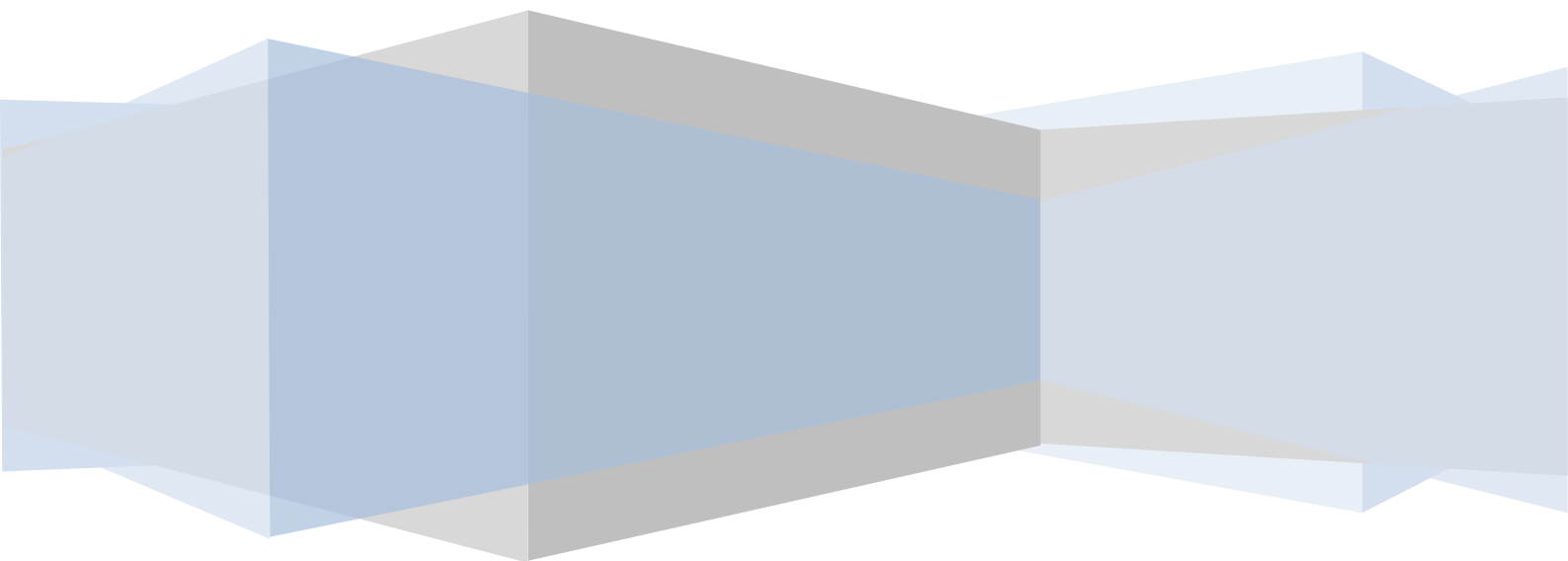


**Data Info Leaders**

# **DW Architect**

**Architecture**



# TABLE OF CONTENTS

INTRODUCTION .....	2
DW ARCHITECT – ARCHITECTURE DIAGRAM .....	3
DESIGNER .....	4
META DATA .....	4
SOURCE CONTROL AND MULTI TEAM MEMBER SUPPORT .....	5
SOURCE SYSTEM CONNECTIONS .....	5
META DATA IMPORT .....	6
RAPID ETL DESIGN WITH AUTO MATCHING .....	7
MAINTAINS LINEAGE .....	7
CUSTOM META DATA .....	7
CONFIGURATION FILES.....	8
GENERATION .....	9
DEPLOYMENT .....	11
BATCH EXECUTION .....	12
WORKFLOW .....	12
TASK EXECUTION PROVIDERS.....	13
CONFIGURATION .....	14
EXECUTING THE BATCH .....	14

## INTRODUCTION

'DW Architect' is designed to simplify building a Data Warehouse and its ETL. Out of the box, it comes with all the necessary patterns and code generators to create a Data Warehouse on the Microsoft SQL Server Platform. Other platform patterns and generators are in development.

'DW Architect' is also extensible and customisable, allowing Data Warehousing professionals to configure the tool to generate code using their own patterns and practices.

