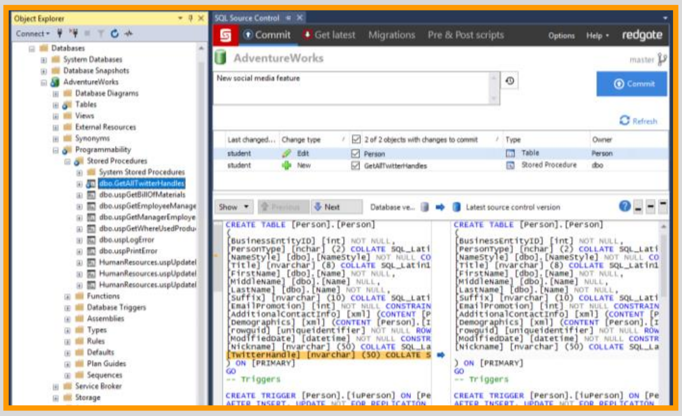


New social media feature

Last changed...	Change type	2 of 2 objects with changes to commit	Type	Owner
student	Edit	☑ Person	Table	Person
student	New	☑ GetAllTwitterHandles	Stored Procedure	dbo

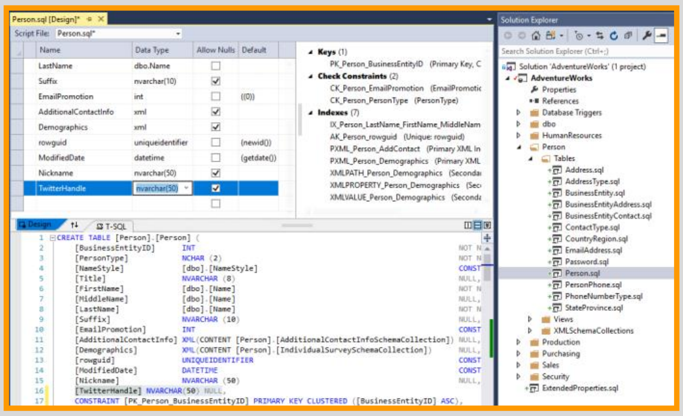
```
CREATE TABLE [Person].[Person]
(
[BusinessEntityID] [int] NOT NULL,
[PersonType] [nchar] (2) COLLATE SQL_Lati
[NameStyle] [dbo].[NameStyle] NOT NULL CO
[Title] [nvarchar] (8) COLLATE SQL_Latin1
[FirstName] [dbo].[Name] NOT NULL,
[MiddleName] [dbo].[Name] NULL,
[LastName] [dbo].[Name] NOT NULL,
[Suffix] [nvarchar] (10) COLLATE SQL_Lati
[EmailPromotion] [int] NOT NULL CONSTRAIN
[AdditionalContactInfo] [xml] (CONTENT [P
[Demographics] [xml] (CONTENT [Person].[I
[rowguid] [uniqueidentifier] NOT NULL ROW
[ModifiedDate] [datetime] NOT NULL CONSTR
[Nickname] [nvarchar] (50) COLLATE SQL_La
[TwitterHandle] [nvarchar] (50) COLLATE S
) ON [PRIMARY]
GO
-- Triggers

CREATE TRIGGER [Person].[iuPerson] ON [Pe
AFTER INSERT, UPDATE NOT FOR REPLICATION
```

```
CREATE TABLE [Person].[Person]
(
[BusinessEntityID] [int] NOT NULL,
[PersonType] [nchar] (2) COLLATE SQL_Lati
[NameStyle] [dbo].[NameStyle] NOT NULL CO
[Title] [nvarchar] (8) COLLATE SQL_Latin1
[FirstName] [dbo].[Name] NOT NULL,
[MiddleName] [dbo].[Name] NULL,
[LastName] [dbo].[Name] NOT NULL,
[Suffix] [nvarchar] (10) COLLATE SQL_Lati
[EmailPromotion] [int] NOT NULL CONSTRAIN
[AdditionalContactInfo] [xml] (CONTENT [P
[Demographics] [xml] (CONTENT [Person].[I
[rowguid] [uniqueidentifier] NOT NULL ROW
[ModifiedDate] [datetime] NOT NULL CONSTR
[Nickname] [nvarchar] (50) COLLATE SQL_La

) ON [PRIMARY]
GO
-- Triggers

CREATE TRIGGER [Person].[iuPerson] ON [Pe
AFTER INSERT, UPDATE NOT FOR REPLICATION
```


