice**

Now you can repeat the above steps to create an ETL stored procedure for FlatInvoice. Perform the following steps:

- Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.
- Create Flat Definition Page with Stored Procedure Name = FlatInvoice_upd, Source Table = DownloadInvoice, and Flat Table = FlatInvoice. Then press OK.
- In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column InvoiceID, and then press the Save icon.
- Press the Create/Alter Procedure button (Lightning/+ icon).
- After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).
Procedure for DimCustomer

Next we will create the ETL for DimCustomer in a similar fashion. Since DimCustomer resides in the DWizWarehouse DB, its stored procedure will reside there as well. A Dim stored procedure will load a Dim table from one or more tables. Perform the following steps:

- Expand DWizWarehouse --> Stored Procedures --> Dim, right-click the Dim folder, and select New Dim.
- Create Dim Definition Page with Stored Procedure Name = DimCustomer_upd, Source Table = FlatCustomer, and Dim Table = DimCustomer. Then press OK.
- In the Dim Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column CustomerID, and then press the Save icon. Note that the Dim's primary key, DimCustomerKey, is not the Natural Key.
- Press the Create/Alter Procedure button (Lightning/+ icon).
- After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

Procedure for DimInvoice

Next we will create the ETL for DimInvoice in a similar fashion. Perform the following steps:

- Expand DWizWarehouse --> Stored Procedures --> Dim, right-click the Dim folder, and select New Dim.
- Create Dim Definition Page with Stored Procedure Name = DimInvoice_upd, Source Table = FlatInvoice and Dim Table = DimInvoice. Then press OK.
- In the Dim Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column InvoiceID, and then press the Save icon.
- Press the Create/Alter Procedure button (Lightning/+ icon).
- After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

Procedure for FactInvoice

Next we will create the ETL for FactInvoice in a similar fashion; however, the ETL processing for this fact table will be a bit more complex. Perform the following steps:

- Expand DWizWarehouse --> Stored Procedures --> Fact, right-click the Fact folder, and select New Fact.
- Create Fact Definition Page with Stored Procedure Name = FactInvoice_upd, Source Table = FlatInvoice and Fact Table = FactInvoice. Then press OK.
- In the Fact Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column InvoiceID, and then press the Save icon.
- Press the Create/Alter Procedure button (Lightning/+ icon).
- After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).
Key box of column InvoiceID, and then press the Save icon.

We saved the Fact Definition page, but we are not finished with it yet, so leave it open please! Now for the new steps. We will define the relationship that the Fact table has with the Dim tables. Press the Add Dim button as shown:

Illustration 31: Adding a Dimension-Table Relationship to a Fact-Table Procedure

In the popup window, select DimCustomer and press OK:

Illustration 32: "Add Dim Table" Pop-Up for Fact-Table Procedure
Now scroll down in the Fact Definition Page to reveal the new Fact To Dim section for the DimCustomer table:

This Fact To Dim section defines the relationship of the Fact table to the DimCustomer dimension. Perform the following steps:

• Set Fact Table Key to "DimInvoiceKey"
• Set Dim Table Key to "DimInvoiceKey"
• Indicate InvoiceID as the first (and only) Natural Key with a "1"
• Press the Save button at the top (Floppy-Disk icon indicated in figure above)

Now we will add the DimInvoice relationship in a similar fashion. Perform the following steps:

• Press the Add Dim button
• In the popup, select DimInvoice and press OK
• Scroll down in the Fact Definition page to reveal the new Fact To Dim section for DimInvoice
• Set Fact Table Key to "DimInvoiceKey"
• Set Dim Table Key to "DimInvoiceKey"
• Indicate InvoiceID as the first (and only) Natural Key with a "1"
• Press the Save button at the top (Floppy-Disk icon indicated in figure above)
• Press the Create/Alter Procedure button (Lightning/+ icon).
• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

This completes the stored procedures that we need for a first deployment, including procedures that update Flats, Dims, and Facts. Next, we will wrap all the ETL processing, including the procedures, into a single package. This package could then be run on a periodic schedule, such as nightly, to maintain the warehouse. The advantage of using an ETL Group can be seen now: everything needed for a single ETL Group is wrapped into a single package; additional ETL Groups can be designed and built in a modular fashion without affecting existing groups. DWiz provides a choice of two ways to build an ETL package:

1. ETL Group Procedure. This is the easier way. One master ETL Group Procedure is created, that calls and orchestrates all the download processing, transform procedures, and Update Procedures, of the ETL group. This ETL Group Procedure can then be scheduled as a Sql Job in Sql Server Agent, or called by a scheduled program.

2. ETL SSIS Package. This way provides superior performance, particularly of benefit when the databases are very big. This method is more complicated to program, as it involves compiling a BIML file in Visual Studio, and building an SSIS package. The SSIS package can then be scheduled in Sql Server Agent.

**ETL Group Procedure**

[Online Video: Compiling an ETL Group]

In this first example, we will create the ETL Group Procedure. Start by creating an ETL Definition Page by performing the following steps:

• Expand DWizStaging-->ETL-->Definitions, right-click the Definitions folder, and select New Definition.
• In the ETL Definition Page, give the package a name such as TutorialETL, set the ETL Group to "TutorialGroup", checkmark the options "Staging Tasks" and "Datamart Tasks", and select your connections for Staging and DataMart (warehouse).

Staging Tasks include all ETL within the Staging DB, including downloads, flat updates, and transform processing.

Datamart Tasks include all ETL within the Warehouse DB, including updates to Dimensions, Facts, and other warehouse tables.

Before pressing Save, the page will look similar to this:
Now press the Save icon. Notice that DWiz adds a connection for the Source DB, which is DWizSource in this example. In this new Source row, select "Change Tracking" under the Download Methods, and press Save.

The available download method choices are:

1. Change Tracking. This is the preferred method, as it is the most efficient. DWiz will use Microsoft Change Tracking in the Source DB to follow and download changes (Updates/Inserts/Deletes). The Source DB must support Change Tracking, and both the DB and the tables to be downloaded must be configured for Change Tracking.

2. Change Data Capture. This is an alternate method using Microsoft Change Data Capture. The Source DB must support Change Data Capture.

3. Whole Table. The entire Source table is downloaded on every run. This method is only practical for small source tables.

4. No Download. No data is downloaded from the Source. This method might be used if the data is already being downloaded by a different ETL Group which runs prior to this one.

5. Excel. The Source data is downloaded from Excel spreadsheet files.
Connection Strings:
The connection strings can be viewed by hovering the mouse over their columns. These strings can be edited and updated through the Object Explorer by expanding Staging-->ETL-->Connections and clicking on the appropriate connection. After editing and saving the connection, re-open the ETL Definition Page and re-save it.

