

Model Comparison & Merging for Analysis Services

Microsoft BI Technical Article

Writer: Christian Wade

Contributors and Technical Reviewers: Javier Guillen (Blue Granite), Chris Webb (Crossjoin Consulting), Marco Russo (SQLBI), Chris Woolderink (Tabular), Bill Anton (Opifex Solutions), Matt Jones, Kay Unkroth

Published: July 2017

Applies to: Microsoft SQL Server Analysis Services, Microsoft Azure Analysis Services

Summary: This whitepaper and associated samples describe how to perform database comparison and merging for Analysis Services tabular models.

Copyright

This document and associated samples are provided as-is. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2016 Microsoft. All rights reserved

Contents

Introduction & use cases	4
BISM Normalizer summary	4
Self-service & corporate BI	5
Migration to corporate BI	5
How the Microsoft platform can help	5
Import from Power BI Desktop & compatibility levels	6
Model definition reuse and merging	6
Tabular model deployment.....	6
Traditional all-or-nothing deployment	6
Partial-metadata deployment.....	7
Deployment features	7
Set up & installation.....	7
Comparison workflow	8
New comparison	8
Select actions to be applied	10
Harvesting objects for reuse in mature models.....	12
Validate selection.....	13
Script generation.....	15
Report differences	16
Update target.....	17
Saving comparisons.....	20
Deployment configurations	20
Add BSMN file to a project.....	20
View code behind	21
Associate BSMN file types with Visual Studio & display icon	22
Comparison options.....	23
Include perspectives	24
For perspective updates, merge selections (not replace)	24
Include cultures.....	27
For culture updates, merge translations (not replace).....	27
Include roles.....	27
Consider partitions when comparing tables.....	27

For table updates, retain partitions (not replace)	29
Display warnings for measure dependencies (DAX reference to missing measure/column)	29
Database deployment.....	30
Processing option.....	30
Process only affected tables	30
Command-line execution.....	30
Syntax.....	31
Arguments.....	31
Automated merging of branches.....	31
Examples	31
Passwords and processing	32
Executable location.....	32
Visual Studio projects as source/target	32
Additional Resources	32

Introduction & use cases

Relational-database schema comparison and merging is a well-established market. Leading products include SSDT Schema Compare and Redgate SQL Compare, which is partially integrated into Visual Studio. These tools are used by organizations seeking to adopt a DevOps culture to automate build-and-deployment processes and increase the reliability and repeatability of mission critical systems.

Such functionality is also available for Analysis Services tabular models. This document describes how to use the BISM Normalizer open-source tool for application-lifecycle management (ALM) scenarios. In addition to enabling the adoption of DevOps processes for tabular models, BISM Normalizer can help bridge the gap between self-service and IT-owned “corporate BI”.

BISM Normalizer summary

BISM Normalizer is a Visual Studio extension that works with Azure Analysis Services (Azure AS) and SQL Server Analysis Services (SSAS) tabular models. It allows a modeler to perform metadata merging across models. All tabular model objects are supported. As a Visual Studio extension, it is tightly integrated with source control and model management workflows.

The screenshot shows the BISM Normalizer tool in Visual Studio. The main window displays a comparison table between a source project and a target database. The table lists various objects like Data Source, Table, Measure, and Relationship, along with their status (e.g., Different Definitions, Missing in Source) and actions (Update, Delete, Create). Below the table, there are sections for Source Object Definition and Target Object Definition, showing JSON-like metadata for objects like 'Freight' and 'Margin'. At the bottom, a BISM Normalizer Warning List is visible, showing messages such as 'Delete relationship' and 'Unable to create Relationship'.

Type	Source Name	Status	Target Name	Action
Data Source	SQL/CHWADESQSQLSAT\SP1:AdventureWorksDW	Different Definitions	SQL/CHWADESQSQLSAT\SP1:AdventureWorksDW	Update
Table		Missing in Source	DimCurrency	Delete
Table	DimDate	Different Definitions	DimDate	Update
Measure	DaysCurrentQuarterToDate	Missing in Target		Create
Measure	DaysInCurrentQuarter	Missing in Target		Create
Table	DimProduct	Different Definitions	DimProduct	Skip
Relationship	'DimProduct'[ProductSubcategoryKey]->DimProd...	Missing in Target		Skip
Table	FactInternetSales	Different Definitions	FactInternetSales	Update
Relationship		Missing in Source	'FactInternetSales'[CurrencyKey]->DimCurrenc...	Delete

```
{  
  "name": "Freight",  
  "dataType": "decimal",  
  "sourceColumn": "Freight"  
},  
{  
  "type": "calculated",  
  "name": "Margin",  
  "dataType": "decimal",  
  "isDataTypeInferred": true,  
  "expression": "[SalesAmount]-[TotalProductCost]"  
}
```

```
{  
  "name": "Freight",  
  "dataType": "decimal",  
  "sourceColumn": "Freight"  
},  
{  
  "name": "OrderDate",  
  "dataType": "dateTime",  
  "sourceColumn": "OrderDate"  
}
```

BISM Normalizer Warning List

- 1 Warnings
- 36 Informational Messages

Message

- Relationships
- Delete relationship 'FactInternetSales'[CurrencyKey]->'DimCurrency'[CurrencyKey].
- Create relationship 'FactResellerSales'[DueDateKey]->'DimDate'[DateKey].
- Unable to create Relationship 'FactResellerSales'[EmployeeKey]->'DimEmployee'[EmployeeKey] because (considering changes) parent table n

Self-service & corporate BI

IT-owned, corporate BI has the following characteristics compared to self-service BI.

- **Self-service BI is characterized by having a large number of small models.** This is a result of enabling agility and freedom for analysts to uncover new insights and data sets without a dependency on IT.
- **Corporate BI is characterized by having a small number of large models.** This promotes reusability, governance, consistent decisions based on corporate metrics, data quality and efficiencies around the management of data.

The reality is most organizations need to strike a balance between the two camps. The [Power BI Governance and Deployment](#) whitepaper describes **bimodal BI** as “the practice of managing two separate, coherent modes of business intelligence and analytics delivery: one focused on stability and the other on agility”. Bimodal BI helps mitigate the risk of having competing IT-owned and business-owned models offering redundant solutions with conflicting definitions. The challenge lies in how the two camps can work together in harmony. This is an organizational problem before a technology problem; however, an end-to-end, modern BI platform spanning both self-service and corporate BI can tremendously help organizations seeking to adopt a bimodal BI strategy.

Migration to corporate BI

To promote collaboration between the two camps, a process may be defined to harvest self-service BI datasets, put them through the necessary rigor to be certified for corporate BI, and benefit from wider reusability. The [Power BI Governance and Deployment](#) whitepaper calls this “ownership transfer”. Considerable refactoring work may be required depending on factors such as technology alignment and data source reuse. This process reduces the overhead of managing redundant copies of the same data, and helps standardize data definitions. For example, different versions of the same metrics can be highlighted and corrected.

It may be common to harvest only a subset of features from a popular self-service model. This could happen, for example, when only a portion of the self-service model is compliant with enterprise standards. The rest of the model may have definitions that perhaps don't yet comply with enterprise patterns and require additional refactoring.

The following workflow is noted on Page 6 of the [Create a Centralized and Decentralized Organizational Model for Business Intelligence](#) Gartner paper to be considered when devising a BI strategy: “Local teams are being empowered to create and innovate. The centralized team identifies the most successful work being done at a local level, and provides a platform to share and promote this work globally.”

How the Microsoft platform can help

Microsoft's end-to-end, modern BI platform spans both self-service BI modeling with Power BI Desktop and corporate BI modeling with Analysis Services. Both products share a common engine, modeling constructs and calculation language. This technology alignment accelerates the following objectives.

- IT adopts popular self-service BI model definitions without having to “re-invent the wheel”.
- Technology skills in the organization are standardized and streamlined.

Microsoft provides a supported way to import a Power BI Desktop model into Analysis Services. When migrating from self-service to corporate BI, it is reasonable to expect that only a subset of definitions in the migrated model need to be harvested for existing corporate models. **Reuse of definitions for corporate models avoids maintaining a separate IT-owned model for each migration. This furthers the strategic objective of corporate BI to maximize reusability by reducing the number of models holding the same (or similar) information.** BISM Normalizer allows IT to pick and choose definitions from self-service migrated models to be promoted to existing corporate models.

[Power BI audit logs](#) can be used to help identify self-service models as candidates for harvesting that are frequently used by the business.

Import from Power BI Desktop & compatibility levels

Microsoft provides a supported way to import a Power BI Desktop model into Analysis Services. This is a prerequisite to perform a comparison with BISM Normalizer; the source and target models must be Analysis Services models. At time of writing, import from Power BI Desktop works using the Azure AS web experience. When performing an import to Azure AS, it is unlikely there will be engine-version incompatibility issues because Azure AS is frequently updated to new versions. Power BI Desktop embedded-model features that are not currently supported in Analysis Services (e.g. binning, grouping, clustering) should be ignored by the import. In some cases, such as unsupported data sources, the import may be disallowed altogether. Once the model is successfully imported to Azure AS, it can be used as the source for a comparison.

If the target model of a comparison is SSAS, it is possible that the target server version is not new enough for all the source model features. This is generally handled using [Analysis Services compatibility levels](#). BISM Normalizer normally requires that the source and target models have the same compatibility level. At time of writing, the only case where different compatibility levels are allowed is if the source has 1200 and the target is 1400; in this case the comparison is allowed.

Model definition reuse and merging

The following scenarios are supported.

- Merge models (or subsets thereof) migrated from self-service BI into existing IT owned models.
- Pick and choose objects to reuse in other models. For example, conformed dimensions, standardized date dimensions, and common measures to be standardized across models.
- Standardize objects across different groups/developers in large, enterprise organizations.

Tabular model deployment

Corporate BI scenarios require deployment of new versions across environments such as development, test and production to allow rigorous testing and user-signoff processes. Deployment of tabular models normally employs an all-or-nothing, or partial-metadata deployment strategy. Some organizations may choose to use a combination; for example, partial-metadata deployment for bug fixes, but not features.

Traditional all-or-nothing deployment

- All features currently in development are packaged into a deployment artifact and deployed across the environments (development, test, production, etc.).
- Existing data in the target environment is retained during the deployment, and the model metadata is brought up to the deployed version.

- A production branch is normally employed for bug fixes. This allows quick deployment of bug fixes without requiring a full deployment cycle.
- Branch management is often required to merge the production bug fixes into the main branch.

BISM Normalizer supports all-or-nothing deployments, and is the only tool that supports merging of branches for tabular models.

Partial-metadata deployment

- Deploy only bug fixes without unfinished features still in development.
- Deploy individual features through environments based on user feedback.

This approach avoids the need for branch management. It is useful for organizations who do not have branch-management skills, or the desire to take on the additional overhead. The ability to deploy individual features through environments (development, test, production, etc.) without full deployment cycles can result in business agility and responsiveness to customer needs.

There are risks associated with partial-metadata deployment. However, tabular models are relatively well structured (for example, Analysis Services multidimensional models have far more interdependencies). The dependency analysis/safeguards (discussed below in [validate selection](#) section) provided by BISM Normalizer make partial-metadata deployment a viable option in many cases.

Deployment features

For either approach, BISM Normalizer supports the following deployment features for tabular models.

- Deployment configurations are supported by saving comparison information such as source/target model, comparison options, and skipped actions to a BSMN file. It can later be used as a configuration for deployment with a database/environment combination. This is discussed below in the [deployment configurations](#) section.
- Retain partitions, role members, environment-specific data sources, etc. Partitions are normally created and managed by automated processes; they don't exist in the version of the tabular model in source control. BISM Normalizer allows partitions, and the data with them, to be retained on deployment.
- Command-line execution for integration with continuous-integration processes.
- TMSL script generation to give to a DBA for execution with elevated permissions.
- Processing (data refresh), limited to affected tables, on database deployment in UI mode through Visual Studio.

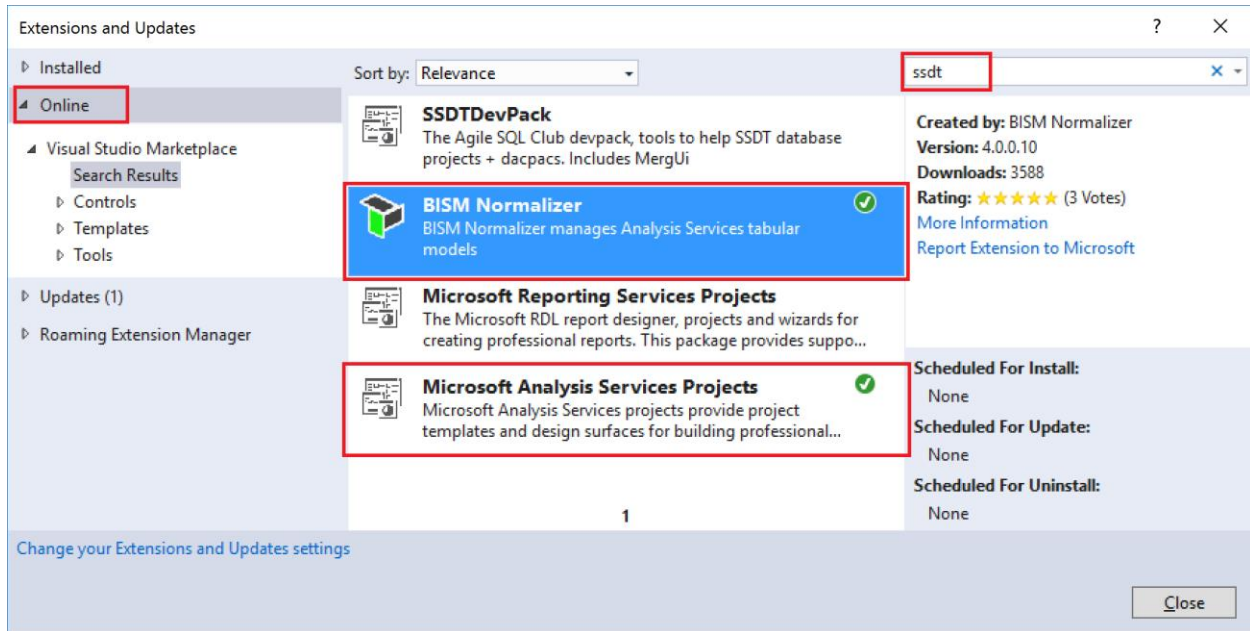
Set up & installation

To get started, you'll need Visual Studio 2017. You can download and install the free Community Edition [here](#). Visual Studio 2015 is also supported, but 2017 is recommended because it allows installation of the Analysis Services SSDT extension from the Extensions and Updates dialog.