Person.sql [Design]*
Script File: Person.sql*
Keys (1)
PK_Person_BusinessEntityID (Primary Key, C
Check Constraints (2)
CK_Person_EmailPromotion (EmailPromotic
CK_Person_PersonType (PersonType)
Indexes (7)
IX_Person_LastName_FirstName_MiddleNam
AK_Person_rowguid (Unique: rowguid)
PXML_Person_AddContact (Primary XML In
PXML_Person_Demographics (Primary XML
XMLPATH_Person_Demographics (Seconda
XMLPROPERTY_Person_Demographics (Sec
XMLVALUE_Person_Demographics (Seconda
Design
T-SQL
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AdventureWorks' (1 project)
AdventureWorks
Properties
References
Database Triggers
dbo
HumanResources
Person
Tables
Address.sql
AddressType.sql
BusinessEntity.sql
BusinessEntityAddress.sql
BusinessEntityContact.sql
ContactType.sql
CountryRegion.sql
EmailAddress.sql
Password.sql
Person.sql
PersonPhone.sql
PhoneNumberType.sql
StateProvince.sql
Views
XMLSchemaCollections
Production
Purchasing
Sales
Security
ExtendedProperties.sql

Name	Data Type	Allow Nulls	Default
LastName	dbo.Name	☐	
Suffix	nvarchar(10)	✓	
EmailPromotion	int	☐	((0))
AdditionalContactInfo	xml	✓	
Demographics	xml	✓	
rowguid	uniqueidentifier	☐	(newid())
ModifiedDate	datetime	☐	(getdate())
Nickname	nvarchar(50)	✓	
TwitterHandle	nvarchar(50)	✓	
		☐	

```
CREATE TABLE [Person].[Person] (
    [BusinessEntityID]      INT                                                             NOT N
    [PersonType]            NCHAR (2)                                                       NOT N
    [NameStyle]             [dbo].[NameStyle]                                               CONST
    [Title]                 NVARCHAR (8)                                                    NULL,
    [FirstName]             [dbo].[Name]                                                    NOT N
    [MiddleName]            [dbo].[Name]                                                    NULL,
    [LastName]              [dbo].[Name]                                                    NOT N
    [Suffix]                NVARCHAR (10)                                                   NULL,
    [EmailPromotion]        INT                                                             CONST
    [AdditionalContactInfo] XML(CONTENT [Person].[AdditionalContactInfoSchemaCollection])  NULL,
    [Demographics]          XML(CONTENT [Person].[IndividualSurveySchemaCollection])       NULL,
    [rowguid]               UNIQUEIDENTIFIER                                                CONST
    [ModifiedDate]          DATETIME                                                        CONST
    [Nickname]              NVARCHAR (50)                                                   NULL,
    [TwitterHandle] NVARCHAR(50) NULL,
    CONSTRAINT [PK_Person_BusinessEntityID] PRIMARY KEY CLUSTERED ([BusinessEntityID] ASC),
```

WHITEPAPER

SQL Server source control and deployment

Redgate SQL Source Control or Microsoft SQL Server Data Tools (SSDT)?

DLM Consultants

The Agile, Continuous Delivery and **DevOps** movements have taught software development teams the importance of mature source control and release automation processes and their relevance to time to market, innovation, quality, reliability, efficiency and, ultimately, **business success**.

These lessons have been reinforced by the annual Puppet State of DevOps reports, which in 2018 specifically called out the database as being a **key differentiator** between high performers and low performers.

*"Shifting left is about bringing more teams into the development and delivery process — for example, quality, security, **database**, audit and networking. Most teams begin the leftward shift by addressing deployment pain, which is the functional boundary between Dev and Ops."*
2018 State of DevOps Report

There are two strategies for database source control and deployment: "model"/"state" and "migrations". Each has pros and cons. You can read more about the two approaches here:

workingwithdevs.com/delivering-databases-migrations-vs-state

DevOps is about people first and tooling second. However, whichever approach you decide to use, and especially if you choose the model approach, the right tooling can help.

Microsoft Data Platform MVP, Alex Yates, has extensive experience in the field of database DevOps using both the model- and the migration-based approach and a wide variety of tools. In this whitepaper he reviews the relative strengths and weaknesses of North America and Europe's two **most trusted** model-based source control and deployment tools for SQL Server databases:

Redgate SQL Source Control and **Microsoft SQL Server Data Tools**.

Contents

About the tools

Microsoft SQL Server Data Tools (SSDT)

SQL Server Data Tools, henceforth referred to as SSDT, refers to a collection of tools and features that can be included with Visual Studio to support the development and deployment of SQL Server databases. SSDT is included with any Visual Studio subscription and, for many, is the default way to source control and deploy SQL Server databases.

visualstudio.microsoft.com/vs/features/ssdt

Users will create a "Database Project" in Visual Studio, which they can compile using MSBuild. This allows developers to work on database code right alongside applications and other associated code in one place. However, since most full-time database folks spend the majority of their time in SQL Server Management Studio, henceforth referred to as SSMS, this may require them to learn a new IDE.