Illustration 35: Choosing a Download Method in the ETL Definition Page

After Saving the ETL Definition Page, press the "Create ETL Group Procedure" button on the Tab Button Bar (icon is lightning bolt with green "plus"). DWiz will write all the code for you and allow you to review it. After reviewing (or not!), press the Save Procedure button on the Tab Button Bar (icon is lightning bolt with floppy disk). Note that the new procedure created will have the same name as the ETL Name in the definition page. This procedure is then all you need to run the ETL. You can skip the next section (ETL SSIS Package) unless you want the better performance of the SSIS package.

**ETL SSIS Package**

Online Video: Compiling a BIML/SSIS Package

The ETL SSIS Package is offered as a higher-performance alternative to the ETL Group Procedure. For each ETL Group, you can choose to run an ETL Group Procedure OR an
ETL SSIS Package. In this example, we will create the ETL SSIS Package. Start by creating an ETL Definition Page by performing the following steps:

- Expand DWizStaging-->ETL-->Definitions, right-click the Definitions folder, and select New Definition.

- In the ETL Definition Page, give the package a name such as TutorialETL, set the ETL Group to "TutorialGroup", checkmark the options "Staging Tasks" and "Datamart Tasks", and select your connections for Staging and DataMart (warehouse).

Before pressing Save, the page will look similar to this:

Illustration 36: Selecting Connections in the ETL Definition Page

Now press the Save icon. Notice that DWiz adds a connection for the Source DB, which is DWizSource in this tutorial. In this new Source row, select "Change Tracking" under the Download Methods, and press Save.

The available download method choices are:

1. Change Tracking. This is the preferred method, as it is the most efficient. DWiz will use Microsoft Change Tracking in the Source DB to follow and download changes (Updates/Inserts/Deletes). The Source DB must support Change Tracking, and both the DB and the tables to be downloaded must be configured for Change Tracking.

2. Change Data Capture. This is an alternate method using Microsoft Change Data Capture. The Source DB must support Change Data Capture.

3. Whole Table. The entire Source table is downloaded on every run. This method is only practical for small source tables.
4. No Download. No data is downloaded from the Source. This method might be used if the data is already being downloaded by a different ETL Group which runs prior.

5. Excel. The Source data is downloaded from Excel spreadsheet files.

Connection Strings:

The connection strings can be viewed by hovering the mouse over their columns. These strings can be edited and updated through the Object Explorer by expanding Staging-->ETL-->Connections and clicking on the appropriate connection. After editing and saving the connection, re-open the ETL Definition Page and re-save it.

Illustration 37: Re-Save the ETL Definition Page After Changing a Connection
Now press the Create BIML button (Scroll+Pen icon on top right):

Illustration 38: "Create BIML" Button on the ETL Definition Page
After the BIML button is pressed, DWiz will create the BIML for the entire ETL Process and display:

**Illustration 39: Reviewing BIML Code Written By DWiz**
You will copy this script into the clipboard and paste it into a Visual Studio project. Leave DWiz open, but also open Microsoft® Visual Studio on your PC:

Illustration 40: Creating an SSIS Package in Visual Studio

Your version of Visual Studio should include SQL Data Tools for Visual Studio, and also the Varigence® BIML Express plugin. For full details and installation tips, see the Data Warehouse Wiz Installation Manual.
In Visual Studio, create a new project (click File-->New-->Project) by using the template under Templates-->Business Intelligence-->Integrated Services Project, and name the project "TutorialETL", then press OK:

Illustration 41: Creating an Integrated Services Project in Visual Studio
In the Solution Explorer of the new project, right-click Package.dtsx, and select "Add New Biml File":

Illustration 42: Adding the BIML to a Visual Studio Project
In the Solution Explorer, now expand the Miscellaneous folder, and double-click on "BimlScript.biml" to open it:

Illustration 43: Opening the Default BIML File
This reveals an "empty" BIML script. Replace this script by cut-and-pasting the script that DWiz generated. Make sure that you overwrite the existing "empty" script completely. Save the file (click File-->Save All). This is what it should look like:

Illustration 44: Saving Your BIML in the Visual Studio Project
Now right-click BimlScript.biml (under Miscellaneous in Solution Explorer) and click "Generate SSIS Packages". This may take a few minutes. Afterward, you may view the data flow by double-clicking on the TutorialETL.dtsx package:

Illustration 45: Generating an SSIS Package

You may run the ETL Package by pressing the Start button on the top menu. Afterward, you can see the filled warehouse tables in SSMS.

The SSIS package dtsx could be run on a periodic schedule to maintain the warehouse.
In this chapter, we will augment the existing design from the previous chapters, by adding a Junk Dimension.

In the design of your warehouse, you may encounter numerous "facts" that are flags, status indicators, and such low-cardinality fields associated with the transactions in your fact table. These are often Y/N bit fields or other enumerated types with a small range of possible values. Junk Dimension tables are a useful tool for retaining this information without exploding the size of your fact table. Simply put, the idea is to collect up all of this "junk", even if the fields are uncorrelated, into one Junk Dimension which only requires one foreign key to be added to fact table instead of the many fields that would be necessary if the indicators were added individually.

In the example of this chapter, we begin with an existing fact table from the Tutorial, FactInvoice, to which we will add a new junk table that holds two status fields. In this example, the "junk" consists of two fields, PaymentStatus and InvoiceStatus, which originate in two separate Source DB tables.

**Design DownloadInvoiceStatus and DownloadPaymentStatus**

We will create download tables, called DownloadInvoiceStatus and DownloadPaymentStatus, in WizStaging to facilitate the downloading of the “InvoiceStatus” and "PaymentStatus" table data from the Source DB. We will use the same method as for DownloadCustomer on p.13.

Right-click the InvoiceStatus table under DWizSource and select Copy Table to define the new DownloadInvoiceStatus. Fill in the definition page, being certain to select the ETL group “TutorialGroup” and to specify the DWizSource Connection. Make sure that column InvoiceStatusID is indicated as the Source Primary Key with a "1", then press the Save icon (Floppy-Disk).

Repeat these steps for DownloadPaymentStatus: Right-click the PaymentStatus table under DWizSource and select Copy Table to define the new DownloadPaymentStatus. In the definition page, select the ETL group “TutorialGroup”, specify the DWizSource Connection, indicate PaymentStatusID as the Source Primary Key with a "1", and press Save.

**Design FlatInvoiceStatus and FlatPaymentStatus**

We will create Flat tables in WizStaging by using the same method as for FlatCustomer on p.16. Right-click the DownloadInvoiceStatus table under DWizStaging and select Copy Table to define the new FlatInvoiceStatus with Table Type = Flat:
Illustration 46: "Copy Table" Pop-up for FlatInvoiceStatus

In the definition page, indicate InvoiceStatusID as the Source Primary Key with a "1", and press Save.