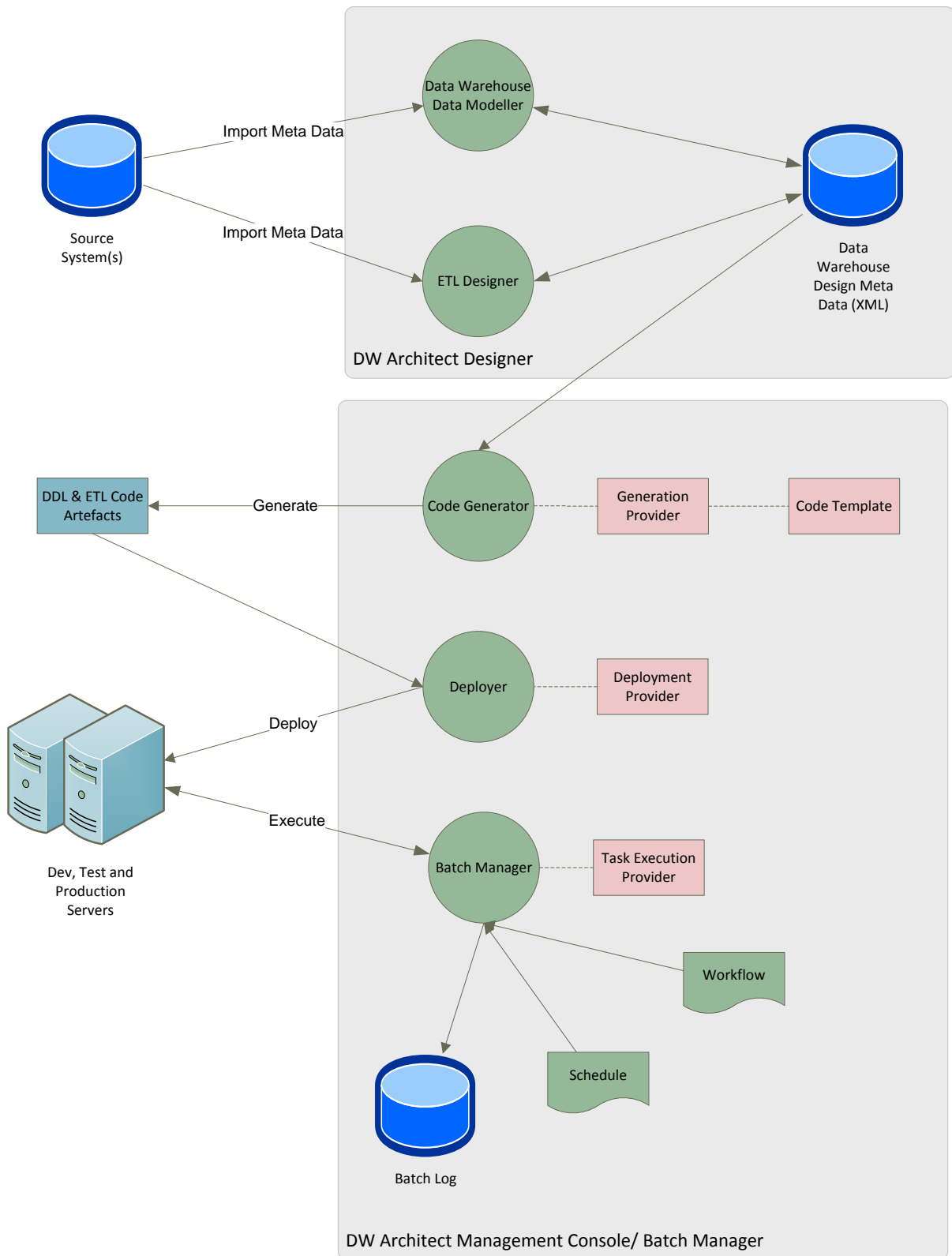
This document describes the architecture of 'DW Architect'.

"DW Architect" consist of 2 modules:

- DW Architect Designer. A Visual Studio plug-in custom project type.
- DW Architect Management Console. A separate application used to generate, deploy and execute the ETL Batch. The Management Console can be licensed separately for Test and Production environments.

The various components of DW Architect are shown in the diagram on the next page.

# DW ARCHITECT – ARCHITECTURE DIAGRAM



## DESIGNER

'DW Architect' Designer is an Integrated Design Environment for Data Warehousing. DW Architect Designer is a custom project type, that plugs into Microsoft Visual Studio. To create a new DW Architect project, you simply use the 'Create New Project' dialog of Visual Studio.

## META DATA

At its core DW Architect Designer is a Data Warehouse Meta Data Editor. The Meta Data describes the schema of the Data Warehouse, the Staging Database, Source Systems, and ETL. The Meta Data is represented as individual XML project files with custom XSD schema in the project.