Database Projects can be checked into source control. When compiled a Data-tier Application (DAC) package, or DACPAC, is created which can be deployed either using SqlPackage.exe or, increasingly, dbatools – an open source PowerShell cmdlet library. Both options use the DacFx Framework to generate and deploy update scripts against target databases.

Redgate SQL Source Control

SQL Source Control is a third-party tool from Redgate Software. To get the most out of SQL Source Control, it is recommended to purchase the full SQL Toolbelt licence since this also includes SQL Compare and SQL Change Automation which are often used alongside SQL Source Control to enable database comparisons, deployment and continuous delivery. Redgate offer a free trial of SQL Source Control and all the other tools included in the SQL Toolbelt.

red-gate.com/products/sql-development/sql-source-control

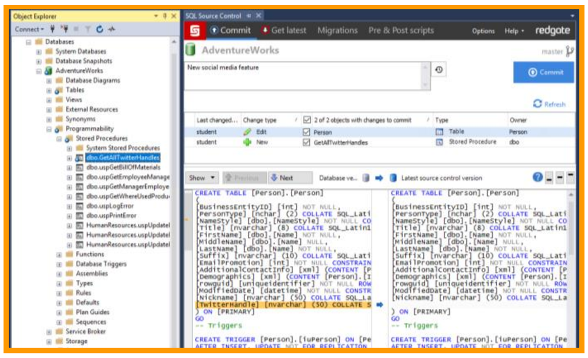
SQL Source Control is a plug-in to SSMS that aims to make database source control as easy as possible for database professionals, allowing them to perform regular tasks, such as committing to source control, without leaving their main IDE.

Redgate Software is based in Cambridge, UK and is the creator of SQL Compare which has been the most trusted schema comparison engine for SQL Server since 1999. The SQL Compare engine is used within SQL Source Control to compare different versions and to update development databases.

Ease of use

Implementing database source control, continuous integration or continuous delivery is a big change for your team and it will be hard on your developers at first. You will move more slowly in the early stages. Choosing a tool that is easy to adopt will make your life much easier in those critical early days and lead to a far greater chance of success.

Redgate SQL Source Control plugs right into SQL Server Management Studio (SSMS), whereas SSDT lives in Visual Studio.



New social media feature

Last changed...	Change type	2 of 2 objects with changes to commit	Type	Owner
student	Edit	☑ Person	Table	Person
student	New	☑ GetAllTwitterHandles	Stored Procedure	dbo

```
CREATE TABLE [Person].[Person]
(
[BusinessEntityID] [int] NOT NULL,
[PersonType] [nchar] (2) COLLATE SQL_Lati
[NameStyle] [dbo].[NameStyle] NOT NULL CO
[Title] [nvarchar] (8) COLLATE SQL_Latin1
[FirstName] [dbo].[Name] NOT NULL,
[MiddleName] [dbo].[Name] NULL,
[LastName] [dbo].[Name] NOT NULL,
[Suffix] [nvarchar] (10) COLLATE SQL_Lati
[EmailPromotion] [int] NOT NULL CONSTRAIN
[AdditionalContactInfo] [xml] (CONTENT [P
[Demographics] [xml] (CONTENT [Person].[I
[rowguid] [uniqueidentifier] NOT NULL ROW
[ModifiedDate] [datetime] NOT NULL CONSTR
[Nickname] [nvarchar] (50) COLLATE SQL_La
[TwitterHandle] [nvarchar] (50) COLLATE S
) ON [PRIMARY]
GO
-- Triggers

CREATE TRIGGER [Person].[iuPerson] ON [Pe
AFTER INSERT, UPDATE NOT FOR REPLICATION
```

```
CREATE TABLE [Person].[Person]
(
[BusinessEntityID] [int] NOT NULL,
[PersonType] [nchar] (2) COLLATE SQL_Lati
[NameStyle] [dbo].[NameStyle] NOT NULL CO
[Title] [nvarchar] (8) COLLATE SQL_Latin1
[FirstName] [dbo].[Name] NOT NULL,
[MiddleName] [dbo].[Name] NULL,
[LastName] [dbo].[Name] NOT NULL,
[Suffix] [nvarchar] (10) COLLATE SQL_Lati
[EmailPromotion] [int] NOT NULL CONSTRAIN
[AdditionalContactInfo] [xml] (CONTENT [P
[Demographics] [xml] (CONTENT [Person].[I
[rowguid] [uniqueidentifier] NOT NULL ROW
[ModifiedDate] [datetime] NOT NULL CONSTR
[Nickname] [nvarchar] (50) COLLATE SQL_La

) ON [PRIMARY]
GO
-- Triggers

CREATE TRIGGER [Person].[iuPerson] ON [Pe
AFTER INSERT, UPDATE NOT FOR REPLICATION
```

Figure 1 SQL Source Control plugs right into SSMS

This is enormous. If your developers and DBAs prefer to work in SSMS, SQL Source Control may feel very natural to them, but Visual Studio is full of new concepts: Solutions files? Project files? MSBuild? The list goes on. This makes the learning curve much more difficult for people who are used to working in SSMS.