Do the same for FlatPaymentStatus: Right-click the DownloadPaymentStatus table under DWizStaging and select Copy Table to define the new FlatPaymentStatus with Table type = Flat. In the definition page, indicate PaymentStatusID as the Source Primary Key with a "1", and press Save.

Add New Columns to FlatInvoice

In Object Explorer, double-click FlatInvoice (DWizStaging->Tables->Flat) to open it. In the definition page, append columns InvoiceStatus varchar(50) and PaymentStatus varchar(50); then press Save at the top of the definition page.

Define Junk Table JDimInvoice

Now we are ready to define the junk table proper, JDimInvoice, in the Warehouse DB. Right-click the FlatInvoice table under DWizStaging->Tables->Flat, and select Copy Table:

Illustration 47: "Copy Table" Pop-Up for Defining Junk Tables
Make sure to set:
• Database Name = DWizWarehouse
• Schema = dbo
• Table Type = Junk
• Table Name = JDimInvoice

In the JDimInvoice definition page, delete all the columns (even the keys) except for the two new columns at the bottom, InvoiceStatus and PaymentStatus; then Save:

Illustration 48: "Junk Table" Definition Page

Don't forget to Save the definition page! Note that DWiz silently creates the surrogate primary key, JDimInvoiceKey. With the tables all set, we are ready to define the stored procedures to load them.

**Procedure for FlatInvoiceStatus**

In a similar fashion to the steps on p.25, we will create an ETL stored procedure for FlatInvoiceStatus. Perform the following steps:
• Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.
• Create Flat Definition Page with Stored Procedure Name = FlatInvoiceStatus_upd, Source Table = DownloadInvoiceStatus, and Flat Table = FlatInvoiceStatus. Then press OK.
• In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column InvoiceStatusID, and then press the Save icon.

• Press the Create/Alter Procedure button (Lightning/+ icon).

• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

**Procedure for FlatPaymentStatus**

Repeat the steps to create an ETL stored procedure for FlatPaymentStatus:

• Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.

• Create Flat Definition Page with Stored Procedure Name = FlatPaymentStatus_upd, Source Table = DownloadPaymentStatus, and Flat Table = FlatPaymentStatus. Then press OK.

• In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column PaymentStatusID, and then press the Save icon.

• Press the Create/Alter Procedure button (Lightning/+ icon).

• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).
**Add New Column ETL to FlatInvoice_upd**

In Object Explorer, double-click FlatInvoice_upd (DWizStaging->Stored Procedures->Flat) to open it. In the top Tab Button Bar, press Define Additional Source:

![Illustration 49: Revising a Flat Update Stored Procedure](image)

In the resulting pop-up, select FlatInvoiceStatus:

![Illustration 50: "Add Additional Source" Pop-Up for a Flat Update Stored Procedure](image)

In the definition page, scroll down to reveal the new section at the bottom, Additional Source. Put a "1" in the Natural Key of column InvoiceStatusID. This section should look like the following figure (if the unused columns seem confusing, UN-checkmark "View All Columns"): 

![Definition Page](image)
Now press Save (Floppy-Disk) to re-save the modified page. Next we will generate the stored procedure, by pressing the Create/Alter Procedure button. After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

Now go back to the tab shown above (FlatInvoice_upd) so you can repeat these steps to add PaymentStatus:

- Press Define Additional Source button
- Select FlatPaymentStatus to be added and press OK
- Scroll down to reveal the new Additional Source section
- Put a "1" in the Natural Key of column PaymentStatusID
- Uncheck "View all columns" button
- Press Save Page button (Floppy-Disk)
- Press Create/Alter Procedure button
- Press Save Procedure button
We need to make one more procedure, that of the Junk Dim itself. The Junk Dim is part of the warehouse DB, so the procedure will reside there as well. Perform the following steps to create an ETL stored procedure for JDimInvoice:

1. Expand **DWizWarehouse** --> Stored Procedures --> **Junk**, right-click the Junk folder, and select New Junk.
2. Create Junk Definition Page with Stored Procedure Name = JDimInvoice_upd, Source Table = FlatInvoice, and Junk Dim Table = JDimInvoice. Then press OK.
3. In the Junk Dim Definition Page, select ETL Group = TutorialGroup, and UnCheck "View All Columns". This should leave only the two new columns in the display: InvoiceStatus & PaymentStatus. Press the **Save** icon.
4. Press the Create/Alter Procedure button (Lightning/+ icon).
5. After the code is displayed, press the Save Procedure button (Lightning over Floppy icon).

Add Foreign Key JDimInvoiceKey to FactInvoice

Expand DWizWarehouse->Tables->Fact, and double-click FactInvoice to open it. Append the column JDimInvoiceKey (type = int) to the table and Save it.
Add New Column ETL to FactInvoice_upd

In Object Explorer, double-click FactInvoice_upd (DWizWarehouse->Stored Procedures->Fact) to open it:

Illustration 52: Adding a Junk Table to a Fact Table Update Procedure

Press the Define Junk button (as shown above). From the pop-up, select JDimInvoice, and press OK:

Illustration 53: "Add Junk Table" Pop-Up

Perform the following:
• Scroll down to reveal the new Fact To Junk section
• Set Fact Table Key to "JDimInvoiceKey"
• Set Junk Table Key to "JDimInvoiceKey"
• Press Save
• Press the Create/Alter Procedure button (Lightning/+ icon).
• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

Recompile ETL Package

With all the tables and procedures done, we can now re-create the ETL Package, in a similar fashion to the steps on p.33. Our choices are 1) to compile an ETL Group Procedure, or 2) to compile an ETL SSIS Package. The first option is simple, just open the ETL Definition Page and compile the ETL Group Procedure. The second option (higher performance) entails the following steps:

• Expand **DWizStaging**--->ETL--->Definitions, right-click the Definitions folder, and double-click on TutorialETL
• Review ETL Definition Page (no changes are necessary), and press the Save button
• Press the Create BIML button (Scroll+Pen icon on top right)
• Open your TutorialETL project in SQL Data Tools for Visual Studio, and double-click the BimlScript.biml file to open it
• Copy (cut-and-paste clipboard) the new BIML from DWiz to overwrite the BimlScript in Visual Studio
• Save All in Visual Studio
• Right-click BimlScript.biml (under Miscellaneous in Solution Explorer) and click "Generate SSIS Packages". This may take a few minutes. Visual Studio may ask if you want to overwrite the package TutorialETL.dtsx; click on Yes.
• Afterward, you may view the data flow by double-clicking on the TutorialETL.dtsx package
• You may run the ETL Package by pressing the Start button on the top menu. Afterward, you can see the filled warehouse tables in SSMS.