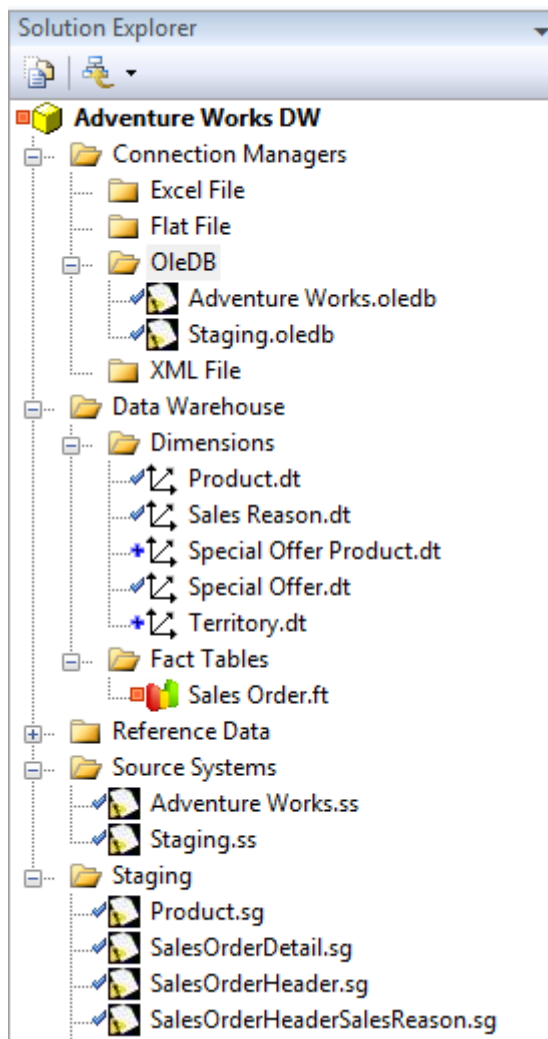


Figure 1 - Visual Studio - DW Architect Designer - Project Artefacts

Each type of document has its own editor.

```

<?xml version="1.0" standalone="yes"?>
<dsTT xmlns="http://tempuri.org/dsTT.xsd">
  <Target_Table>
    <Uid>c3872e66-78ad-4551-9032-8bdb9b9c270</Uid>
    <Table_Name>Product</Table_Name>
    <Cube_Name>Product</Cube_Name>
    <Load_Freq>cf67b1c2-8429-4ef0-a34e-c8234bbe576f</Load_Freq>
    <Type>Dimension</Type>
    <Description />
    <Table_Reference />
    <Custom_Surrogate_Key_Name_Indicator>False</Custom_Surrogate_Key_Name_Indicator>
    <Custom_Surrogate_Key_Name />
    <Custom_Surrogate_Key_Type_Indicator>False</Custom_Surrogate_Key_Type_Indicator>
    <Custom_Surrogate_Key_Type />
    <Surrogate_Key_Is_Auto_Generated_Indicator>True</Surrogate_Key_Is_Auto_Generated_Indicator>
    <Generate_DDL_Indicator>True</Generate_DDL_Indicator>
    <Conformed>False</Conformed>
    <Target_Columns>
      <Target_Column>
        <Uid>15aa4a92-e019-49f8-9546-a47273e41c2a</Uid>
        <Name>ProductID</Name>
        <Column_Type>Business Key</Column_Type>
        <Scd_Type />
        <Description />
        <Data_Type>int</Data_Type>
        <Merge_Replace_Reuse>False</Merge_Replace_Reuse>
        <Parent>False</Parent>
        <Column_Reference />
      </Target_Column>
      <Target_Column>
        <Uid>33081748-eeac-470b-80c3-a0624daa3f5e</Uid>
        <Name>Name</Name>

```

Figure 2 - Snippet from XML Meta Data for a Product Dimension.

## SOURCE CONTROL AND MULTI TEAM MEMBER SUPPORT

A DW Architect project is just like any other Visual Studio project. Storing the design in individual XML files makes it is compatible with source control systems, code merging and multi person teams.

## SOURCE SYSTEM CONNECTIONS

With DW Architect the design process starts with defining connections to Source Systems. Connections are used by DW to import Source System Meta Data for the purpose of defining Staging table, Fact Tables, Dimensions and Extract Procedures.

During code Generation and Deployment, the connection strings are overridden by alternative connection strings in a configuration file. A configuration file should be created for each environment you are deploying to, so the appropriate connection strings can be defined for each environment.

Various connection types are supported including Excel, Flat File (csv, tab delimited etc), XML, and database data sources that provide either an OLEDB or ODBC interfaces.

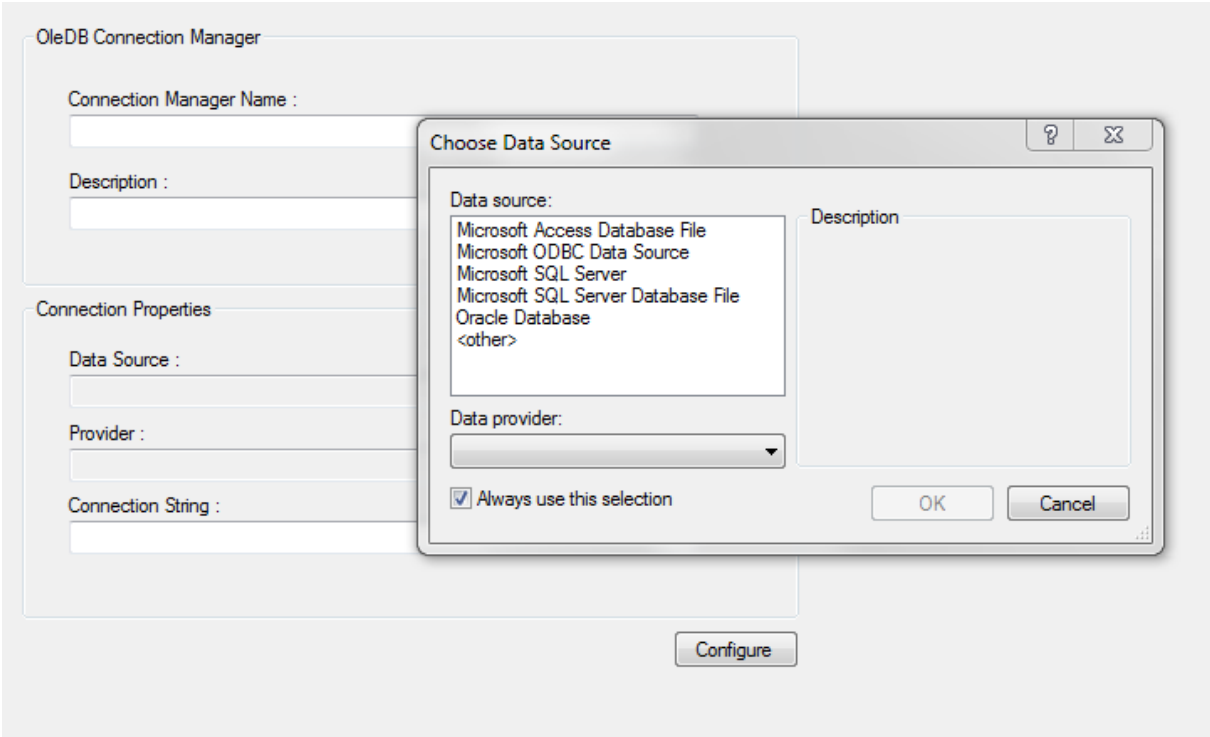


Figure 3 - An example of the OLEDB/ODBC connection configuration dialog.