To demonstrate further, read Bob Walker's blog post about why they chose to pay for Redgate licences over using SSDT for free:

codeaperture.io/2017/08/08/redgate-over-ssdt-my-biased-opinion

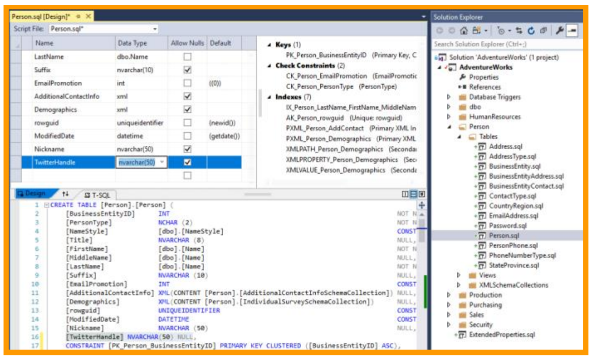
On the other hand, if your team are already familiar with Visual Studio and .NET projects this isn't as big a deal because SSDT will probably feel pretty natural to them. Just remember that whether you choose SSDT or Redgate, you need to do it as a team. Everyone will need to interact with it.

Even if you do love Visual Studio, if you still like to code against a real database the process of getting your changes in and out of your database project is a little cumbersome. You end up using schema compare which is unaware of which changes were made in your project and which were made on your database.

In contrast, Redgate SQL Source Control does a three way compare behind the scenes to work out what changes need to move from your source code to your dev database and vice versa. You can read more about that here:

documentation.red-gate.com/soc7/reference-information/how-sql-source-control-works-behind-the-scenes

It's not entirely one-sided, however. The SSDT table designer is amazing, combining the SSMS table designer and a query window and allowing you to use whichever you prefer.



Name	Data Type	Allow Nulls	Default
LastName	dbo.Name	☐	
Suffix	nvarchar(10)	✓	
EmailPromotion	int	☐	((0))
AdditionalContactInfo	xml	✓	
Demographics	xml	✓	
rowguid	uniqueidentifier	☐	(newid())
ModifiedDate	datetime	☐	(getdate())
Nickname	nvarchar(50)	✓	
TwitterHandle	nvarchar(50)	✓	
		☐	

Keys (1)
- PK_Person_BusinessEntityID (Primary Key, C

Check Constraints (2)
- CK_Person_EmailPromotion (EmailPromotic
- CK_Person_PersonType (PersonType)

Indexes (7)
- IX_Person_LastName_FirstName_MiddleNam
- AK_Person_rowguid (Unique: rowguid)
- PXML_Person_AddContact (Primary XML In
- PXML_Person_Demographics (Primary XML
- XMLPATH_Person_Demographics (Seconda
- XMLPROPERTY_Person_Demographics (Sec
- XMLVALUE_Person_Demographics (Seconda

```
CREATE TABLE [Person].[Person] (
    [BusinessEntityID]      INT                                                              NOT N
    [PersonType]            NCHAR (2)                                                        NOT N
    [NameStyle]             [dbo].[NameStyle]                                                CONST
    [Title]                 NVARCHAR (8)                                                     NULL,
    [FirstName]             [dbo].[Name]                                                     NOT N
    [MiddleName]            [dbo].[Name]                                                     NULL,
    [LastName]              [dbo].[Name]                                                     NOT N
    [Suffix]                NVARCHAR (10)                                                    NULL,
    [EmailPromotion]        INT                                                              CONST
    [AdditionalContactInfo] XML(CONTENT [Person].[AdditionalContactInfoSchemaCollection])   NULL,
    [Demographics]          XML(CONTENT [Person].[IndividualSurveySchemaCollection])        NULL,
    [rowguid]               UNIQUEIDENTIFIER                                                 CONST
    [ModifiedDate]          DATETIME                                                         CONST
    [Nickname]              NVARCHAR (50)                                                    NULL,
    [TwitterHandle] NVARCHAR(50) NULL,
    CONSTRAINT [PK_Person_BusinessEntityID] PRIMARY KEY CLUSTERED ([BusinessEntityID] ASC),
```

Figure 2 The SSDT table designer in Visual Studio

On balance, SQL Source Control is generally a more natural tool for most database folks.

Winner: SQL Source Control

Refactoring

This whitepaper is scoped to model-based source control solutions. The main problem with model-based source control solutions is that you lose some control over the update scripts applied to your target databases because they are typically generated by software. This can be a problem with certain types of changes.