This concludes our second iteration of the tutorial warehouse, by adding a Junk Dimension to an existing warehouse. The SSIS package TutorialETL.dtsx could be run on a periodic schedule to maintain the warehouse. The augmented warehouse is "finished" and usable. In the next chapter, we will add a Bridge Dimension.

DataWarehouseWiz.com
9. Bridge Dimension Tables

In this chapter, we will augment the existing design, from the previous chapters, by adding a Bridge Dimension.

During the design of your warehouse, you may encounter certain dimensions which have a many-to-many relationship with the grain of your fact table, which poses a problem for your design because a single transaction in your fact table can have multiple simultaneous values of this certain dimension type. Some common scenarios are:

* A patient receives multiple diagnosis codes for a single hospital visit
* An account has multiple owners or beneficiaries
* A sale or account is serviced by a team of employees

Bridge Dimension tables are a very tidy solution to this problem, without breaking the ease-of-use or efficiency of the fact table. The many-to-many values are collected into groups, and only the group key is added to the fact table. The bridge dimension then makes the many-to-many relationship of groups to individual values.

This is best illustrated by using an example. In this example, we have an existing invoice fact table of high-end sales, and we wish to add the (multiple) employees that contributed to each sale. To avoid the inherent problems with this many-to-many relationship, we will group the employees into ad-hoc "sales teams", creating a "team" for every combination of employees encountered in the sales. Now we can add a single foreign key DimSalesTeamKey to the fact table. Separately, we create an DimEmployee dimension that includes all sales employees. Finally, the BridgeSalesTeam table will contain the mappings of employees (from DimEmployee) to each group (DimSalesTeam).

*Design DownloadSalesTeam and DownloadEmployee*

We will create download tables, called DownloadSalesTeam and DownloadEmployee in WizStaging to facilitate the downloading of the “SalesTeam” and "Employee" table data from the Source DB. We will use the same method as for DownloadCustomer on p.13.

Right-click the SalesTeam table under DWizSource and select Copy Table to define the new DownloadSalesTeam. Fill in the definition page, being certain to select the ETL group “TutorialGroup” and to specify the DWizSource Connection. Make sure that column ID is indicated as the Source Primary Key with a "1", then press the Save icon (Floppy-Disk).

Repeat these steps for DownloadEmployee: Right-click the Employee table under DWizSource and select Copy Table to define the new DownloadEmployee. In the definition page, select the ETL group “TutorialGroup”, specify the DWizSource Connection, indicate EmployeeID as the Source Primary Key with a "1", and press Save.
**Design FlatSalesTeam and FlatEmployee**

We will create Flat tables in WizStaging by using the same method as for FlatCustomer on p.16. Right-click the DownloadSalesTeam table under **DWizStaging** and select Copy Table to define the new FlatSalesTeam with Table Type = **Flat**:

![Copy Table](image)

In the definition page, indicate ID as the Source Primary Key with a "1", and press Save.

Do the same for FlatEmployee: Right-click the DownloadEmployee table under **DWizStaging** and select Copy Table to define the new FlatEmployee with Table type = Flat. In the definition page, indicate EmployeeID as the Source Primary Key with a "1", and press Save.

**Design DimEmployee**

We will create a DimEmployee table in DWizWarehouse by using the same method as for DimCustomer on p.18. Right-click the FlatEmployee table under **DWizStaging** and select Copy Table to define the new DimEmployee with:

- Database Name = DWizWarehouse
- Schema = dbo
- Table type = Dim
- Table Name = DimEmployee

In the definition page, indicate EmployeeID as the Natural Key with a "1", and press Save.

**Define Bridge Table BridgeSalesTeam**

Now we are ready to define the Bridge table proper, BridgeSalesTeam, in the Warehouse DB. Right-click the FlatSalesTeam table under DWizStaging->Tables->Flat, and select Copy Table:
Illustration 54: "Copy Table" Pop-Up for Defining a Bridge Table

Make sure to set:
- Database Name = DWizWarehouse
- Schema = dbo
- Table Type = Bridge
- Table Name = BridgeSalesTeam

In the BridgeSalesTeam definition page, append two new columns at the bottom, BridgeSalesTeamKey and DimEmployeeKey. Also indicate "ID" as the Natural Key with a "1", then Save:

Illustration 55: "Bridge Table" Definition Page

Don't forget to Save the definition page! Note that DWiz silently creates the primary key, BridgeSalesTeamID.
**Add New Column BridgeSalesTeamKey to FactInvoice**

In Object Explorer, double-click FactInvoice (DWizWarehouse->Tables->Fact) to open it. In the definition page, append column BridgeSalesTeamKey (int); then press Save at the top of the definition page. With the tables all set, we are ready to define the stored procedures to load them.

**Procedure for FlatSalesTeam**

In a similar fashion to the steps on p.25, we will create an ETL stored procedure for FlatSalesTeam. Perform the following steps:

- Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.
- Create Flat Definition Page with Stored Procedure Name = FlatSalesTeam_upd, Source Table = DownloadSalesTeam, and Flat Table = FlatSalesTeam. Then press OK.
- In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column ID, and then press the Save icon.
- Press the Create/Alter Procedure button (Lightning/+ icon).
- After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

**Procedure for FlatEmployee**

In a similar fashion to the steps on p.25, we will create an ETL stored procedure for FlatEmployee. Perform the following steps:

- Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.
- Create Flat Definition Page with Stored Procedure Name = FlatEmployee_upd, Source Table = DownloadEmployee, and Flat Table = FlatEmployee. Then press OK.
- In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column EmployeeID, and then press the Save icon.
- Press the Create/Alter Procedure button (Lightning/+ icon).
- After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

**Procedure for DimEmployee**

In a similar fashion to the steps on p.17, we will create an ETL stored procedure for DimEmployee. Perform the following steps:

- Expand DWizWarehouse-->Stored Procedures-->Dim, right-click the Dim folder, and select New Dim.
- Create Dim Definition Page with Stored Procedure Name = DimEmployee_upd, Source Table = FlatEmployee, and Dim Table = DimEmployee. Then press OK.
• In the Dim Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column EmployeeID, and then press the Save icon.

• It is not necessary to compile the procedure (with the Create/Alter Procedure button) just yet. We will compile each of the warehouse procedures (Fact, Bridge, and Dim) only after all three have been defined and SAVED. The reason for this is that DWiz needs the definition info from all three before writing the code for any of them.