## META DATA IMPORT

Using the Connections defined in the Project, DW Architect is able to connect to Source Systems, and retrieve Meta Data about Source System tables and columns. This Meta Data is imported to define Staging Table columns, Fact Table measures, Dimension attributes and Extract Procedures source tables. This feature makes defining these objects very quick, and accurate.

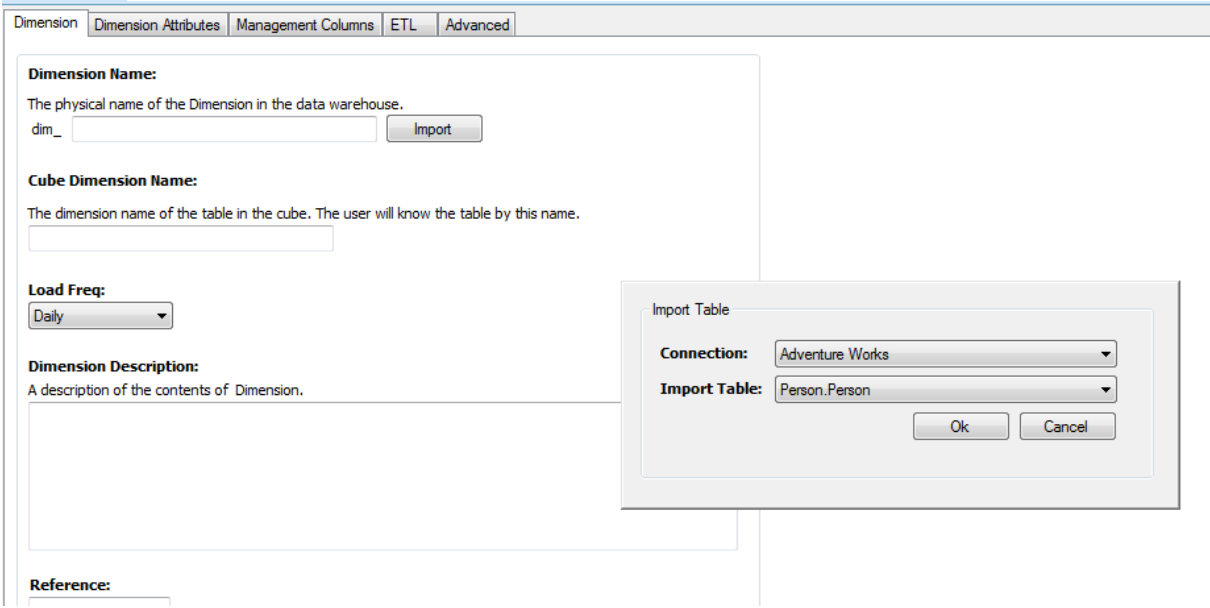
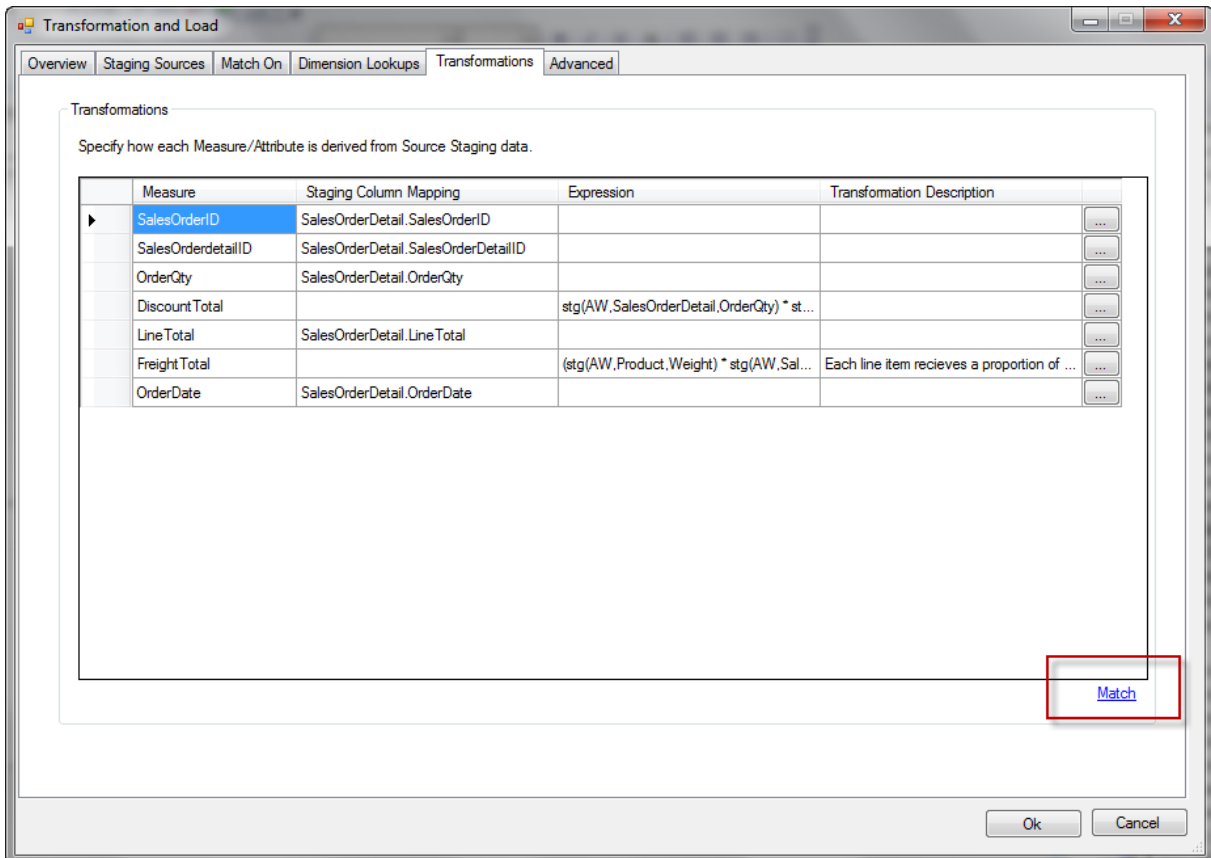