When deployments scripts are generated by schema comparison tools, which do not look at or understand your data, there are some scenarios which can cause the software problems. For example:

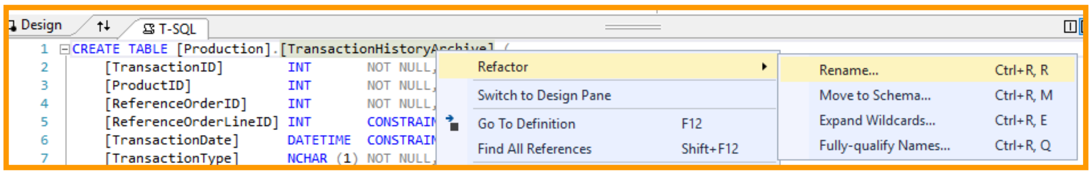
- Renaming a column or table
- Splitting a table
- Adding a new NOT NULL column
- Moving a table to a different schema
- Updating data

Often, if left unchecked and untested, software can generate poorly optimised scripts or scripts that accidentally drop data or fail to handle NOT NULL constraints - causing deployment failures. It is important to understand this and have a plan in place for how to avoid these problems.

Both SSDT and Redgate SQL Source Control have features to help you in these scenarios.

SSDT has two relevant features:

- The **Refactor Log** keeps track of refactors and the DacFx framework (which generates the upgrade script) understands how to interpret this information and create an appropriate upgrade script. That makes most changes as simple as “right-click, refactor”. However, the Refactor Log is limited and does not cover all scenarios.

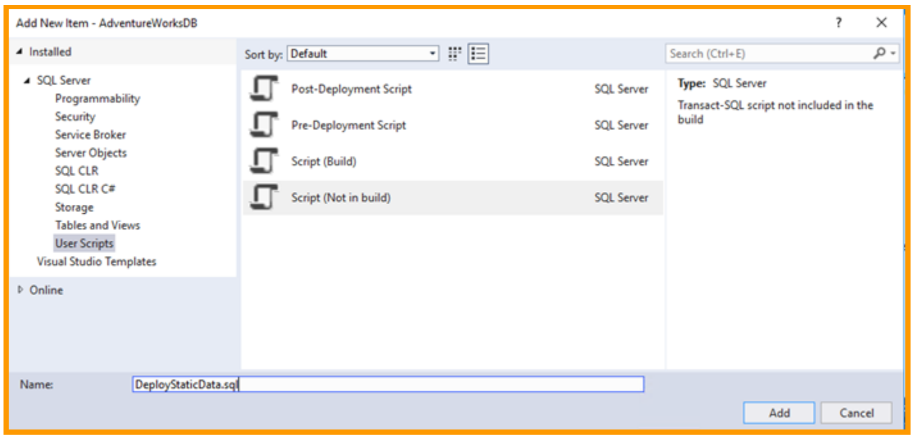


Design
T-SQL
Refactor
Switch to Design Pane
Go To Definition F12
Find All References Shift+F12
Rename... Ctrl+R, R
Move to Schema... Ctrl+R, M
Expand Wildcards... Ctrl+R, E
Fully-qualify Names... Ctrl+R, Q

```
CREATE TABLE [Production].[TransactionHistoryArchive] (
    [TransactionID]        INT        NOT NULL,
    [ProductID]            INT        NOT NULL,
    [ReferenceOrderID]     INT        NOT NULL,
    [ReferenceOrderLineID] INT        CONSTRAIN
    [TransactionDate]      DATETIME   CONSTRAIN
    [TransactionType]      NCHAR (1)  NOT NULL,
```

Figure 3 SSDT enables "Right-click Refactor" and records the refactor in the Refactor Log

- For the scenarios not covered by the Refactor Log, SSDT provides **Pre- and Post-deployment Scripts**. These are powerful but over time do become cumbersome and annoying to manage.



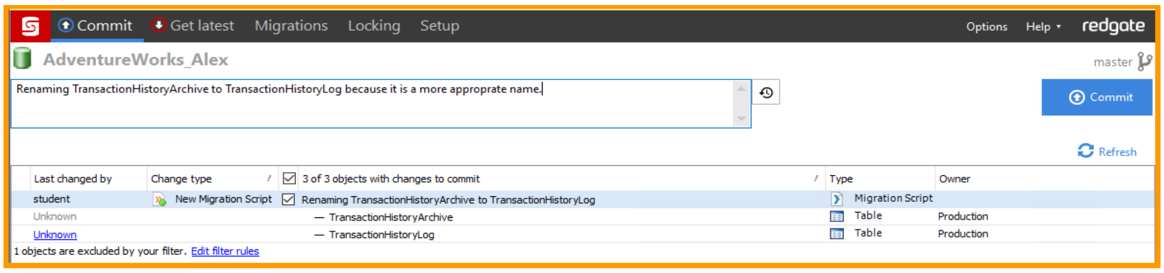
Type: SQL Server
Transact-SQL script not included in the build