The examples in this document use Adventure Works, but any model will work. At time of writing, all tabular compatibility levels are supported – 1400, 1200, 1103, 1100. Source and target models can be projects in SSDT and/or databases on a server, which can be Azure AS and/or SSAS.

In Visual Studio 2017, take the following actions.

1. Select the Tools > Extensions and Updates menu option.
2. Select the Online tab on the left of the Extensions and Updates Dialog.
3. Type “ssdt” in the search box.
4. If you have not already installed the Analysis Services SSDT extension, do so.
5. Install BISM Normalizer and restart Visual Studio.



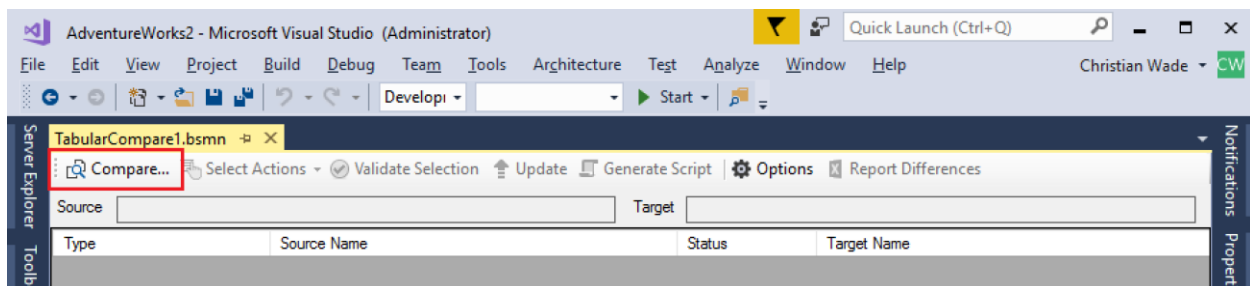
Comparison workflow

To follow the examples in this section, you may use the sample solution below by cloning the Analysis Services Git [repo](https://github.com/Microsoft/Analysis-Services/tree/master/BismNormalizer/Sample). The TargetModel project should be pre-deployed to an Azure AS or SSAS instance to allow working through the database deployment and processing (data refresh) steps.

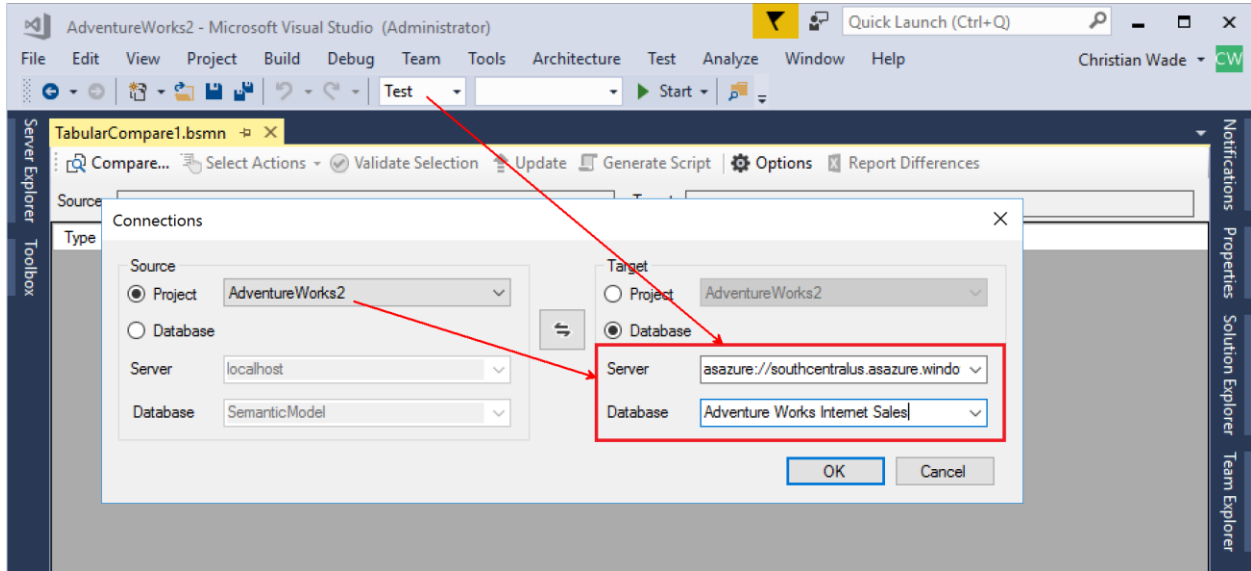
<https://github.com/Microsoft/Analysis-Services/tree/master/BismNormalizer/Sample>

New comparison

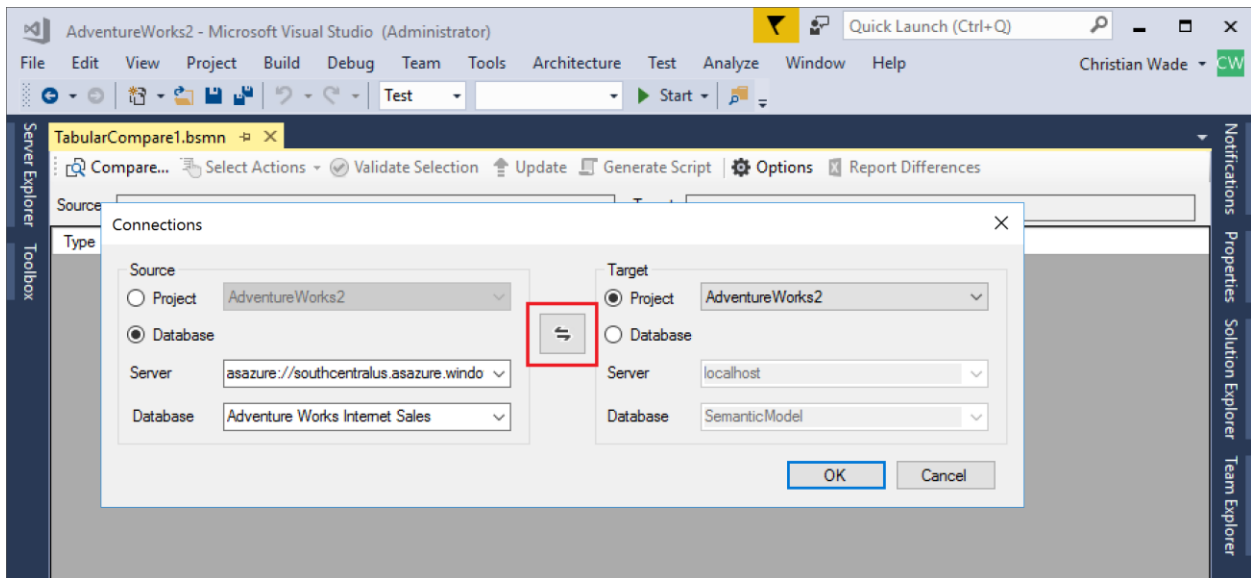
Select the Tools > New Tabular Model Comparison menu option. Click the Compare button.



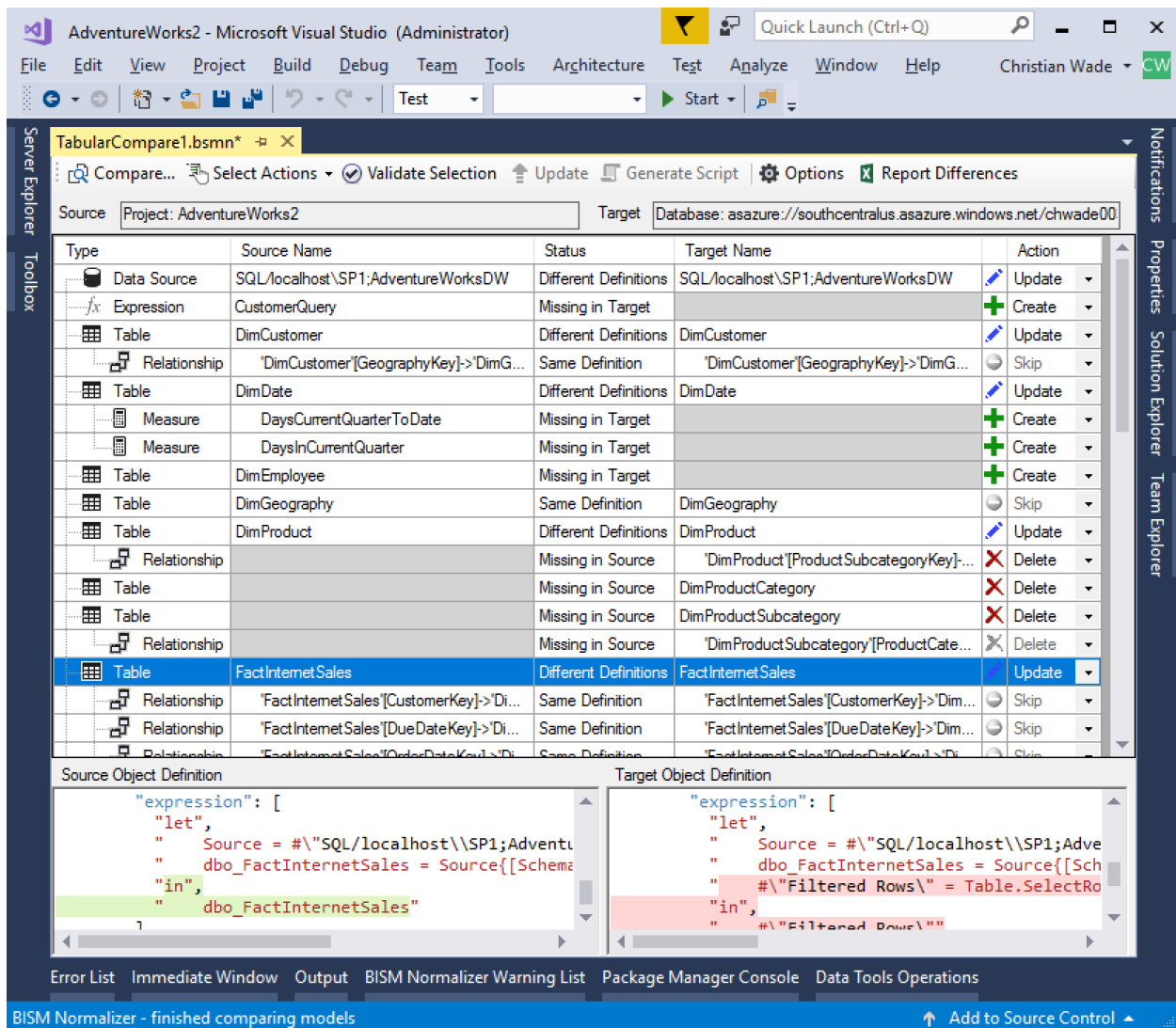
In the Connections dialog, specify the source and target databases/projects. Tabular projects loaded in the current solution will be available for selection. Default values for the target server and database are based on the selected source project's deployment properties, and the current Visual Studio configuration. Selecting a different source project resets the defaults.



You can switch source and target in Connections dialog.