**Procedure for BridgeSalesTeam**

We will create an ETL stored procedure for BridgeSalesTeam. As Bridge tables reside in the Warehouse, our stored procedure will reside there also. Perform the following steps:

• Expand **DWizWarehouse** --> Stored Procedures --> Bridge, right-click the Bridge folder, and select New Bridge.

• Create Bridge Definition Page with Stored Procedure Name = BridgeSalesTeam_upd, Source Table = FlatSalesTeam, and Bridge Table = BridgeSalesTeam, and Dim Table = DimEmployee. Then press OK:

  ![Create Bridge Definition Page](image)

  *Illustration 56: "Create Bridge Definition Page" Pop-Up*

  In the Bridge Definition Page that appears in the page area, perform the following steps:

  • Select SCD = Type1
  • Select ETL Group = TutorialGroup
  • Select Bridge Group Key = BridgeSalesTeamKey
  • Select Bridge Table Key = DimEmployeeKey
  • Select Dim Table Key = DimEmployeeKey
  • In the Staging to Bridge Mapping, enter a "1" in the Natural Key box of column ID
  • In the Bridge to Dim Mapping, enter a "1" in the Natural Key box of column EmployeeID
  • Press the Save icon.
It is not necessary to compile the procedure (with the Create/Alter Procedure button) just yet. We will compile each of the warehouse procedures (Fact, Bridge, and Dim) only after all three have been defined and SAVED. The reason for this is that DWiz needs the definition info from all three before writing the code for any of them.

**Add New Column BridgeSalesTeamKey to FactInvoice_upd**

In Object Explorer, double-click FactInvoice_upd (DWizWarehouse-->Stored Procedures-->Fact) to open it. In the top Tab Button Bar, press Add Bridge:
In the resulting pop-up, select BridgeSalesTeam:

![Image of Add Bridge Table pop-up]

Illustration 58: "Add Bridge Table" Pop-Up

In the definition page, scroll down to reveal the new section at the bottom, "Fact To Bridge", and perform the following steps:

- Select Fact Table Key = BridgeSalesTeamKey
- Select Bridge Table Key = BridgeSalesTeamKey
- Put a "1" in the Natural Key of column SalesTeamID. This section should look like the following figure:

![Image of Fact To Bridge section]

Illustration 59: Scrolling Down to View the Fact-To-Bridge Section

Now press Save (Floppy-Disk) to re-save the modified page. Next we will generate the stored procedure by clicking the Create/Alter Procedure button (Lightning icon). After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).
Now return to the BridgeSalesTeam_Upd procedure page, and compile it by pressing the Create/Alter Procedure button, followed by the Save Procedure button. Then do the same for the DimEmployee_Upd procedure. When designing a bridge, it is important to compile/recompile all warehouse procedures (Fact Update, Bridge Update, and Dim Update) after all three have been defined and SAVED. So first define and save each of these procedures, then go back and compile each of them with the Create/Alter Procedure button followed by the Save Procedure button. This is necessary because DWiz needs the definition info of all three before it can correctly write code for any of them.

Recompile ETL Package

With all the tables and procedures done, we can now re-create the ETL Package, in a similar fashion to the steps on p.33. Our choices are 1) to compile an ETL Group Procedure, or 2) to compile an ETL SSIS Package. The first option is simple, just open the ETL Definition Page and compile the ETL Group Procedure. The second option (higher performance) entails the following steps:

- Expand **DWizStaging**-->ETL-->Definitions, right-click the Definitions folder, and double-click on TutorialETL
- Review ETL Definition Page (no changes are necessary), and press the Save button
- Press the Create BIML button (Scroll+Pen icon on top right)
- Open your TutorialETL project in SQL Data Tools for Visual Studio, and double-click the BimlScript.biml file to open it
- Copy (cut-and-paste clipboard) the new BIML from DWiz to overwrite the BimlScript in Visual Studio
- Save All in Visual Studio
- Right-click BimlScript.biml (under Miscellaneous in Solution Explorer) and click "Generate SSIS Packages". This may take a few minutes. Visual Studio may ask if you want to overwrite the package TutorialETL.dtsx; click on Yes.
- Afterward, you may view the data flow by double-clicking on the TutorialETL.dtsx package
- You may run the ETL Package by pressing the Start button on the top menu. Afterward, you can see the filled warehouse tables in SSMS.

This concludes our third iteration of the tutorial warehouse, by adding a Bridge Dimension to an existing warehouse. The SSIS package TutorialETL.dtsx could be run on a periodic schedule to maintain the warehouse. The augmented warehouse is "finished" and usable. In the next chapter, we will add an Outrigger Dimension.
In this chapter, we will augment the existing design, from the previous chapters, by adding an Outrigger Dimension.

An Outrigger Dimension links to another Dimension instead of (or in addition to) a Fact table. Outriggers are particularly useful when you want to search one of your dimensions, such as Customers, by a category such as Demographics. Henceforth you can add a foreign key (DimDemographyKey) to Customers, which points to an Outrigger table (DimDemography), through which you can then do fast searching of the Customers table. In this chapter, we will augment the existing design from the previous chapters, by adding an Outrigger Dimension.

**Design DownloadDemography**

We will create a download table, called DownloadDemography in WizStaging to facilitate the downloading of the “Demography” table data from the Source DB. We will use the same method as for DownloadCustomer on p.13.

Right-click the Demography table under DWizSource and select Copy Table to define the new DownloadDemography. Fill in the definition page, being certain to select the ETL group “TutorialGroup” and to specify the DWizSource Connection. Make sure that column DemographyID is indicated as the Source Primary Key with a "1", then press the Save icon (Floppy-Disk).

**Design FlatDemography**

We will create a Flat table in WizStaging by using the same method as for FlatCustomer on p.16. Right-click the DownloadDemography table under DWizStaging and select Copy Table to define the new FlatDemography with Table Type = Flat.

In the definition page, indicate DemographyID as the Source Primary Key with a "1", and press Save.

**Design DimDemography**

We will create a DimDemography table in DWizWarehouse by using the same method as for DimCustomer on p.18. Right-click the FlatDemography table under DWizStaging and select Copy Table to define the new DimDemography with:

- Database Name = DWizWarehouse
- Schema = dbo
• Table type = Dim
• Table Name = DimDemography

In the definition page, indicate DemographyID as the Natural Key with a "1", and press Save. Notice that DWiz silently adds surrogate primary key DimDemographyKey.

**Add New Column DimDemographyKey to DimCustomer**

In Object Explorer, double-click DimCustomer (DWizWarehouse->Tables->Dim) to open it. In the definition page, append column DimDemographyKey (int); then press Save at the top of the definition page. With the tables all set, we are ready to define the stored procedures to load them.