Figure 4 - Importing Attributes of a Dimension from the Adventure Works Source System, Person.Person table

## RAPID ETL DESIGN WITH AUTO MATCHING

The ETL Designer gives you the option to auto match source and destination column based on name. In the example below, on the transformations tab of the ETL designer for the Sales Order Fact table, the designer has the option to use the 'Match' option to make mappings between the Staging Source columns, and the Fact table Measures based on name. This option is also available on other ETL designer tabs.

This feature makes designing ETL, in most cases, very quick.



## MAINTAINS LINEAGE

All table and column references use IDs instead of Names in the Meta Data. This allows 'DW Architect' to keep the lineage of ETL intact if a table or column name or data type changes. For example, if you changed a Staging column's name, and regenerated the code, the name change would propagate through the generated Staging Table Create/Alter statement, Extract procedure, synonym, and Transform procedures.

## CUSTOM META DATA

Certain patterns, especially those defined by end users, require customized Meta Data to operate correctly. 'DW Architect' supports the creation of custom Meta Data in various places throughout the set of Meta Data Editors.

## CONFIGURATION FILES

The Generation, Deployment and Batch execution processes rely on a Configuration file. A Configuration file should exist for each target environment (Development, Test, Production etc.) A Configuration File specifies the following things:

- Data Warehouse Database Connection.
- Staging Database Connection.
- OLAP Database Connection.
- Source System Connections.
- Custom Configuration Meta Data.
- Generation, Deployment and Batch Execution provider paths.
- Code Generation output Path.
- Batch workflow path.
- Batch execution concurrent tasks.

These variables can be used in Generation, Deployment and Batch Execution for the given environment.

## GENERATION

The Generation process is managed by a Generation Service. The generation process is executed from the Management Console, or by command line.

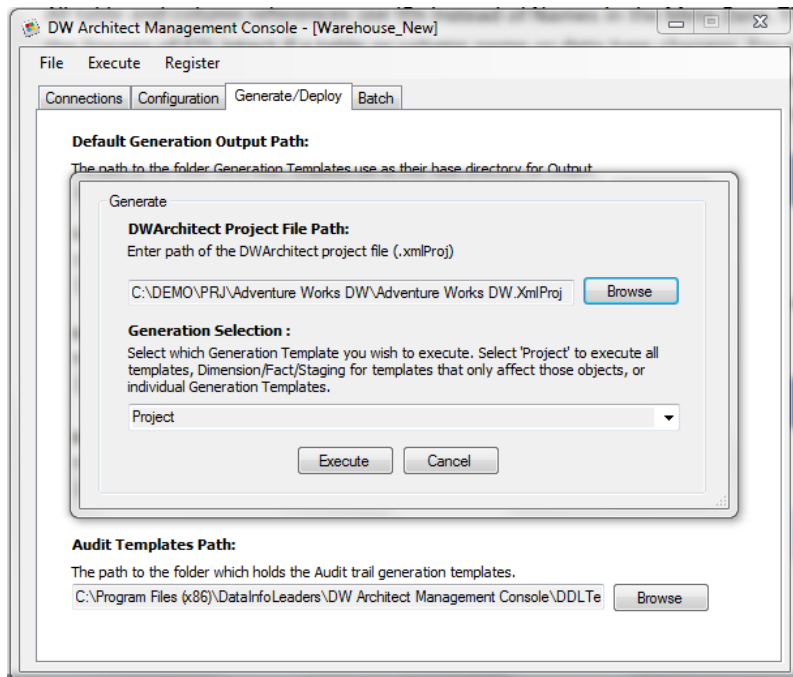


Figure 5 - Generating Code using the Management Console.

The Generation Service takes 2 inputs.

- The Project you wish to generate code for.
- What parts of the project you wish to generate.

The Generation Service relies on Generation Providers to operate. The providers are the engines that do the actual code generation. 'DW Architect' provides an API for Generation Providers, so Data Warehouse professionals can create their own Generation Providers. A single XSLT Generation Provider is shipped with DW Architect. This provider uses XSLT templates to generate code.

The Generation Service looks for Generation providers at the configurable Generation Provider path.