Name: DeployStaticData.sql

Figure 4 SSDT provides Pre- and Post-Deployment scripts as a "catch all" to handle any other complex refactoring tasks

Redgate also has two relevant features:

- SQL Source Control **Migration Scripts** were for a long time seen as Redgate's killer feature that blew SSDT out of the water. Instead of a limited and auto-generated Refactor Log or a cumbersome Post-deployment Script, users were able to add their own custom script to handle complicated refactors as and when required. This would give them more control and confidence. Unfortunately, this feature never fully delivered on its promises. If anything, it's too clever, so people find it hard to understand and when they make mistakes they can break things pretty badly. Migration scripts can also have a nasty impact on performance. DLM Consultants do not advise you use them.



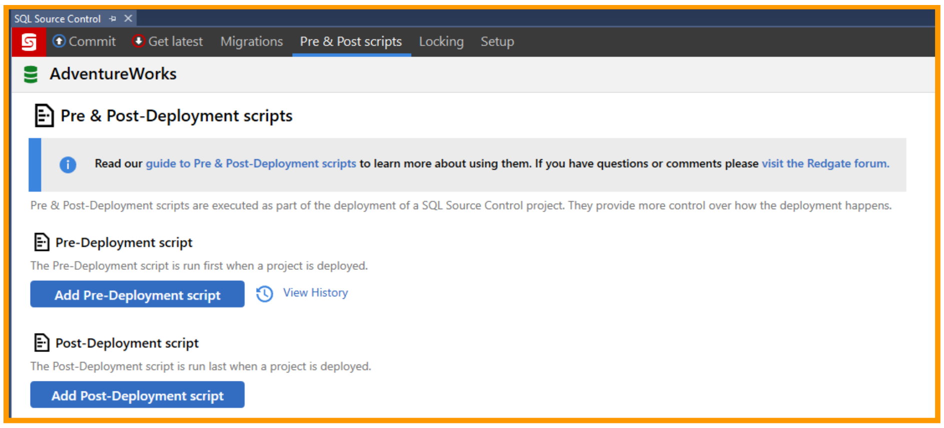
Renaming TransactionHistoryArchive to TransactionHistoryLog because it is a more appropate name.

Last changed by	Change type	3 of 3 objects with changes to commit	Type	Owner
student	New Migration Script	Renaming TransactionHistoryArchive to TransactionHistoryLog	Migration Script	
Unknown		— TransactionHistoryArchive	Table	Production
Unknown		— TransactionHistoryLog	Table	Production

1 objects are excluded by your filter. Edit filter rules

Figure 5 SQL Source Control Migration Scripts are elegant in theory, but problematic in practice

- Hot off the press, Redgate have just released **Pre and Post Scripts** for SQL Source Control which broadly match the SSDT Pre- and Post-deployment Scripts. (See version 7.0.0.8628, released on Nov 20th 2018.) This is excellent news because it means we no longer need to use SQL Source Control migration scripts and it goes some way to catching up with SSDT. However, these scripts are not as fully featured as the SSDT scripts. They don't, for example, support SQLCMD syntax and variables. This means that, unlike SSDT, you cannot run a script and apply the "DEV" or "PROD" variable to support conditional logic based on environment. Instead you need to either rely on hardcoding @@SERVERNAME or maintaining a config table, but neither of these workarounds are as elegant as the SSDT approach.



Pre & Post-Deployment scripts

Read our guide to Pre & Post-Deployment scripts to learn more about using them. If you have questions or comments please visit the Redgate forum.

Pre & Post-Deployment scripts are executed as part of the deployment of a SQL Source Control project. They provide more control over how the deployment happens.

Pre-Deployment script

The Pre-Deployment script is run first when a project is deployed.

Post-Deployment script

The Post-Deployment script is run last when a project is deployed.

Figure 6 SQL Source Control now supports Pre and Post Scripts, very similar to SSDT Pre- and Post-deployment Scripts

Based on the features above, it's a clear win for SSDT. That said, if complicated refactors are a regular and significant problem for you, you should probably consider moving to a full migrations solution.

If you use either SSDT or Redgate SQL Source Control, the most natural migrations-based tool for you to switch to is the Redgate SQL Change Automation Visual Studio extension (previously called ReadyRoll). Even Microsoft are recommending it for folks who want to "extend DevOps practices to SQL Server":

blogs.msdn.microsoft.com/visualstudio/2017/03/07/redgate-data-tools-in-visual-studio-2017

SQL Change Automation is covered by the same SQL Toolbelt licence as SQL Source Control. If you can't afford the full version there is a free version (which lacks some of the best features) that comes with Visual Studio 2017 Enterprise. Alternatively, you could consider DbUp, which is open source and free but even more limited.