**Procedure for FlatDemography**

In a similar fashion to the steps on p.25, we will create an ETL stored procedure for FlatDemography. Perform the following steps:

• Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.

• Create Flat Definition Page with Stored Procedure Name = FlatDemography_upd, Source Table = DownloadDemography and Flat Table = FlatDemography. Then press OK.

• In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column DemographyID, and then press the Save icon.

• Press the Create/Alter Procedure button (Lightning/+ icon).

• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).
Add New Outrigger Relationship to DimCustomer_upd

In Object Explorer, double-click DimCustomer_upd (DWizWarehouse-->Stored Procedures-->Dim) to open it. Make sure that there is a "1" in the Natural Key column of CustomerId. In the top Tab Button Bar, press Add Outrigger:

Illustration 60: Adding an Outrigger Table to a Dim Table Procedure

In the resulting pop-up, select DimDemography:

Illustration 61: "Add Outrigger Table" Pop-Up

In the definition page, scroll down to reveal the new section at the bottom, "Dim To Outrigger", and perform the following steps:

- Select Dim Table Key = DimDemographyKey
- Select Outrigger Table Key = DimDemographyKey
- Put a "1" in the Natural Key of column DemographyID. This section should look like
Now press Save (Floppy-Disk) to re-save the modified page. Next we will generate the stored procedure by pressing the Create/Alter Procedure button. After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

**Procedure for DimDemography**

We will make a stored procedure for the new DimDemography. The Dim is part of the warehouse DB, so the procedure will reside there as well. Perform the following steps to create an ETL stored procedure for DimDemography:

- Expand **DWizWarehouse**-->Stored Procedures-->**Dim**, right-click the Dim folder, and select New Dim.

- Create Dim Definition Page with Stored Procedure Name = DimDemography_upd, Source Table = FlatDemography, and Dim Table = DimDemography. Then press OK.

- In the Dim Definition Page, select ETL Group = TutorialGroup, and put a "1" in the Natural Key column of DemographyId. Press the **Save** icon.

- Press the Create/Alter Procedure button (Lightning/+ icon).

- After the code is displayed, press the Save Procedure button (Lightning over Floppy icon).

**Recompile ETL Package**
With all the tables and procedures done, we can now re-create the ETL Package, in a similar fashion to the steps on p.33. Our choices are 1) to compile an ETL Group Procedure, or 2) to compile an ETL SSIS Package. The first option is simple, just open the ETL Definition Page and compile the ETL Group Procedure. The second option (higher performance) entails the following steps:

- Expand **DWizStaging-->ETL-->Definitions**, right-click the Definitions folder, and double-click on TutorialETL
- Review ETL Definition Page (no changes are necessary), and press the Save button
- Press the Create BIML button (Scroll+Pen icon on top right)
- Open your TutorialETL project in SQL Data Tools for Visual Studio, and double-click the BimlScript.biml file to open it
- Copy (cut-and-paste clipboard) the new BIML from DWiz to overwrite the BimlScript in Visual Studio
- Save All in Visual Studio
- Right-click BimlScript.biml (under Miscellaneous in Solution Explorer) and click "Generate SSIS Packages". This may take a few minutes. Visual Studio may ask if you want to overwrite the package TutorialETL.dtsx; click on Yes.
- Afterward, you may view the data flow by double-clicking on the TutorialETL.dtsx package
- You may run the ETL Package by pressing the Start button on the top menu. Afterward, you can see the filled warehouse tables in SSMS.

This concludes our fourth iteration of the tutorial warehouse, by adding an Outrigger Dimension to an existing warehouse. The SSIS package TutorialETL.dtsx could be run on a periodic schedule to maintain the warehouse. The augmented warehouse is "finished" and usable. In the next chapter, we will demonstrate a Pivot table.
11. Pivot Tables

Pivot Tables are a special type of Flat table, wherein certain data from rows in the source table (Download) is pulled out to become separate columns in the resulting pivot (flat) table. The Pivot table can then be used to provide data for warehouse tables such as Dimensions. For example, in the Tutorial Source DB, we have a Phone table that may have multiple records for a given customer, with each record labeled as "HomePhone", "CellPhone", or "OfficePhone". In keeping with best practices of data warehousing, we want our Customer Dimension to have only one row per distinct customer, and for that one row to have columns of "HomePhone", "CellPhone", and "OfficePhone". In this example, we will accomplish this desired result easily using DWiz's Pivot Table feature.

Design DownloadPhone

We will create a download table, called DownloadPhone in WizStaging to facilitate the downloading of the “Phone” table data from the Source DB. We will use the same method as for DownloadCustomer on p.13.

Right-click the Phone table under DWizSource and select Copy Table to define the new DownloadPhone. Fill in the definition page, being certain to select the ETL group “TutorialGroup” and to specify the DWizSource Connection. Make sure that column PhoneID is indicated as the Source Primary Key with a "1", then press the Save icon (Floppy-Disk). Note that the Phone table has columns "PhoneNumber" and "PhoneType"--this latter column will specify whether the phone is Home, Cell, or Office.

Design FlatPhone

Create a Flat table in WizStaging by using the same method as for FlatCustomer on p.16. Right-click the DownloadPhone table under DWizStaging and select Copy Table to define the new FlatPhone with Table Type = Flat.

In the definition page, indicate PhoneID as the Source Primary Key with a "1", and press Save.

Design Pivot Table PivotPhoneWithExt

Create the Pivot table in WizStaging as follows. Right-click the FlatPhone table under DWizStaging and select Copy Table to define the new PivotPhoneWithExt with Table Type = Pivot.

In the definition page, add and delete columns so that the resulting table has only:

- CustomerId int, Allow Nulls = No, Primary Key = 1
- HomePhone varchar(10), Allow Nulls = Yes
• CellPhone varchar(10), Allow Nulls = Yes
• OfficePhone varchar(10), Allow Nulls = Yes
• OfficeExtension varchar(4), Allow Nulls = Yes

Note that our primary key will now be CustomerID, not PhoneId. After the pivot, we will have one row per customerid. Be sure to indicate CustomerID as the Primary Key with a "1", and press Save:

Illustration 63: "Pivot Table" Definition Page

Add New Columns to FlatCustomer and DimCustomer

In Object Explorer, double-click DimCustomer (DWizWarehouse->Tables->Dim) to open it. In the definition page, append the following column:

* HomePhone varchar(10), Allow Nulls = Yes
* CellPhone varchar(10), Allow Nulls = Yes
* OfficePhone varchar(10), Allow Nulls = Yes
* OfficeExtension varchar(4), Allow Nulls = Yes

Then press Save at the top of the definition page. Do the same for FlatCustomer (DWizStaging->Tables->Flat).