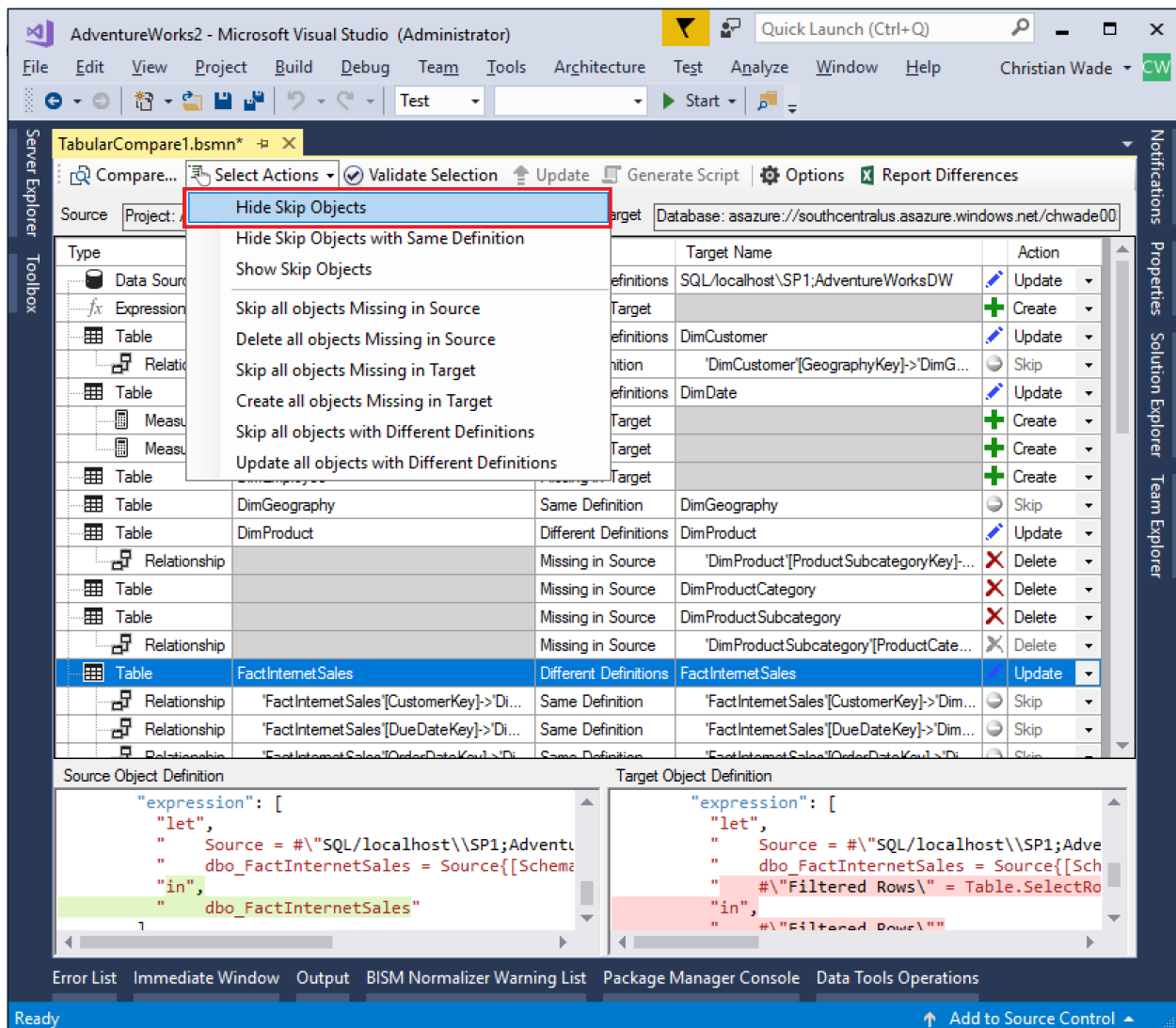


With the desired source and target specified, click OK and consider the differences displayed. In this example, the source and target models have compatibility level 1400 and therefore M expressions are included in the comparison.



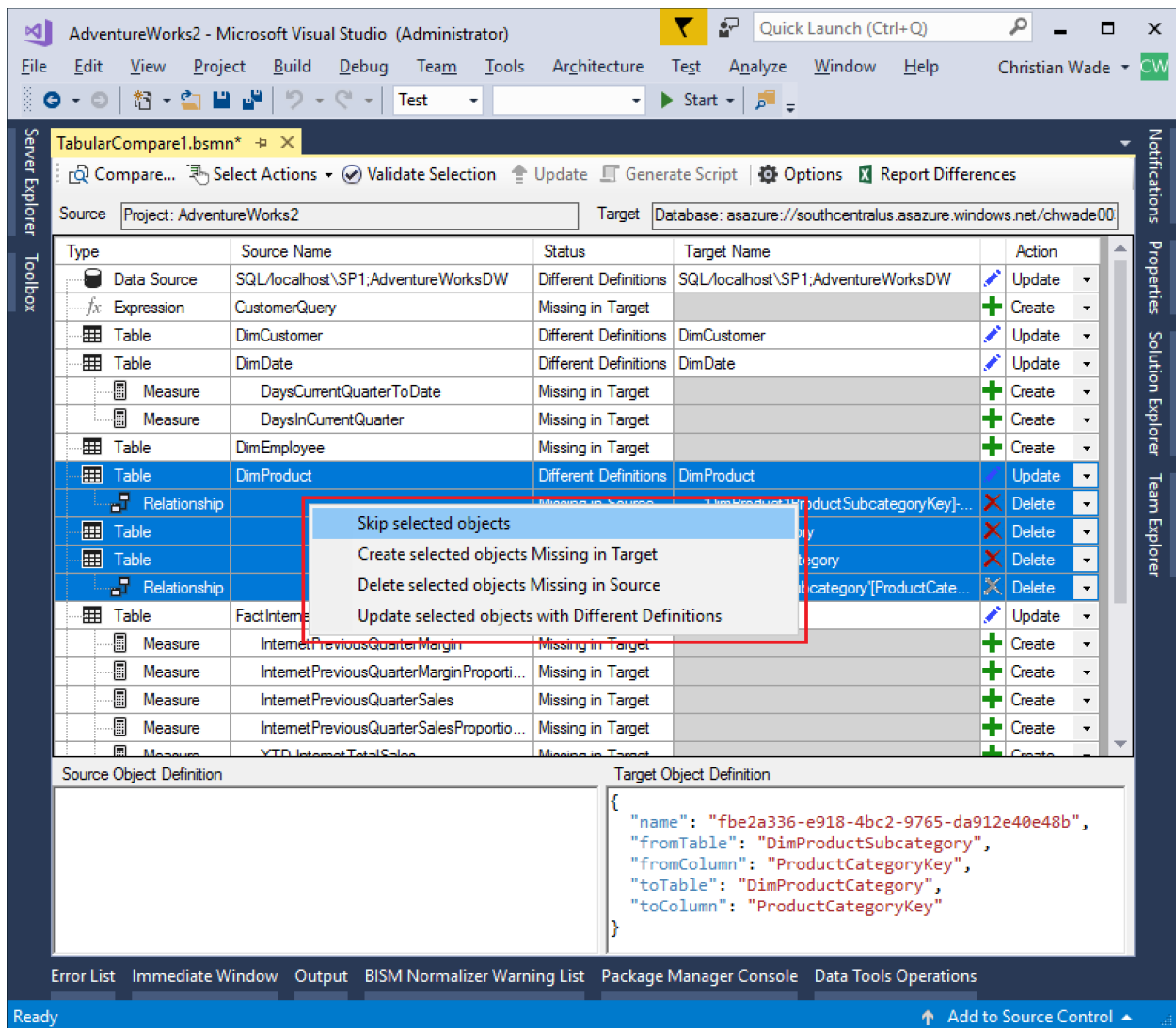
Select actions to be applied

Hide Skip Objects to focus on the differences. Objects with the same definition are automatically skipped and the action is greyed out.



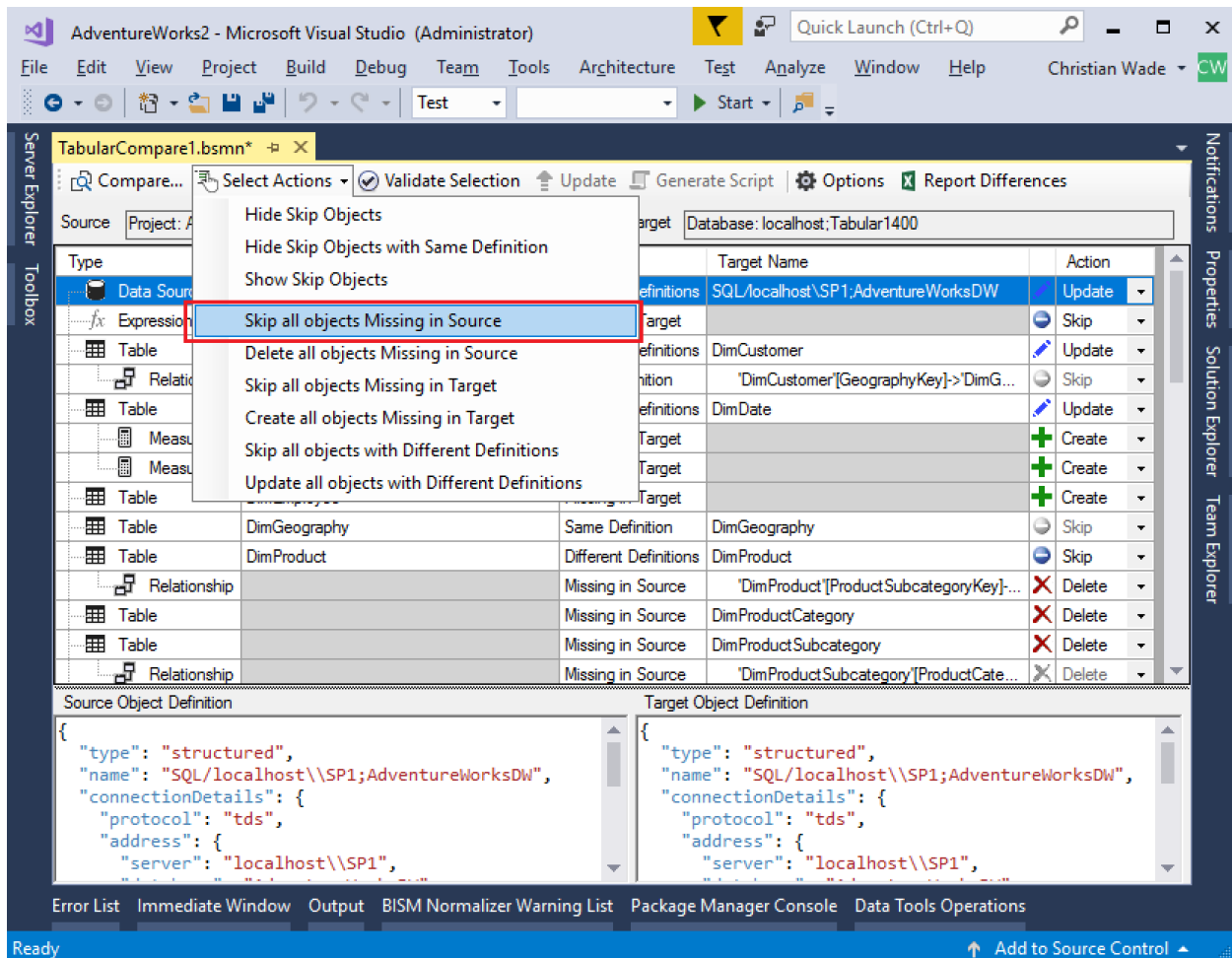
Actions available are Create, Update and Delete. Select the actions you wish to take.

- Select items individually using the dropdowns in the Action column.
- Specify multiple actions at once using the Select Actions dropdown on the toolbar (see image above).
- Specify multiple actions at once by selecting multiple items, right-click and use the context menu (see image below).



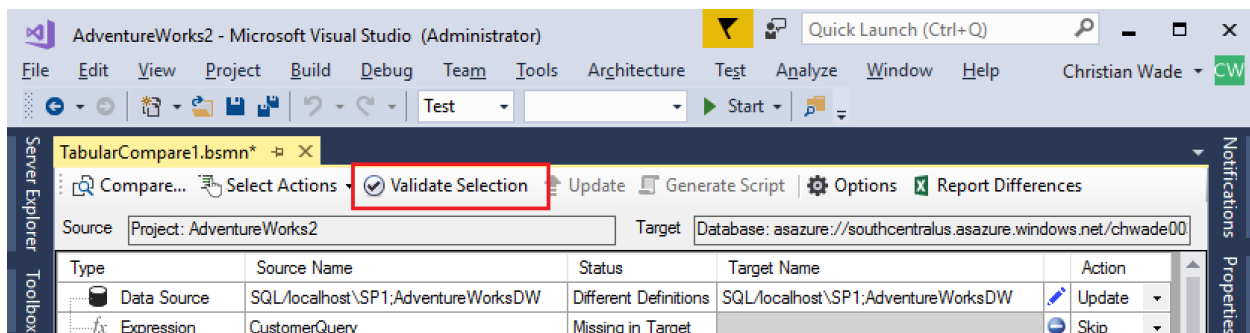
Harvesting objects for reuse in mature models

When harvesting objects – perhaps from a prototype model, or a model migrated from self-service BI – for reuse in a mature model, we often don't want to affect any of the objects already in the target. Such objects have a Delete action by default because they exist in the target, but not the source model. Instead of skipping all the Delete actions individually, we can select Skip all objects Missing in Source to skip all deletes in one go.