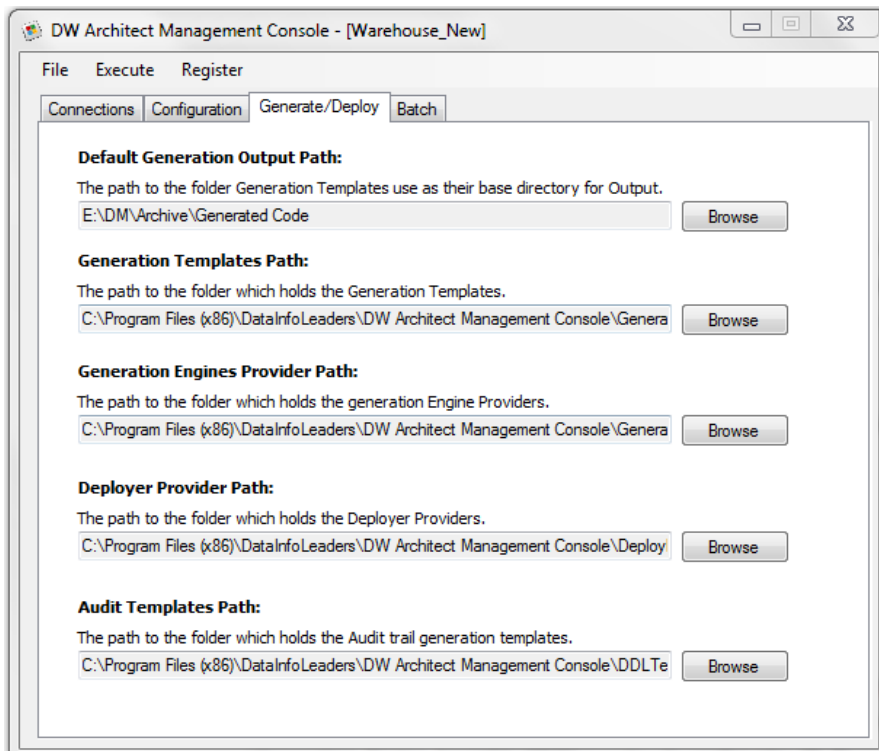


Figure 6 - Configuring Generation Provider and Template paths.

Generation provider are described by simple manifest files. E.g.

```
<?xml version="1.0" encoding="utf-8" ?>
<Generation_Engine_Manifest>
  <Generation_Engine_Name>XSLT Generation Engine</Generation_Engine_Name>
  <Generation_Engine_Class>com.datainfoleaders.deploy.XSLTGenerationEngineProvider</Generation_Engine_Class>
  <Generation_Engine_Assembly>XSLTGenerationEngineProvider.dll</Generation_Engine_Assembly>
</Generation_Engine_Manifest>
```

A Generation provider may use Generation Templates for generation. In the case of the XSLT generation provider, XSLT templates are defined for each type of code artefact that needs to be produced. Again Generation Templates are found by the Generation Service at the configurable Generation Template path and described by a simple manifest file. E.g.

```
<?xml version="1.0"?>
<Generation_Template_Manifest>
  <Output_Relative_Location>080 Procedures</Output_Relative_Location>
  <Template_Display_Name>Extract Procedures</Template_Display_Name>
  <Generation_Engine_Name> XSLT Generation Engine </Generation_Engine_Name>
  <Generation_Result_File_Name_Pattern>Extract_Procedures.sql</Generation_Result_File_Name_Pattern>
  <Template_Relative_Location>Extract_Procedures.enc</Template_Relative_Location>
  <OperatesOn>Staging</OperatesOn>
  <Generates_For_Collection>
    <Generates_For_TargetType>Staging</Generates_For_TargetType>
  </Generates_For_Collection>
</Generation_Template_Manifest>
```

The Generation Service passes what the user wants to generate to the generation providers it finds. The generation providers use details from Generation templates manifests to decide what Generation Templates it needs to execute, where to put the output, and the file name to give the output.

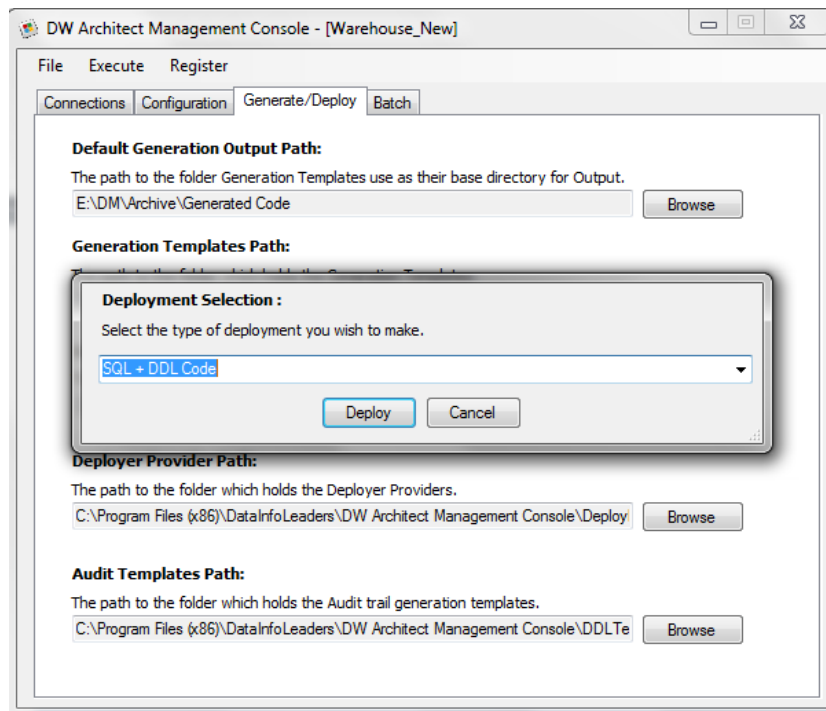
The set of XSLT templates shipped with DW Architect will generate all the code required to create/alter the Data Warehouse/Staging databases, and generate stored procedures for the ETL. Additional XSLT templates

can be created to generate other code, without needing to develop a custom Generation provider. Other XSLT Templates are in development, targeting other platforms like Oracle and IBM DB2.