Procedure for FlatPhone

In a similar fashion to the steps on p.25, we will create an ETL stored procedure for FlatPhone. Perform the following steps:

• Expand DWizStaging-->Stored Procedures-->Flat, right-click the Flat folder, and select New Flat.
• Create Flat Definition Page with Stored Procedure Name = FlatPhone_upd, Source Table = DownloadPhone and Flat Table = FlatPhone. Then press OK.

• In the Flat Definition Page, select ETL Group = TutorialGroup, enter a "1" in the Natural Key box of column PhoneID, and then press the Save icon.

• Press the Create/Alter Procedure button (Lightning/+ icon).

• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

**Procedure for PivotPhoneWithExt**

Here is where the actual pivoting happens. We will make a stored procedure for the new PivotPhoneWithExt. The Pivot is part of the Staging DB, so the procedure will reside there as well. Perform the following steps to create an ETL stored procedure for PivotPhoneWithExt:

• Expand **DWizStaging**--*>(Stored Procedures--->**Pivot**, right-click the Pivot folder, and select New Pivot.

• Create Pivot Definition Page with Stored Procedure Name = PivotPhoneWithExt_upd, Source Table = FlatPhone, and Pivot Table = PivotPhoneWithExt, then press OK.

• In the Pivot Definition Page, select ETL Group = TutorialGroup, Source Natural Key = PhoneID, Aggregate Function = max, Pivot-On Column = PhoneType, and put a "1" in the Natural Key column of CustomerId. Then enter the Pivot-On Values as shown in the following illustration. The Pivot-On values determine when the source PhoneNumber is entered into a destination column; for example, when the source PhoneType = "Home", then the source PhoneNumber will be entered into the HomePhone column. Press the Save icon.

• Press the Create/Alter Procedure button (Lightning/+ icon).

• After the code is displayed, press the Save Procedure button (Lightning over Floppy icon).
Update Procedure for FlatCustomer_upd

Now we can use the new pivot table by updating the existing ETL procedures to fill the new columns in FlatCustomer and DimCustomer. Double-click the FlatCustomer proc DWizStaging-->Stored Procedures-->Flat-->FlatCustomer_upd. Above the Flat Definition page, click the Additional Source button and select "PivotPhoneWithExt".
Illustration 65: Adding a Pivot Table to a Flat Update Procedure

Then scroll down in the page to view the new Additional Source section. This section should look like the following illustration:
• Ensure that there is a "1" in the Natural Key column of CustomerId.
• Press the **Save** icon.
• Press the Create/Alter Procedure button (Lightning/+ icon).
• After the code is displayed, press the Save Procedure button (Lightning Over Floppy icon).

**Update the Procedure for DimCustomer_upd**

Now we can update the DimCustomer_Upd procedure to fill the new columns. Double-click the DimCustomer proc  DWizWarehouse-->Stored Procedures--> Dim-->DimCustomer_upd. Verify that the new columns have been sensed (HomePhone, CellPhone, OfficePhone, OfficeExtension), then press the Save icon. Next press the Create/Alter Procedure button and finally the Save Procedure button.
Recompile ETL Package

With all the tables and procedures done, we can now re-create the ETL Package, in a similar fashion to the steps on p.33. Our choices are 1) to compile an ETL Group Procedure, or 2) to compile an ETL SSIS Package. The first option is simple, just open the ETL Definition Page and compile the ETL Group Procedure. The second option (higher performance) entails the following steps:

• Expand **DWizStaging** --> ETL --> Definitions, right-click the Definitions folder, and double-click on TutorialETL

• Review ETL Definition Page (no changes are necessary), and press the Save button

• Press the Create BIML button (Scroll+Pen icon on top right)

• Open your TutorialETL project in SQL Data Tools for Visual Studio, and double-click the BimlScript.biml file to open it

• Copy (cut-and-paste clipboard) the new BIML from DWiz to overwrite the BimlScript in Visual Studio

• Save All in Visual Studio

• Right-click BimlScript.biml (under Miscellaneous in Solution Explorer) and click "Generate SSIS Packages". This may take a few minutes. Visual Studio may ask if you
want to overwrite the package TutorialETL.dtsx; click on Yes.

- Afterward, you may view the data flow by double-clicking on the TutorialETL.dtsx package

- You may run the ETL Package by pressing the Start button on the top menu. Afterward, you can see the filled warehouse tables in SSMS.

This concludes our example of adding a Pivot Table to an existing warehouse. The augmented warehouse is "finished" and usable.

**Alphabetical Index**

A........................................................................................................................................... 52p., 73p.
Additional Source.................................................................................................................... 72
Aggregate................................................................................................................................. 70p.
Allow Nulls............................................................................................................................... 70p.
B.................................................................................................................................................. 57
biml........................................................................................................................................... 35, 40pp., 44pp., 56, 64, 69, 76
Bridge Table.............................................................................................................................. 57
C.................................................................................................................................................. 57
Change Data Capture............................................................................................................... 36, 38
Change Tracking...................................................................................................................... 36, 38
Connect to the Databases........................................................................................................ 9
connection string..................................................................................................................... 37, 39
Create BIML button............................................................................................................... 40, 56, 64, 69, 76
Create/Alter Procedure button.............................................................................................. 29
D................................................................................................................................................ 35, 38
default value.......................................................................................................................... 15, 18, 20
Define Additional Source button.......................................................................................... 53
Dim Definition.......................................................................................................................... 20, 54, 60p., 68
Dimension Table................................................................................................................... 14, 19, 20, 22, 23p., 26, 48, 57, 65
download method................................................................................................................... 36, 38
Download Table..................................................................................................................... 48, 57, 65, 70
E................................................................................................................................................ 35, 38
ETL Group............................................................................................................................... 13, 35pp., 48, 51, 54, 56p., 60p., 64pp., 68pp., 72, 76
ETL Group Procedure........................................................................................................... 35, 37, 56, 64, 69, 76
ETL Package........................................................................................................................... 35, 47, 56, 64, 68p., 76p.
ETL SSIS Package.................................................................................................................. 35, 37p., 56, 64, 69, 76
ETL stored procedure............................................................................................................. 50p., 54, 60p., 66, 68, 71p.
Excel....................................................................................................................................... 36, 39
F............................................................................................................................................ 22, 34, 48, 55, 57, 63, 65
fact table............................................................................................................................... 48
Fact To Dim section............................................................................................................... 34
**Illustration Index**