dbup.github.io

A word of caution. Before moving to a migrations-based solution, you should consider that you are swapping one set of problems for another. If complex migrations are only an occasional problem for you, the 80-20 rule probably applies. A model-based source control approach often solves 80% of the problems with 20% of the effort. For more information on model- and migration-based solutions, read the blog post:

workingwithdevs.com/delivering-databases-migrations-vs-state

Winner: SSDT

DLM Consultants

Filters

Perhaps you aren't interested in any objects prefixed with "test_" or suffixed with "_toDelete". Or perhaps you want to filter out some other objects that are managed by a different team or which are supposed to exist in your dev environment, but not production, such as the tSQLt unit testing framework?

tsqlt.org

The SQL Source Control "Filters" feature makes this straight-forward. Whether you want to filter out something small (like the IsSqlCloneDatabase extended property added by SQL Clone) or whether you want to set up your own general naming conventions, you should be able to set up a filter to achieve your goal.

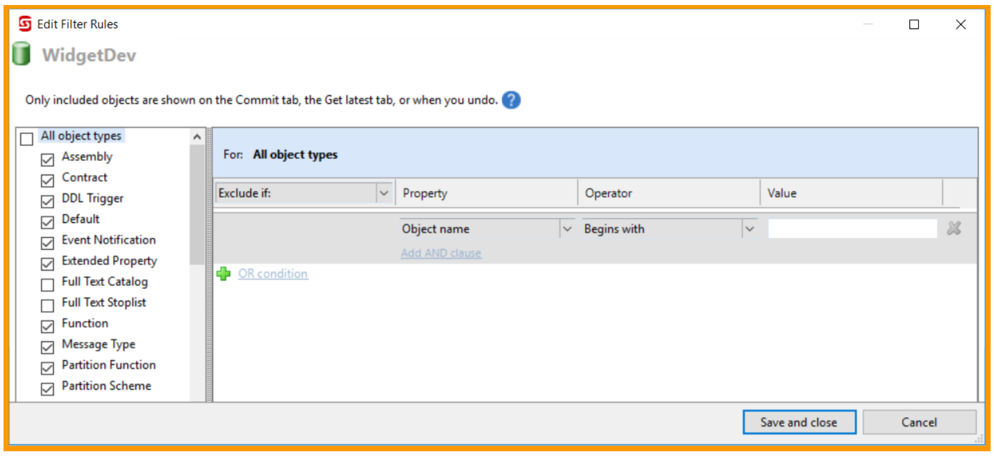


Figure 7 Redgate SQL Source Control filters are effective and easy to use

You can even use them to do more complex tasks, such as managing multiple databases with several known variations, but I'd generally discourage going down that road if you can:

red-gate.com/hub/product-learning/sql-compare/how-to-build-multiple-database-versions-from-the-same-source-using-sql-compare-filters

Redgate uses this filter functionality under the hood, with a default option to include and/or exclude the entire tSQLt framework from your deployments depending on the context. In contrast, SSDT doesn't have anything like this, and according to this uservoice entry it doesn't look like they plan to implement something similar any time soon either.

visualstudio.uservoice.com/forums/121579-visual-studio-ide/suggestions/3232905-add-object-name-filtering-to-schema-compare

To demonstrate the implications of this, to use tSQLt with SSDT you have to create multiple dependent projects with database references and use Pre- and Post-deployment Scripts to add your tests to source control but filter them out of your deployments. While some people like this as it keeps their database code and their tests separated, others find it more complicated.

kzhendev.wordpress.com/2014/01/08/setting-up-ssdt-database-projects-and-tsqlt

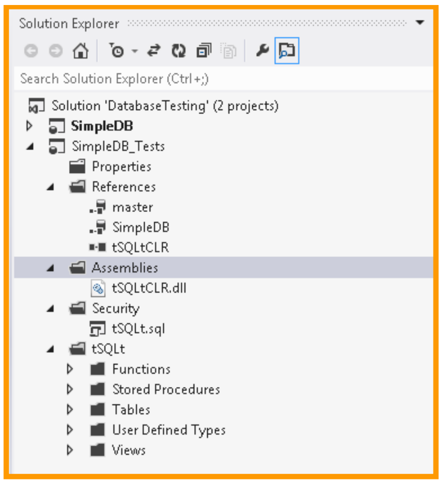


Figure 8 Ken Ross' SampleDB SSDT project including the tSQLt unit testing framework

To get around this limitation in SSDT, Ed Elliot has created a Filter Dacpac Deployments contributor, which you can add to your project to enable similar functionality in SSDT. This is an excellent contribution to SSDT, but if your database folks aren't comfortable writing C# they will find it more complicated to configure.