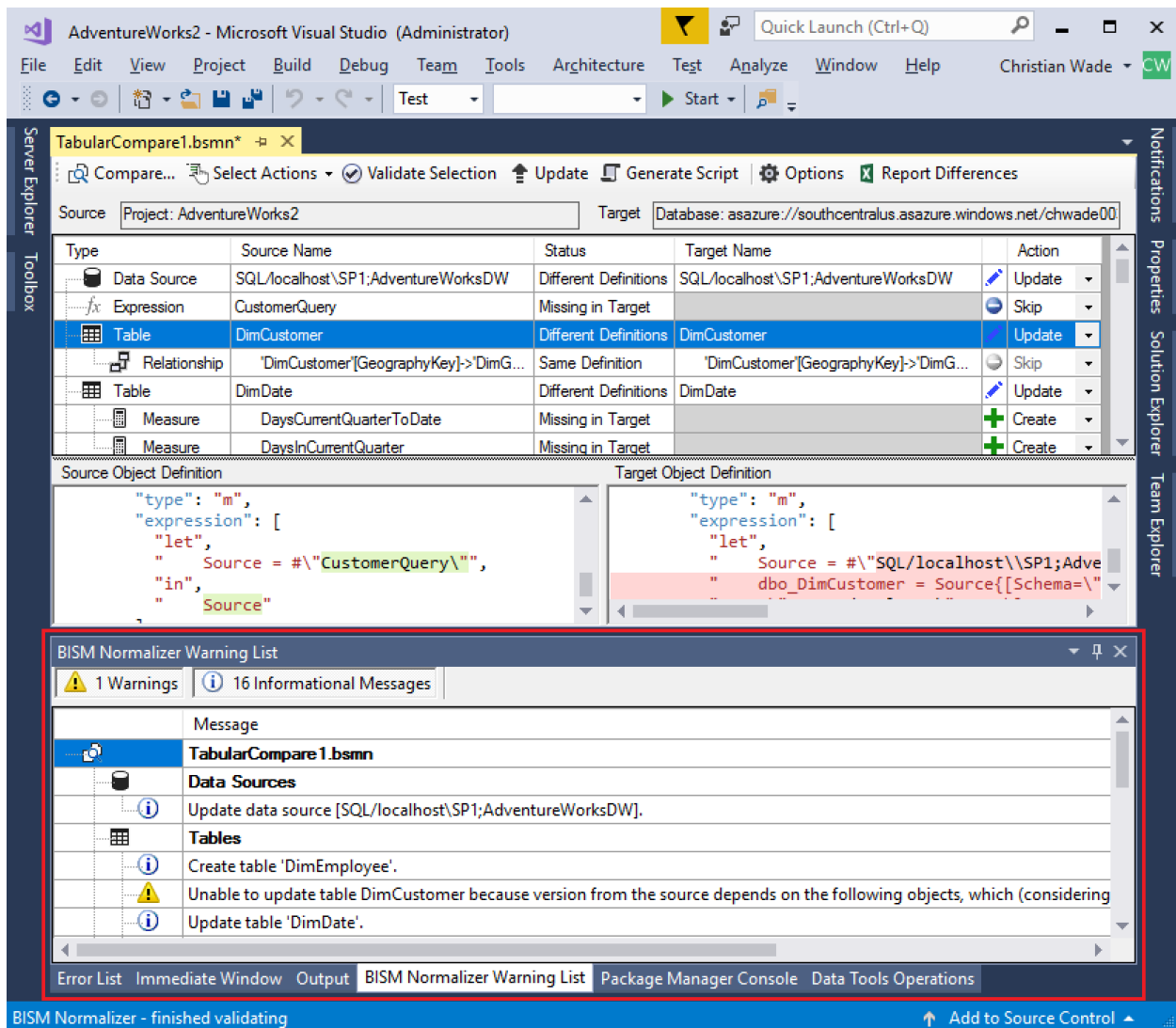


Validate selection

Click Validate Selection on the toolbar.



Check any warnings in the BISM Normalizer Warning List. This impact analysis safeguards the integrity of the target model. In the example below, the partition definition for DimCustomer depends on the CustomerQuery M expression, which is skipped. Therefore, updating the table would introduce an invalid object reference, and BISM normalizer does not update the table.



Invalid object dependencies that are handled include the following.

- Ambiguous relationship paths resulting from multiple active relationships leading to a dimension table.
- Relationships referring to nonexistent tables/columns.
- Measures/KPIs with conflicting names.
- M expression references to other M expressions and data sources that are missing.
- Table/partition dependencies on M expressions and data sources that are missing.
- Invalid DAX measure references display warnings if the option is selected (see [section](#) below), but they do not prevent objects from being created/updated/deleted.

In addition, some validation is automatically imposed by the UI.

- Deleting a table automatically sets all relationships and measures/KPIs within it also be deleted and greyed out.

Table		Missing in Source	FactInternetSales	Delete
Relationship		Missing in Source	'FactInternetSales'[CustomerKey]->'Dim...	Delete
Relationship		Missing in Source	'FactInternetSales'[DueDateKey]->'Dim...	Delete
Relationship		Missing in Source	'FactInternetSales'[OrderDateKey]->'Di...	Delete
Relationship		This object's action option is disabled due to a parent object selection]->'DimP...		Delete
Relationship		Missing in Source	'FactInternetSales'[ShipDateKey]->'Dim...	Delete
Measure		Missing in Source	InternetCurrentQuarterMargin	Delete
Measure		Missing in Source	InternetCurrentQuarterMarginPerforman...	Delete
Measure		Missing in Source	InternetCurrentQuarterSales	Delete

- Skipping creation of a table automatically skips and greys out creation of all relationships and measures/KPIs within it.

Table	FactInternetSales	Missing in Target	Skip
Relationship	'FactInternetSales'[CustomerKey]->'Di...	Missing in Target	Skip
Relationship	'FactInternetSales'[DueDateKey]->'Di...	Missing in Target	Skip
Relationship	'FactInternetSales'[OrderDateKey]->'Di...	Missing in Target	Skip
Relationship	'FactInternetSales'[ProductKey]->'Din	This object's action option is disabled due to a parent object selection	
Relationship	'FactInternetSales'[ShipDateKey]->'Di...	Missing in Target	Skip
Measure	InternetCurrentQuarterMargin	Missing in Target	Skip
Measure	InternetCurrentQuarterSales	Missing in Target	Skip
Measure	InternetDistinctCountSalesOrder	Missing in Target	Skip

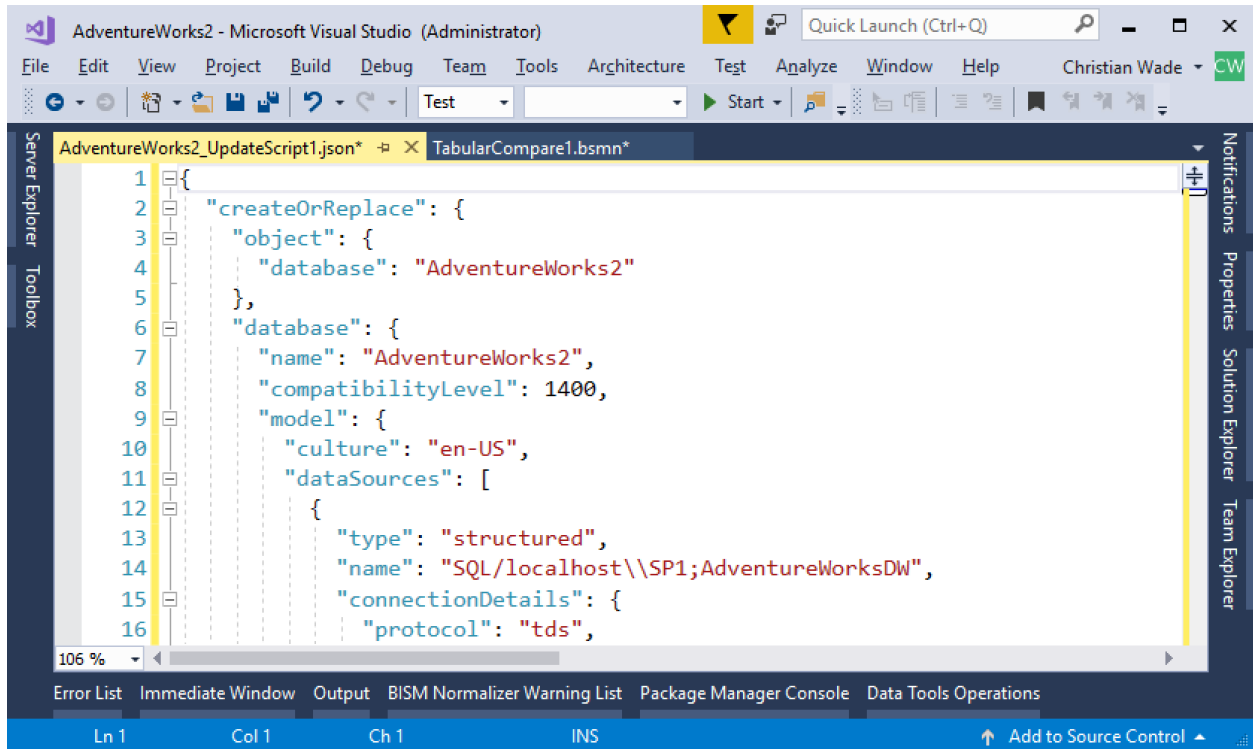
Script generation

To generate a [Tabular Model Scripting Language](#) (TMSL) script of the changes, perhaps to give to a DBA for execution with elevated permissions, click the Generate Script button.

The screenshot shows the 'Generate Script' dialog in Visual Studio. The 'Generate Script' button is highlighted with a red box. Below the dialog, a table shows the actions to be performed:

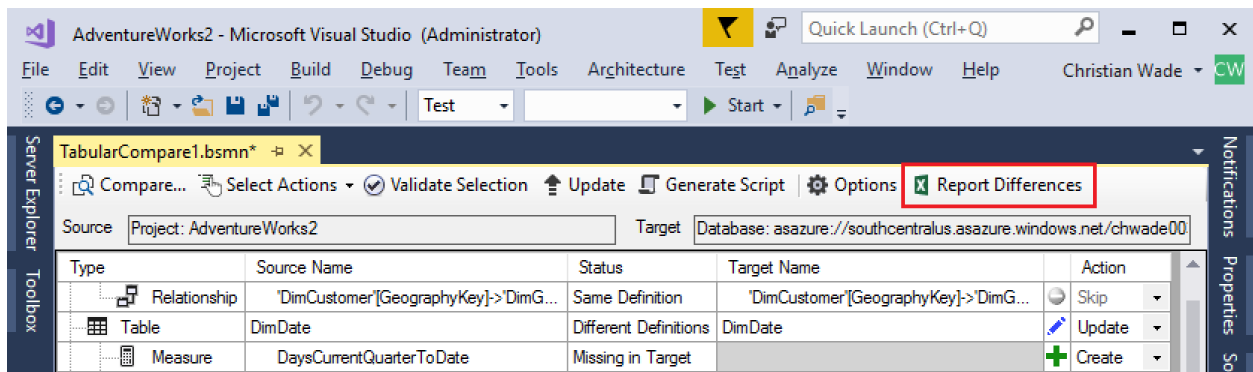
Type	Source Name	Status	Target Name	Action
Relationship	'DimCustomer'[GeographyKey]->'DimG...	Same Definition	'DimCustomer'[GeographyKey]->'DimG...	Skip
Table	DimDate	Different Definitions	DimDate	Update
Measure	DaysCurrentQuarterToDate	Missing in Target		Create

A TMSL script is generated based on the actions selected above.

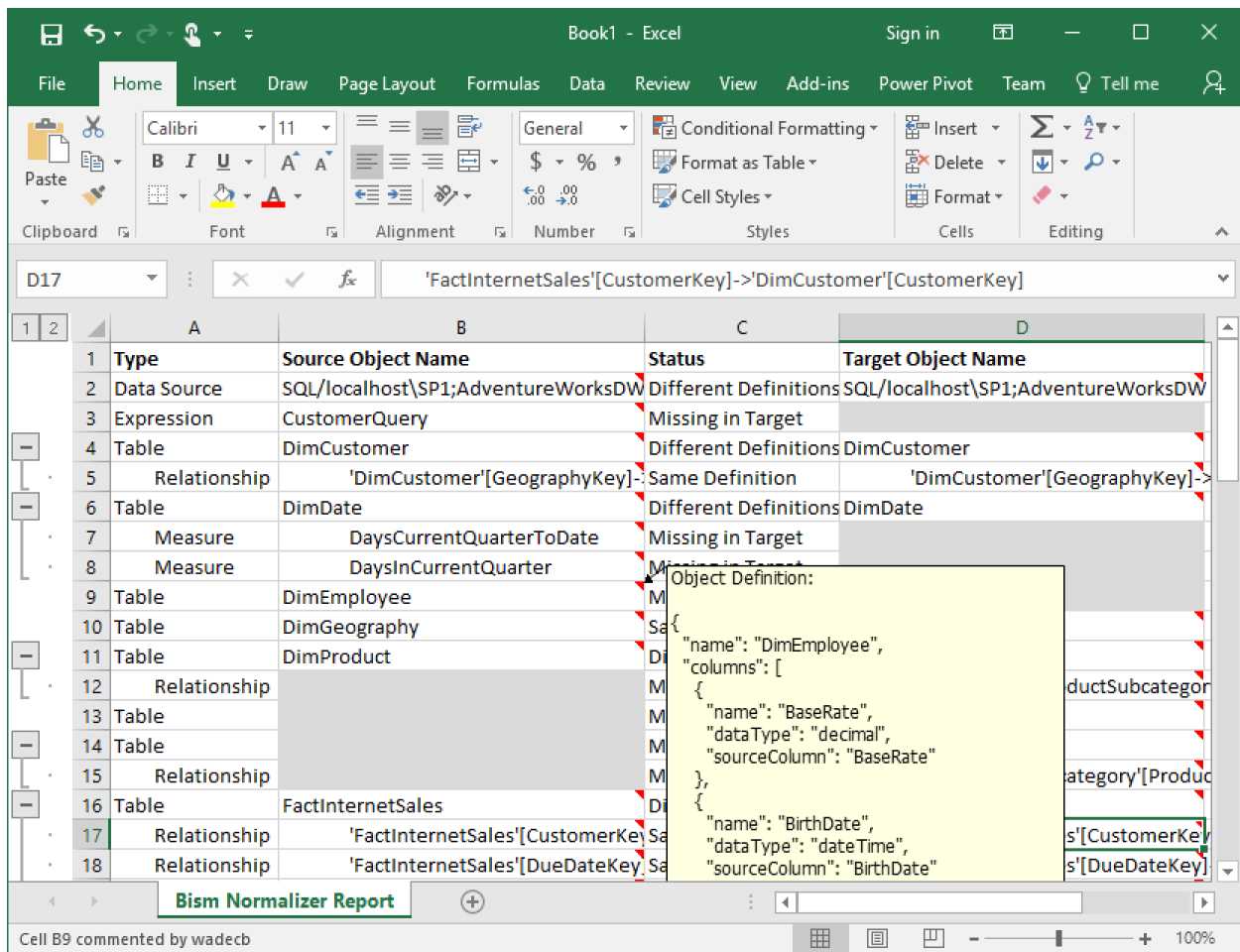


Report differences

To create a differences report, perhaps for auditing purposes, click the Report Differences button.