<table>
<thead>
<tr>
<th>Illustration</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DataWarehouseWiz Control Areas</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Creating Databases in Microsoft Sql Server Management Studio</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Connecting to Staging &amp; Warehouse DBs</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Connect-To-Database Pop-Up</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>OneTime Initialization of DWiz Metadata</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Installing the Tutorial Tables</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Connect-To-Source-DB Pop-Up for the Tutorial Source DB, DWizSource</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>Creating an ETL Group</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>Save Page (Floppy-Disk Icon Button)</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>Copying a Table's Schema</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>&quot;Copy Table&quot; Pop-Up</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>Download-Table Definition Page</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>DownloadInvoice Definition Page</td>
<td>17</td>
</tr>
<tr>
<td>14</td>
<td>&quot;Copy Table&quot; Pop-Up for Creating Flat Tables</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>&quot;Flat Table&quot; Definition Page</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>FlatInvoice Table Definition Page</td>
<td>19</td>
</tr>
<tr>
<td>17</td>
<td>&quot;Copy Table&quot; Pop-Up for Dimension Tables</td>
<td>19</td>
</tr>
<tr>
<td>18</td>
<td>&quot;Dimension Table&quot; Definition Page</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>Completing a Dimension-Table Definition Page Form</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>DWiz Automatically Creates Surrogate Keys</td>
<td>21</td>
</tr>
<tr>
<td>21</td>
<td>&quot;Copy Table&quot; Pop-up for DimInvoice</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>DimInvoice Table Definition Page</td>
<td>23</td>
</tr>
<tr>
<td>23</td>
<td>DimInvoice Table Definition Page Completed</td>
<td>24</td>
</tr>
<tr>
<td>24</td>
<td>&quot;Copy Table&quot; Pop-Up for Fact-Tables</td>
<td>25</td>
</tr>
<tr>
<td>25</td>
<td>&quot;Fact Table&quot; Definition Page</td>
<td>26</td>
</tr>
<tr>
<td>26</td>
<td>Creating a Fact-Table Stored Procedure</td>
<td>28</td>
</tr>
<tr>
<td>27</td>
<td>Pop-Up for Creating a Flat-Table Stored Procedure</td>
<td>28</td>
</tr>
<tr>
<td>28</td>
<td>Flat-Table Stored Procedure Definition Page</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>DWiz Creates a Flat-Table Stored Procedure From Your Form Input</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>Reviewing DWiz's Stored Procedure Code Prior to Compiling It</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>Adding a Dimension-Table Relationship to a Fact-Table Procedure</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>&quot;Add Dim Table&quot; Pop-Up for Fact-Table Procedure</td>
<td>33</td>
</tr>
<tr>
<td>33</td>
<td>Scrolling Down in the Fact-Procedure Page to View the Dim Section</td>
<td>34</td>
</tr>
<tr>
<td>34</td>
<td>ETL Definition Page</td>
<td>36</td>
</tr>
<tr>
<td>35</td>
<td>Choosing a Download Method in the ETL Definition Page</td>
<td>37</td>
</tr>
<tr>
<td>36</td>
<td>Selecting Connections in the ETL Definition Page</td>
<td>38</td>
</tr>
<tr>
<td>37</td>
<td>Re-Save the ETL Definition Page After Changing a Connection</td>
<td>39</td>
</tr>
<tr>
<td>38</td>
<td>&quot;Create BIML&quot; Button on the ETL Definition Page</td>
<td>40</td>
</tr>
<tr>
<td>39</td>
<td>Reviewing BIML Code Written By DWiz</td>
<td>41</td>
</tr>
<tr>
<td>40</td>
<td>Creating an SSIS Package in Visual Studio</td>
<td>42</td>
</tr>
<tr>
<td>41</td>
<td>Creating an Integrated Services Project in Visual Studio</td>
<td>43</td>
</tr>
<tr>
<td>42</td>
<td>Adding the BIML to a Visual Studio Project</td>
<td>44</td>
</tr>
</tbody>
</table>

**DataWarehouseWiz.com**
Illustration 43: Opening the Default BIML File.................................................................45
Illustration 44: Saving Your BIML in the Visual Studio Project........................................46
Illustration 45: Generating an SSIS Package.................................................................47
Illustration 46: "Copy Table" Pop-up for FlatInvoiceStatus...........................................49
Illustration 47: "Copy Table" Pop-Up for Defining Junk Tables........................................49
Illustration 48: "Junk Table" Definition Page.....................................................................50
Illustration 49: Revising a Flat Update Stored Procedure..............................................52
Illustration 50: "Add Addition Source" Pop-Up for a Flat Update Stored Procedure...........52
Illustration 51: Create or Alter a Stored Procedure.........................................................53
Illustration 52: Adding a Junk Table to a Fact Table Update Procedure.........................55
Illustration 53: "Add Junk Table" Pop-Up.........................................................................55
Illustration 54: "Copy Table" Pop-Up for Defining a Bridge Table.................................59
Illustration 55: "Bridge Table" Definition Page...............................................................59
Illustration 56: "Create Bridge Definition Page" Pop-Up..................................................61
Illustration 57: Adding a Bridge Table to a Fact Update Procedure..................................62
Illustration 58: "Add Bridge Table" Pop-Up.....................................................................63
Illustration 59: Scrolling Down to View the Fact-To-Bridge Section.................................63
Illustration 60: Adding an Outrigger Table to a Dim Table Procedure.............................67
Illustration 61: "Add Outrigger Table" Pop-Up.................................................................67
Illustration 62: Scrolling Down to View the "Dim To Outrigger" Section............................68
Illustration 63: "Pivot Table" Definition Page.................................................................71
Illustration 64: Defining Pivot-On Values in the Pivot Update Procedure.........................73
Illustration 65: Adding a Pivot Table to a Flat Update Procedure....................................74
Illustration 66: Scrolling Down to View the Additional Source Section............................75
Illustration 67: Using the Pivot Table Results in the DimCustomer_upd Procedure...........76

**Online Video Index**

- Online Video: Starting DWiz for the First Time..........................................................7
- Online Video: Creating a New Project...........................................................................12
- Online Video: Designing Download Tables.................................................................14
- Online Video: Designing Flat Tables............................................................................17
- Online Video: Designing Dimension Tables.................................................................19
- Online Video: Designing Fact Tables..........................................................................25
- Online Video: Designing Flat Procedures.....................................................................27
- Online Video: Designing Dimension Procedures.........................................................32
- Online Video: Designing Fact Procedures....................................................................32
- Online Video: Compiling an ETL Group......................................................................35
- Online Video: Compiling a BIML/SSIS Package.........................................................37
- Online Video: Adding A Junk Dimension.....................................................................48
- Online Video: Adding a Bridge Dimension....................................................................57
- Online Video: Adding An Outrigger Dimension...........................................................65
- Online Video: Pivoting.................................................................................................70

**Bibliography**

*DataWarehouseWiz.com* 80

###