Generation can also be triggered from the command line, for integration into automated build and deployment tools.

## DEPLOYMENT

The deployment process is managed by a Deployment Service. The deployment process is executed from the Management Console.



The Deployment Service takes 1 input.

- The type of code you wish to deploy.

The Deployment Service relies on Deployment Providers to operate. The providers are the engines that do the actual code Deployment. 'DW Architect' provides an API for Deployment Providers, so Data Warehouse professionals can create their own Deployment Providers. A single Microsoft SQL Server SQL Deployment Provider is shipped with DW Architect. Other deployment providers are in development, targeting other platforms like Oracle and IBM DB2.

The Deployment Service looks for Deployment providers at the configurable Deployment Provider path.

Deployment provider are described by simple manifest files. E.g.

```
<?xml version="1.0" encoding="utf-8" ?>
<Deployer_Manifest>
  <Deployer_Name>SQL Deployer Provider</Deployer_Name>
  <Deployer_Class>com.datainfoleaders.deploy.SQLDeployerProvider</Deployer_Class>
  <Deployer_Path>SQLDeployerProvider.dll</Deployer_Path>
  <Target_Type>SQL + DDL Code</Target_Type>
  <Target_Extension>.sql</Target_Extension>
</Deployer_Manifest>
```

The Deployment Service uses the user's selection to determine which deployment provider to invoke. The Deployment providers use details from the Deployment provider's manifest and other configuration details, like warehouse connection strings and the generated code output path, to find code to deploy that matches its target extension, and deploy it to the correct server.

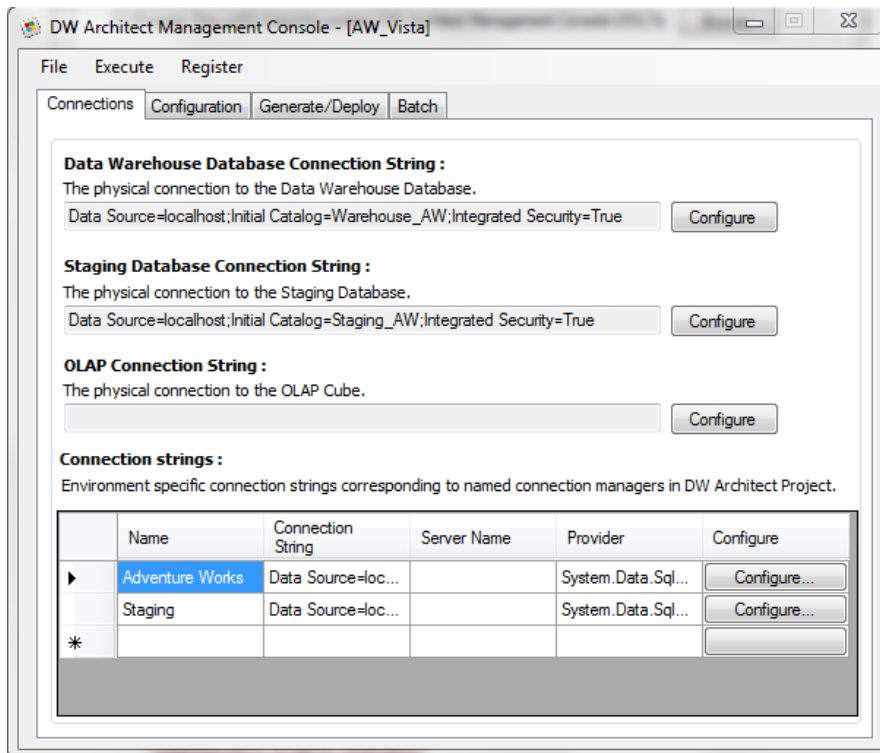


Figure 7 - Data Warehouse and Staging Database Connection Configuration.

Deployment can also be triggered from the command line, for integration into automated build and deployment tools.

## BATCH EXECUTION

For each of your production and Test environments you will want to install a server version of the Management Console so you can execute ETL Batches in these environments.

To run the Batch you need 5 things

1. DW Architect Management Console installed.
2. A configuration file, created through DW Architect Management Console.
3. Deployed Code.
4. A Batch Workflow file.
5. A .bat command file.