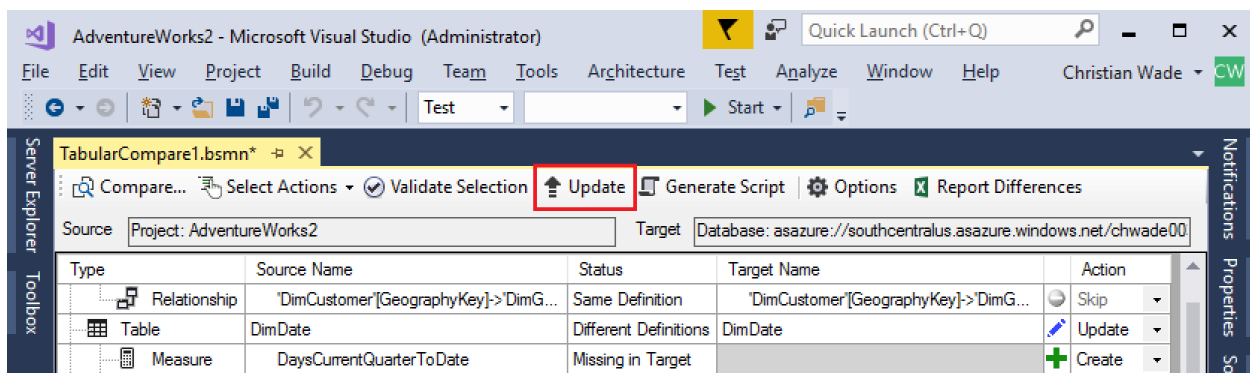


An Excel report is generated and displayed.



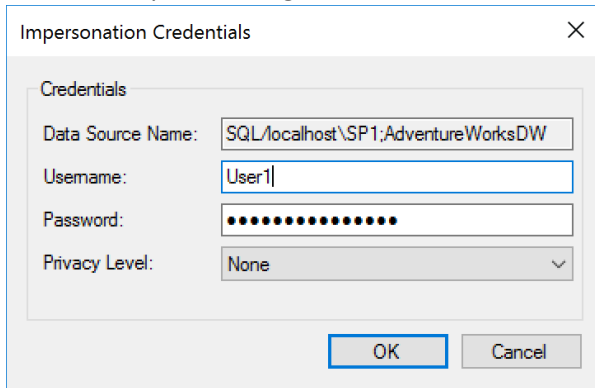
Update target

Click the Update toolbar button and confirm you want to update the target when prompted.



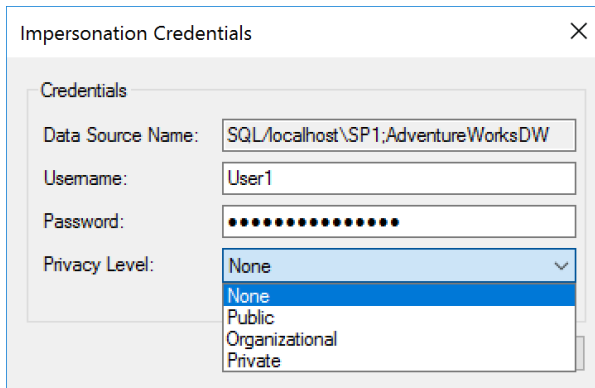
- If the target is a project, its metadata stored in the BIM file and workspace database will be updated.
- If the target is a database, processing (data refresh) actions may be taken depending on the comparison options specified (see below), and credentials will be prompted for data sources.

In this example, the target is a database and Windows impersonation credentials are required.



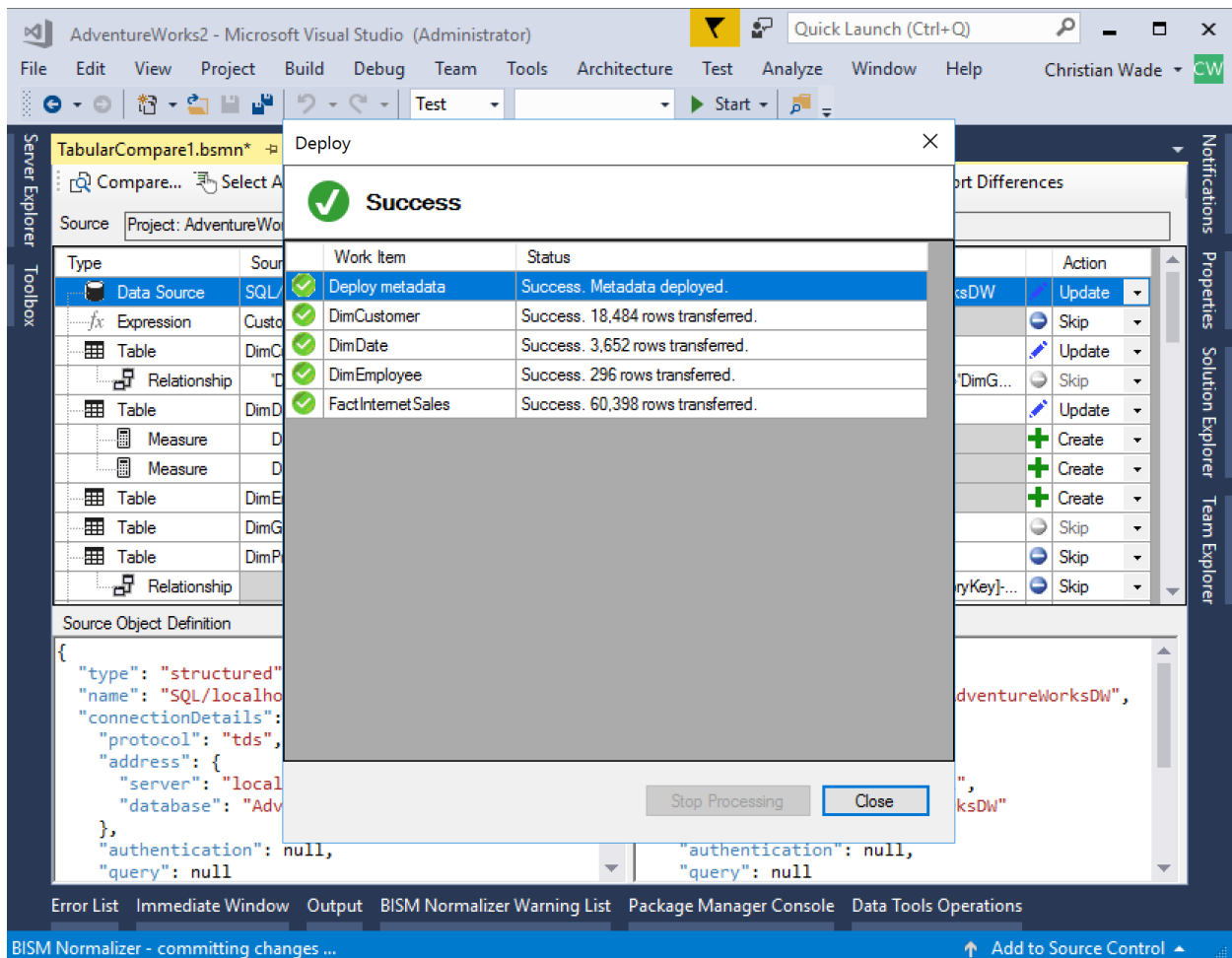
The screenshot shows a dialog box titled "Impersonation Credentials" with a close button (X) in the top right corner. Inside the dialog, there is a section labeled "Credentials" containing four fields: "Data Source Name" with the value "SQL/localhost\SP1;AdventureWorksDW", "Username" with the value "User1", "Password" which is masked with 12 black dots, and "Privacy Level" which is a dropdown menu currently set to "None". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

The privacy level can also be specified at this point. See this [Power BI article](#) for more information on privacy levels in Power Query. The same concepts apply to Analysis Services.

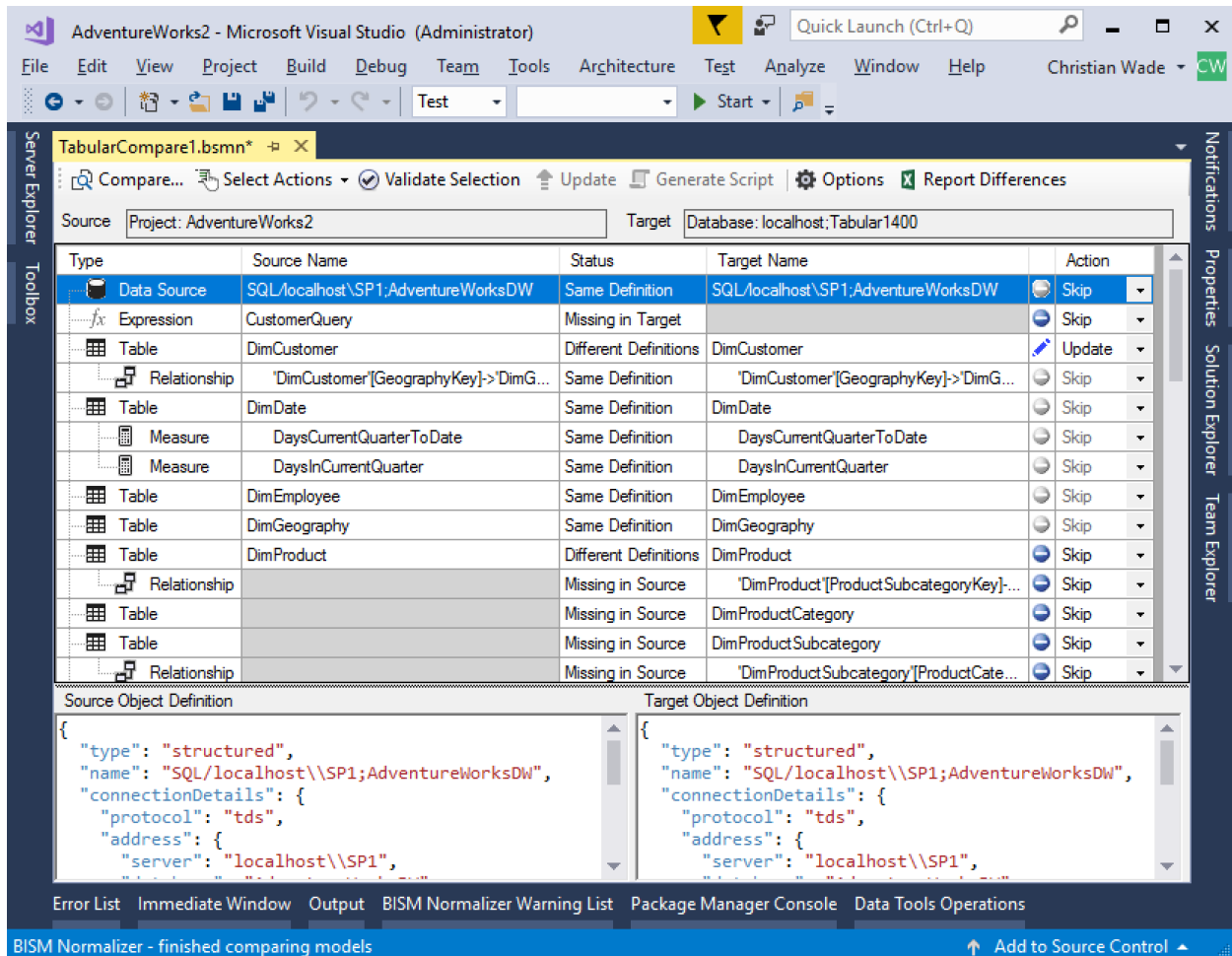


This screenshot is identical to the previous one, but the "Privacy Level" dropdown menu is open, showing a list of options: "None", "Public", "Organizational", and "Private". The "None" option is currently selected and highlighted in blue.

Having specified the necessary credentials for each data source, only the affected (created/updated) tables are processed. Again, this behavior is configurable in the comparison options (see [database deployment](#) section below).



With the changes applied and the comparison refreshed, all objects now have the same definitions except those that were skipped or with changes that could not be applied due to validation constraints.



Saving comparisons

Comparisons can be saved to BSMN files to be applied/reapplied later. The following information is retained.