## WORKFLOW

A Batch workflow consists of an XML file with custom schema. An example of the schema is shown below. This example is used in our Tutorial and Example data warehouses.

```
<Batch_Schedule>
  <Batch_Type>Adventure Works Nightly</Batch_Type>
  <Schedule>
    <Phase>
      <Phase_Name>Extract</Phase_Name>
      <Phase_Provider>StoredProcExecutionProvider</Phase_Provider>
    </Phase>
  </Schedule>
</Batch_Schedule>
```

```

<Task>
  <Task_Name>Extract%</Task_Name>
  <Task_Provider>StoredProcExecutionProvider</Task_Provider>
</Task>
</Phase>
<Phase>
  <Phase_Name>Derived Extract</Phase_Name>
  <Phase_Provider>StoredProcExecutionProvider</Phase_Provider>
  <Task>
    <Task_Name>DerivedExtract%</Task_Name>
    <Task_Provider>StoredProcExecutionProvider</Task_Provider>
  </Task>
</Phase>
<Phase>
  <Phase_Name>Transform_Dim</Phase_Name>
  <Phase_Provider>StoredProcExecutionProvider</Phase_Provider>
  <Task>
    <Task_Name>Transform_Dim%</Task_Name>
    <Task_Provider>StoredProcExecutionProvider</Task_Provider>
  </Task>
</Phase>
<Phase>
  <Phase_Name>Transform_Updates</Phase_Name>
  <Phase_Provider>StoredProcExecutionProvider</Phase_Provider>
  <Task>
    <Task_Name>Transform_Update%</Task_Name>
    <Task_Provider>StoredProcExecutionProvider</Task_Provider>
  </Task>
</Phase>
<Phase>
  <Phase_Name>Transform_Inserts</Phase_Name>
  <Phase_Provider>StoredProcExecutionProvider</Phase_Provider>
  <Task>
    <Task_Name>Transform_Insert%</Task_Name>
    <Task_Provider>StoredProcExecutionProvider</Task_Provider>
  </Task>
</Phase>
<Phase>
  <Phase_Name>Table Management</Phase_Name>
  <Phase_Provider>StoredProcExecutionProvider</Phase_Provider>
  <Task>
    <Task_Name>Tbl_Mng%</Task_Name>
    <Task_Provider>StoredProcExecutionProvider</Task_Provider>
  </Task>
</Phase>
</Schedule>
</Batch_Schedule>

```

The Workflow contains a set of phases which in turn contain a set of tasks. Each phase executes in order, and each task in order within the phase. The task name can be wild carded.

## TASK EXECUTION PROVIDERS

Tasks are executed by Task execution Providers. A Task Execution providers is required for each Server Type/Platform that batch tasks execute on. The StoredProcExecutionProvider shipped with 'DW Architect' will execute stored procedures on Microsoft SQL Server. The name in the <Task\_Provider> element of a task must match the name in the Task Providers manifest file.

Each task execution provider should be written to interpret wild card task names into a set of tasks to execute, as is the case with the StoredProcExecutionProvider.

DW Architect supports an open API for developers to create custom Task Execution Providers. Additional task providers are in development to support platforms like Oracle and IBM DB2.

Task providers are defined by a provider Manifest. An example is below:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```

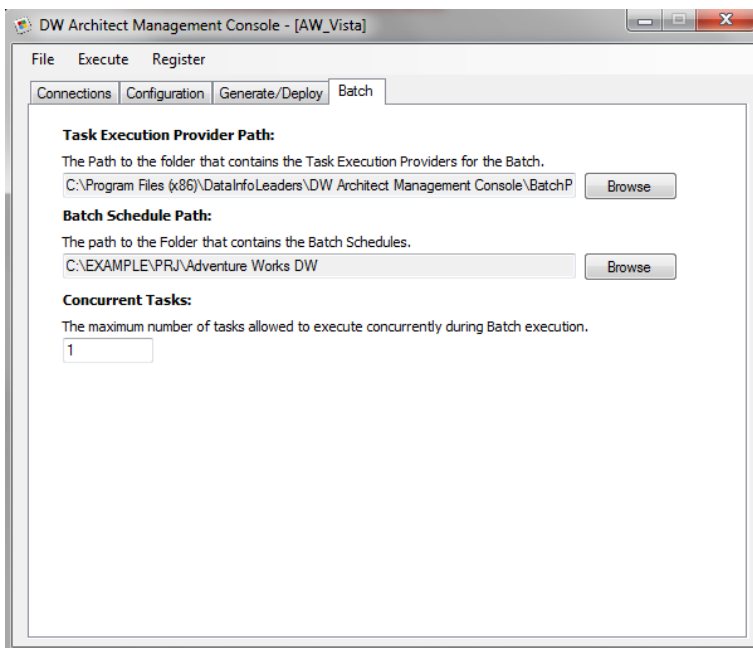
<Task_Execution_Provider_Manifest>
  <Provider_Name>StoredProcExecutionProvider</Provider_Name>
  <Provider_Class>com.datainfoleaders.batch.StoredProcExecutionProvider</Provider_Class>
  <Provider_Assembly>StoredProcExecutionProvider.dll</Provider_Assembly>
</Task_Execution_Provider_Manifest>

```

## CONFIGURATION

There are 2 additional configuration setting that need to be set prior to execution the batch.

- The path to the directory that contains your Batch workflow files. Set the Batch Schedule path on the Batch tab.
- Concurrent Tasks defines the number of batch tasks that can execute concurrently within a phase of the Batch.



## EXECUTING THE BATCH

A Batch is executed via the command line, or can be scheduled using Windows scheduler (or equivalent). Under the covers, the Batch Manager uses an internal database to record the progress of every batch task executed. The execution details are also written to an output file.

The value of Concurrent Tasks configuration setting on the Batch tab determines the number of threads that are started to execute tasks in the batch. The batch processor manages these threads, and which tasks are executed on them at which time. Only tasks within a phase can be executed Concurrently. I.e. all Extracts are executed, then all Dimension Transforms, then all Fact Transforms etc. Over time the batch processor will dynamically adjust based on the history of executing a given batch. It will determines which tasks are the longest running within a phase, and will start these first in order to shorten the overall length of the batch.