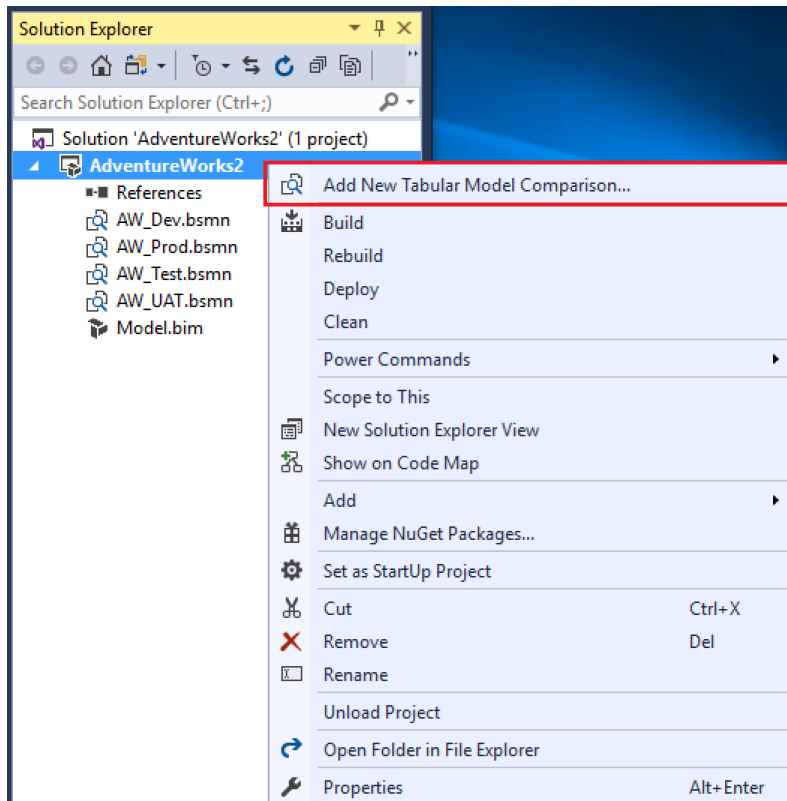
- Connection information for source and target models such as database/server/project.
- Comparison options.
- Skipped actions for model differences.

Deployment configurations

BSMN files can be used as deployment configurations. For example, a model can be associated with separate BSMN files for development, test and production environments. Saved skipped actions often include roles with different members in each environment, and data sources pointing at different instances of data sources.

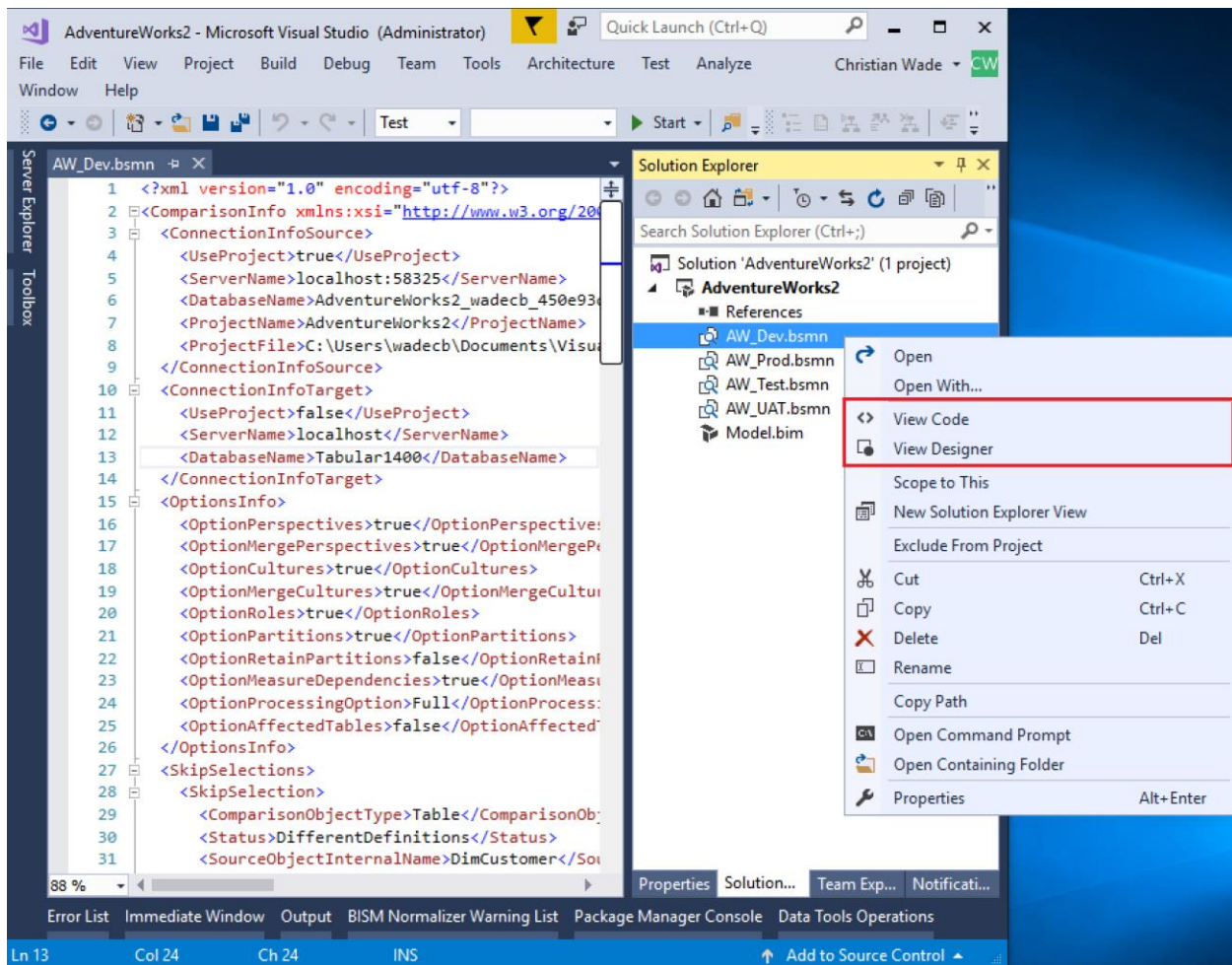
Add BSMN file to a project

To add a BSMN file to a project in Solution Explorer, right-click and select Add New Tabular Model Comparison.



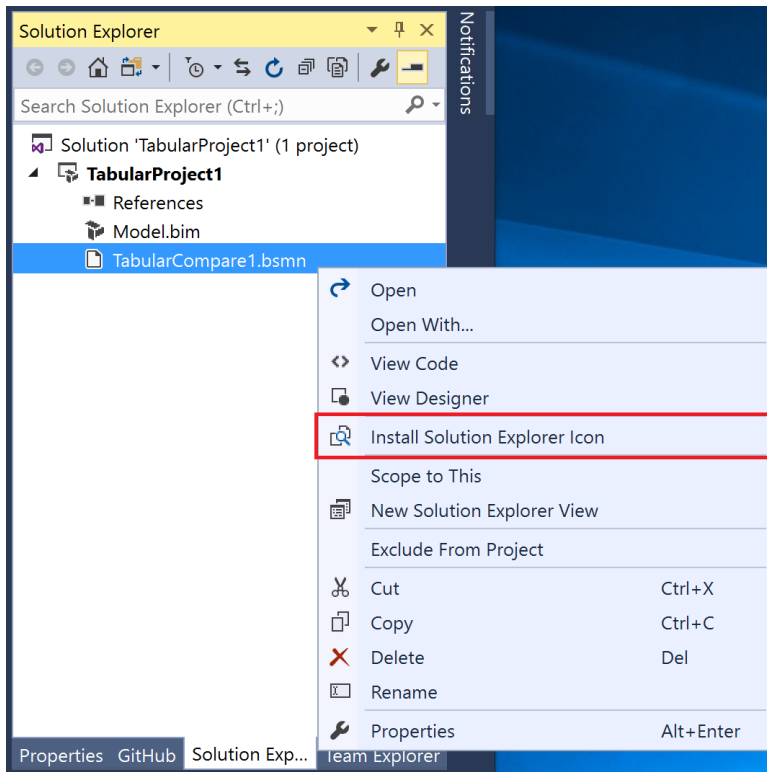
View code behind

To view or modify the code behind a BSMN file, right-click and select View Code. To switch back to design view, select View Designer.



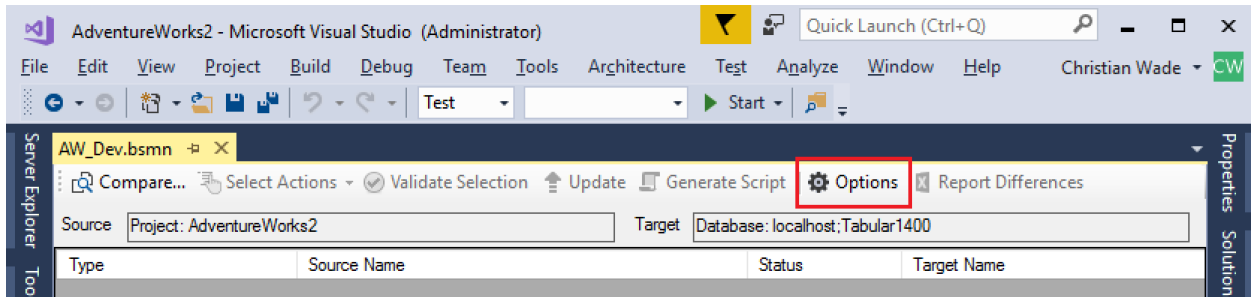
Associate BSMN file types with Visual Studio & display icon

BSMN files can be opened from Windows Explorer with Visual Studio as the default application, and displayed with the icon in Solution Explorer. To enable this, right click a BSMN file in Solution Explorer and select Install Solution Explorer Icon. Note: administrator permissions are required to even see the menu option. This can't be set up automatically using standard VSIX deployment from the Visual Marketplace with auto-updates because of the admin requirement.

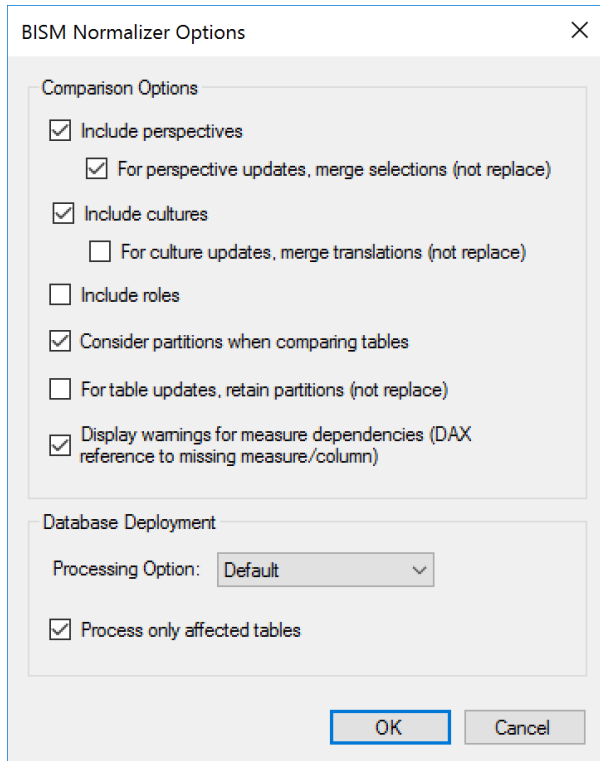


Comparison options

Click Options on the toolbar.



The Options dialog is displayed.



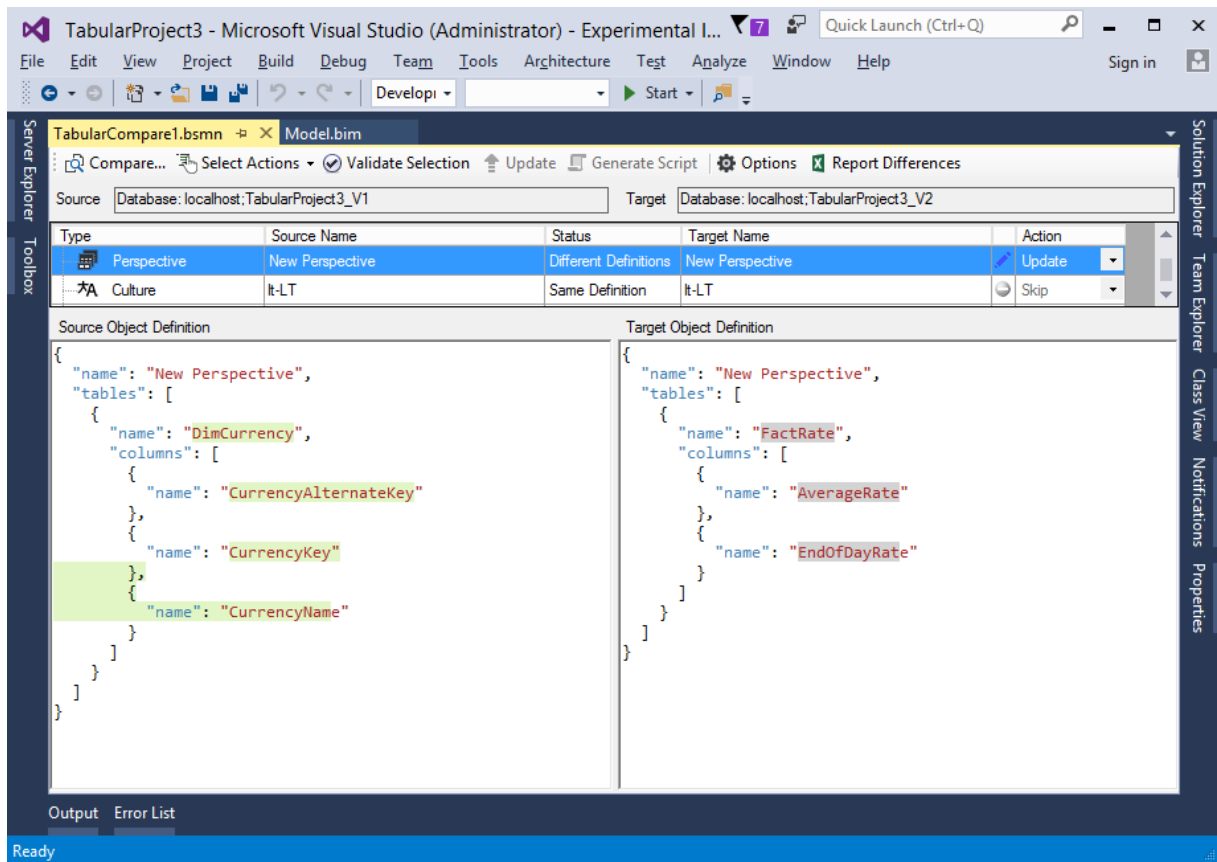
Include perspectives

Excludes perspectives from comparison if unchecked.

For perspective updates, merge selections (not replace)

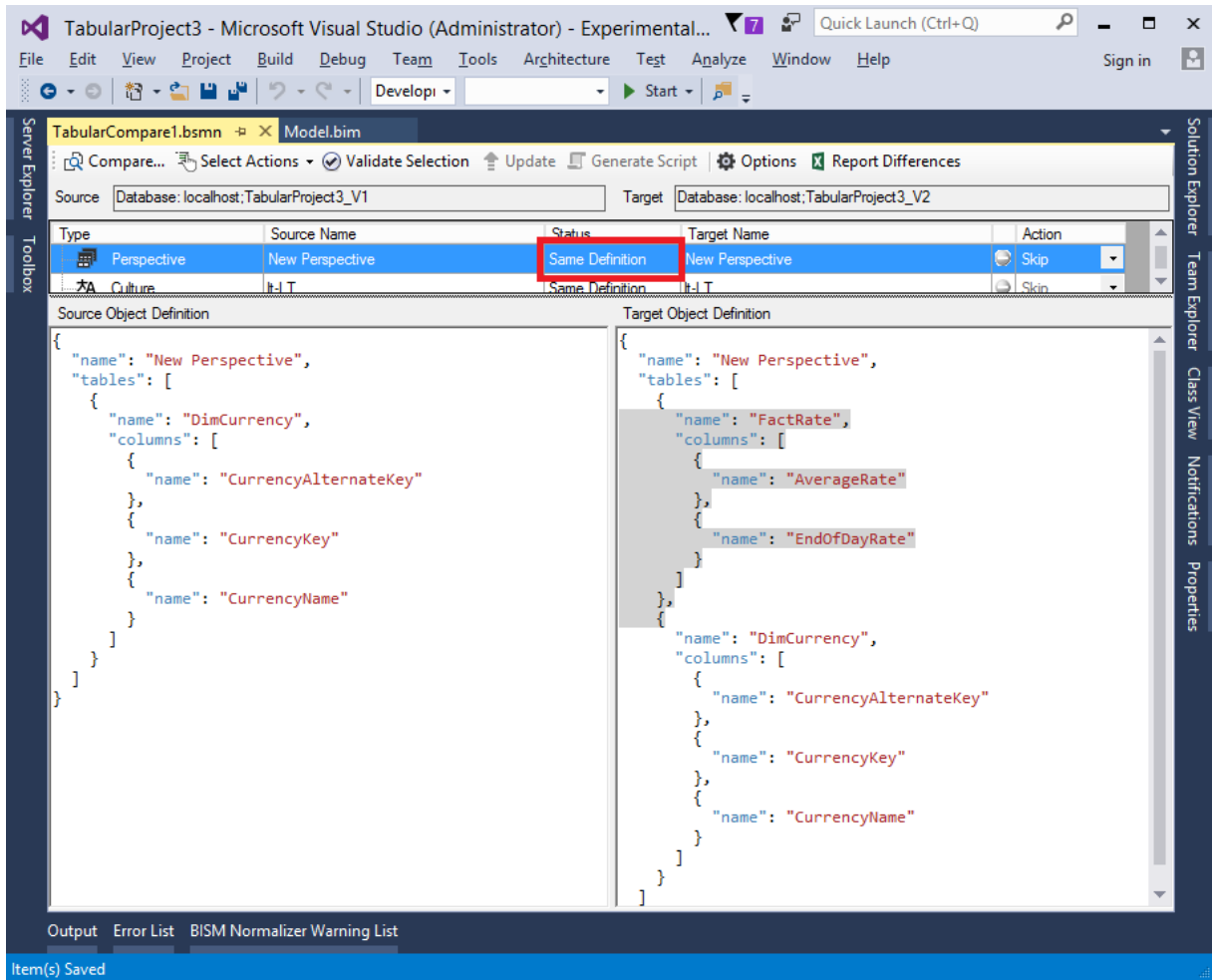
When merging models, it may be useful to create selections from a source perspective without losing existing selections that were already in the target.

Consider the following comparison taken with the merge perspectives option checked.



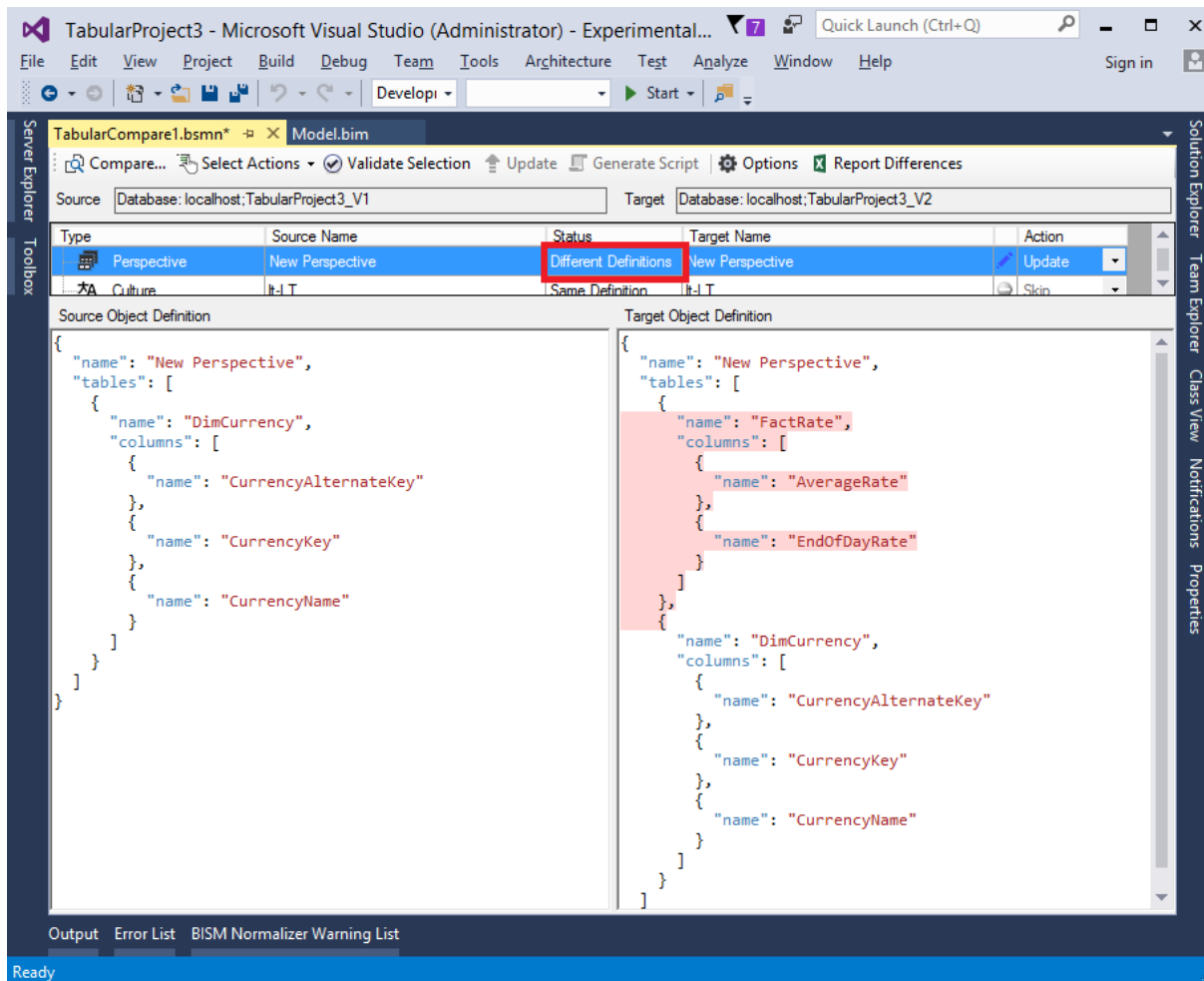
The differences in the target object definition are highlighted in grey instead of the normal red. This is to indicate that they will not be removed.

After applying the update (assuming FactRate is still in the target database) and re-running the comparison, the following definitions are displayed.



For the purpose of the comparison, the definitions are considered the same because an update would have no effect.

However, if the merge perspectives option is unchecked and the comparison is re-run, the definitions are considered different as shown below.



A subsequent update will remove selections for the FactRate table.

Include cultures

Excludes cultures from comparison if unchecked.

For culture updates, merge translations (not replace)

The same principle is applied as when merging perspective selections. When the merge translations checkbox is checked, existing translations will not be removed. Different translations for an object property that is both in the source and target cultures will be overwritten.

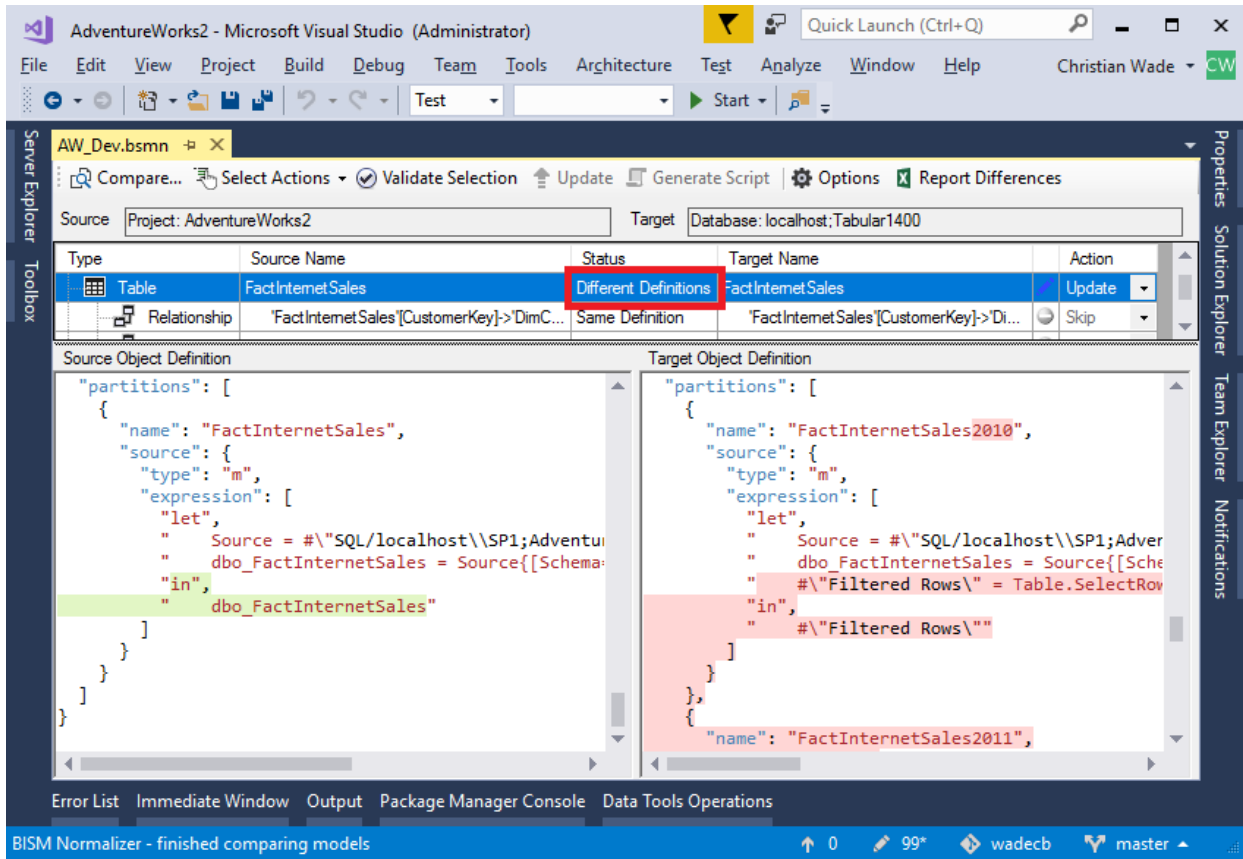
Include roles

Excludes roles from comparison if unchecked.

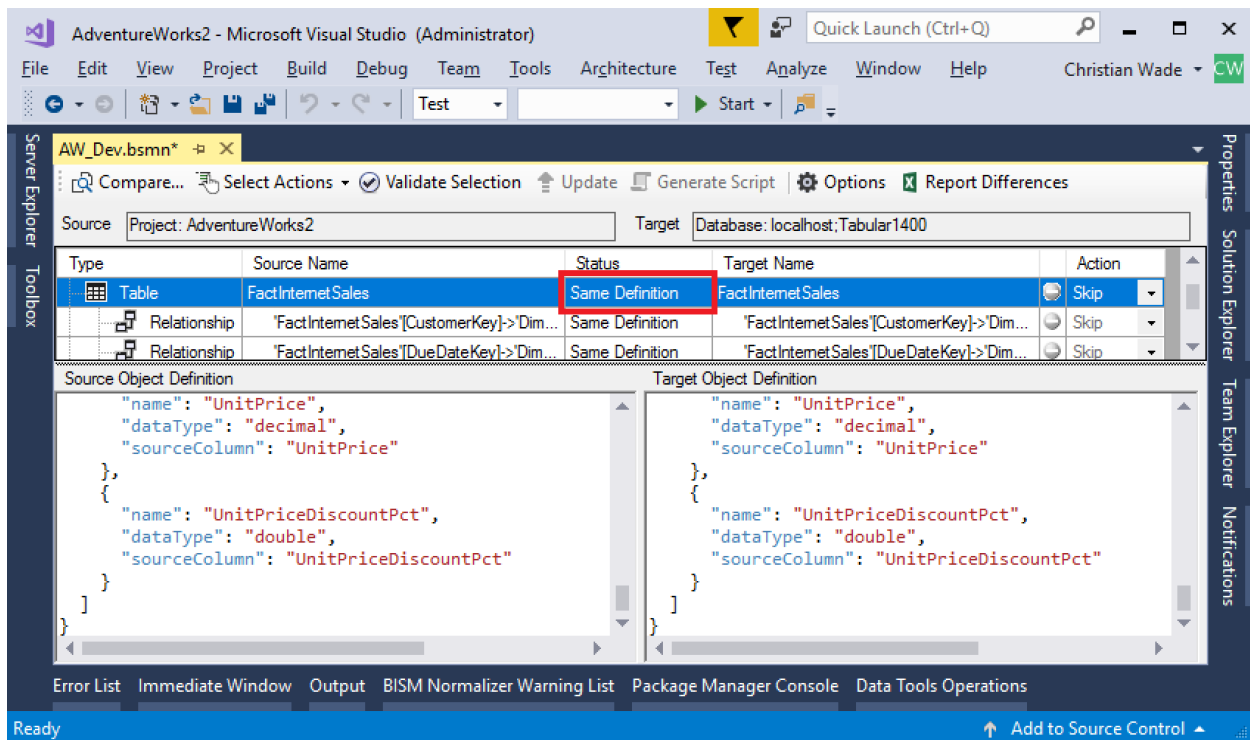
Consider partitions when comparing tables

When unchecked, partitions definitions are not included in the JSON object definitions for tables. This means that tables with different partitions – but otherwise same definitions – are considered equal and automatically skipped.

In the following example, the FactInternetSales tables in the source and target have different partition definitions, and all other properties are the same. With this option checked, the partition definitions are displayed, and the tables are considered to have different definitions.



With this option unchecked, partitions are not included in the JSON object definitions, and the tables are considered to have the same definition.

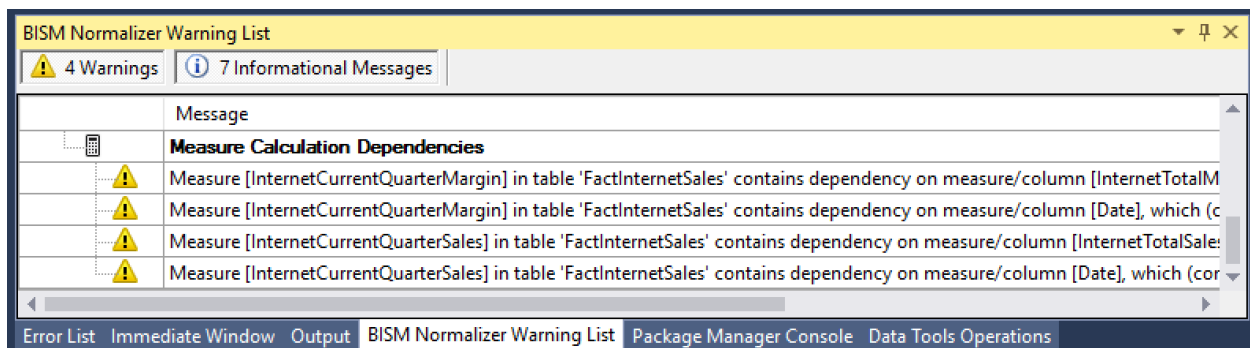


For table updates, retain partitions (not replace)

In some cases, it may be necessary to perform an update on a table and still retain partitions in the target. For example, specifying a display folder has no structural impact on the list of columns and does not require rebuilding partitions. This option retains target partitions when checked even for table updates.

Display warnings for measure dependencies (DAX reference to missing measure/column)

When checked, warnings are displayed for measures and KPIs that reference nonexistent columns or measures. This option does not affect whether actions take place. This is in contrast with other warnings – for example, invalid dependencies on M expressions and data sources – that do prevent actions from taking place



Database deployment

Database deployment options only apply when the target is a database on a server, not a project in SSDT. Data source credentials are prompted for to perform processing. These options have no effect when running in command-line mode.

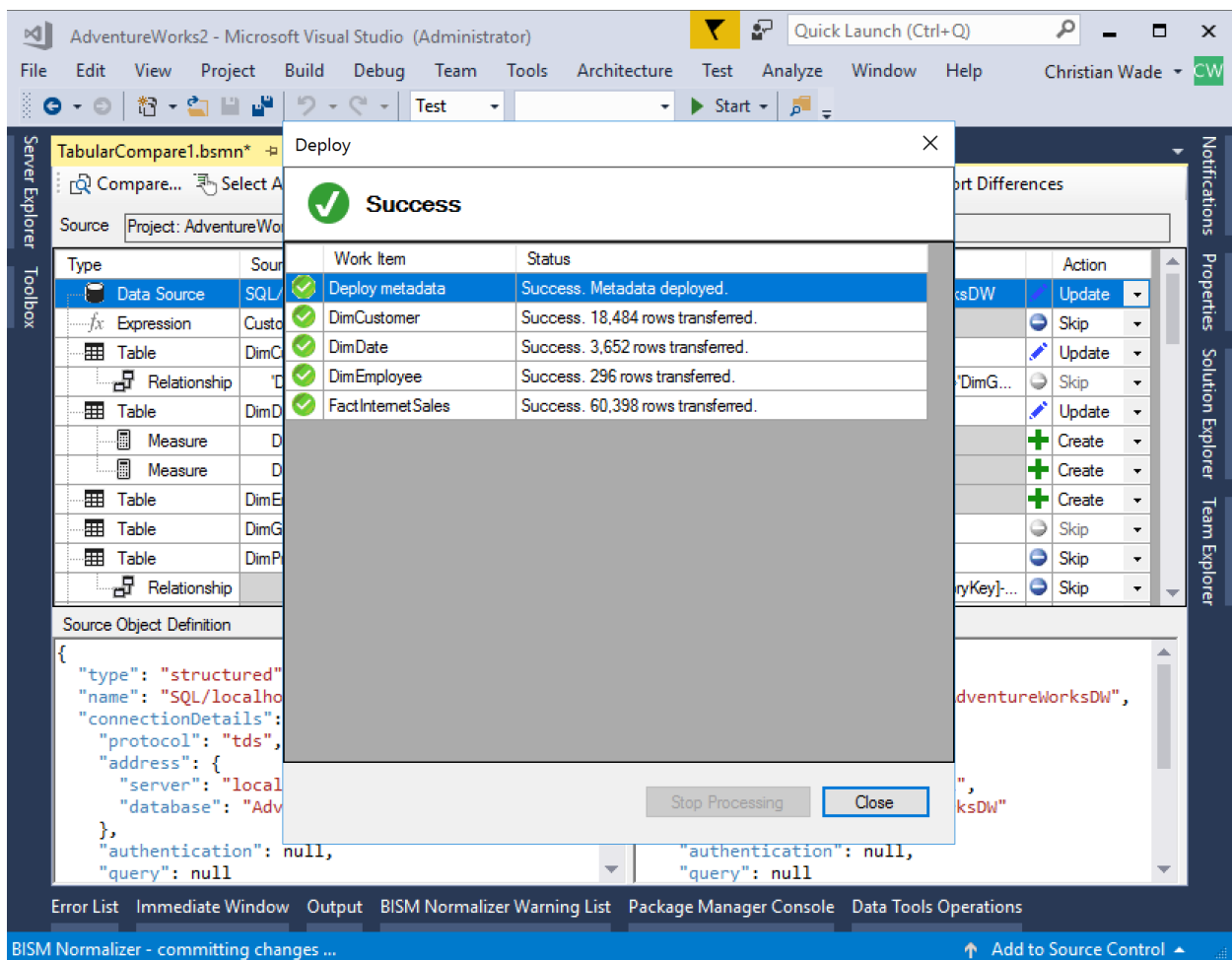
Processing option

Values to be selected are:

- **Default.** Applies Process Default as documented [here](#).
- **Do Not Process.** No processing is performed.
- **Full.** Applies Process Full as documented [here](#).

Process only affected tables

When checked, only tables affected by the comparison with Create or Update actions are processed.



Command-line execution

Run BISM Normalizer from the command line passing the BSMN file as an argument. This allows integration with automated builds and DevOps processes.

Syntax

```
BismNormalizer.exe BsmnFile [/Log:LogFile] [/Script:ScriptFile]
  [/Skip:{MissingInSource | MissingInTarget | DifferentDefinitions}]
```

Arguments

BsmnFile

Full path to the .bsmn file.

/Log:LogFile

All messages are output to LogFile. If the log file already exists, the contents will be replaced.

/Script:ScriptFile

Does not perform actual update to target database; instead, a deployment script is generated and stored to ScriptFile.

/Skip:{MissingInSource | MissingInTarget | DifferentDefinitions}

Skip all objects that are missing in source, missing in target, or with different definitions. This is in addition to the skip actions already defined in the BSMN file.

It is possible to pass a comma-separated list of multiple skip options. For example, “/Skip:MissingInSource,DifferentDefinitions” will skip all objects that are missing in source and those with different definitions.

Automated merging of branches

The /Skip argument can be used for automated merging of branches. For example, by setting /Skip:MissingInSource, it is possible to create/update new/modified objects in a primary branch, without deleting existing ones.

Examples

The following example updates the target database, logging progress and error messages for later review.

```
BismNormalizer.exe TabularCompare1.bsmn /Log:log.txt
```

The following example does not update the target database. Instead, a script is generated and progress is logged.

```
BismNormalizer.exe TabularCompare1.bsmn /Log:log.txt /Script:script.xml
```

The following example updates the target database and progress is logged. None of the objects missing in source are deleted from the target model.

```
BismNormalizer.exe TabularCompare1.bsmn /Log:log.txt /Skip:MissingInSource
```

Passwords and processing

BISM Normalizer in command-line mode does not set passwords or data source credentials. They need to be set separately, either manually or securely as part of a build. Automated processing (data refresh) therefore also needs to be set up separately if required.

Executable location

BISM Normalizer only supports standard VSIX deployment with auto-updates to Visual Studio. BismNormalizer.exe is located in the extension directory, which looks like "C:\Users\XXX\AppData\Local\Microsoft\VisualStudio\15.0\Extensions\XXX\", and is shown in the log file produced by the VSIX installer (link available on the installation complete dialog). The executables in this directory can be copied to another location using XCOPY deployment.

Visual Studio projects as source/target

Despite Visual Studio not being loaded (or required) in command-line mode, projects can act as the source/target for comparisons (in addition to databases). This is useful for automated builds, which get the latest version of files from source control. Access to a workspace database is required, which can reside in Azure AS.

The project path is stored in the BSMN file. It can be changed by editing the file.

```
<ComparisonInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <ConnectionInfoSource>
    <UseProject>true</UseProject>
    <ServerName>localhost</ServerName>
    <DatabaseName>Tabular1200_wade_21527ea8-74fd-4bb6-b2e2-3d1cd3f70063</DatabaseName>
    <ProjectName>Tabular1200</ProjectName>
    <ProjectFile>C:\Projects\TabularProject1\TabularProject1.smproj</ProjectFile>
  </ConnectionInfoSource>
  ...

```

Additional Resources

Microsoft Data Insights Summit 2017 Session: Creating Enterprise Grade BI Models with Azure Analysis Services

Published June 2017. Discusses bridging the gap between self-service and corporate BI, and demonstrates BISM Normalizer merging.

<https://youtu.be/44I48ufKhOs>

Power BI Governance and Deployment Whitepaper

Published March 2016.

<https://powerbi.microsoft.com/en-us/documentation/powerbi-admin-governance/>

Create a Centralized and Decentralized Organizational Model for Business Intelligence

Published November 2014; refreshed March 2016.

Retrieved from Gartner database; "Gartner Foundational". Gartner subscription required.

<https://www.gartner.com/doc/2897718/create-centralized-decentralized-organizational-model>

BISM Normalizer 3 Demo

Published June 2016. Discusses BISM Normalizer use cases and provides a detailed demonstration.

<https://youtu.be/LZdOwfJqFrM>

Analysis Services Git Repo

BISM Normalizer is open source. The source code is available in the Analysis Services Git repo.

<https://github.com/Microsoft/Analysis-Services/tree/master/BismNormalizer>

BISM Normalizer Internals Video

Published June 2016. Covers object model and build process for BISM Normalizer.

<https://youtu.be/r3eGK-dSYuw>

BISM Normalizer website

<http://bism-normalizer.com/>