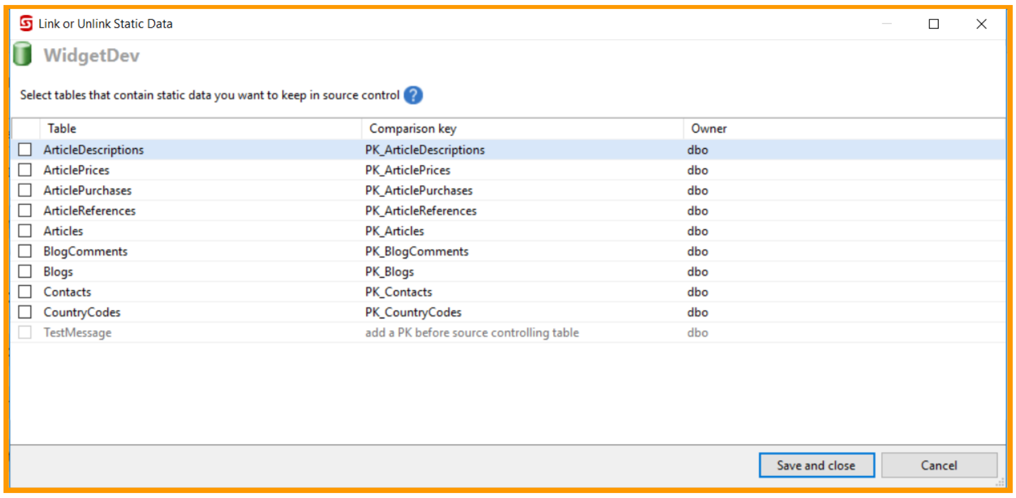
the.agilesql.club/2015/01/howto-filter-dacpac-deployments

Winner: SQL Source Control

Static data

Most databases contain static data tables. These tables contain relatively few rows that need to be maintained by developers. Lookup codes and other reference data for example. This data should be kept in source control alongside your schema.

Redgate SQL Source Control makes adding static data to source control easy – just right-click and select the tables you want and voila.



Select tables that contain static data you want to keep in source control

Table	Comparison key	Owner
☐ ArticleDescriptions	PK_ArticleDescriptions	dbo
☐ ArticlePrices	PK_ArticlePrices	dbo
☐ ArticlePurchases	PK_ArticlePurchases	dbo
☐ ArticleReferences	PK_ArticleReferences	dbo
☐ Articles	PK_Articles	dbo
☐ BlogComments	PK_BlogComments	dbo
☐ Blogs	PK_Blogs	dbo
☐ Contacts	PK_Contacts	dbo
☐ CountryCodes	PK_CountryCodes	dbo
☐ TestMessage	add a PK before source controlling table	dbo

Figure 9 SQL Source Control allows you to select which data you would like to add to source control

However, there are a few issues with it.

- All static data tables require a primary key (although they should generally have one anyway).

- The scripts generated are not easily ‘diff-able’ because changes to the data do not appear in-line.
- Column-level or row-level versioning, (e.g. to support seed data), is not supported. It’s all or nothing.
- Adding lots of static data can have a negative impact on performance.

In SSDT static data is handled using Post-deployment Scripts and MERGE statements:

blogs.msdn.microsoft.com/ssdt/2012/02/02/including-data-in-a-sql-server-database-project

Post-deployment Scripts can be a pain to manage and the MERGE scripts can be annoying to write by hand, but these pains can be minimised by separating out the MERGE scripts into helper scripts and/or stored procedures that are referenced/executed by a coordinating Post-deployment Script, and by using sp_generate_merge to create your merge scripts:

github.com/readyroll/generate-sql-merge/blob/master/master.dbo.sp_generate_merge.sql

If you can get over the pain, SSDT allows you to create data scripts that are “diff-able” and with far greater flexibility, for example to support seed data.

This makes picking a winner difficult. Redgate SQL Source Control is certainly easier to use and for most people it will do the job really well. The SSDT solution, while more complicated and a pain to manage, is more flexible and supports more use cases.

Until Redgate released Pre and Post Scripts (Nov 20th 2018, see the *Refactoring* section of this whitepaper for details) this author was siding with SSDT. However, now Redgate has Post Scripts that does tip things back towards Redgate because it is possible to use the SSDT approach with Redgate SQL Source Control where required or preferred.

That makes Redgate the winner – but it’s a close one.

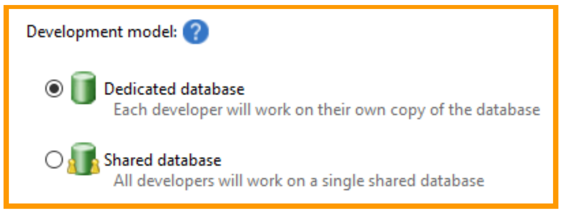
Winner: SQL Source Control

Shared development databases

While the author understands that many teams do use shared development databases, this is an antipattern that should be avoided in almost all circumstances.

workingwithdevs.com/shared-vs-dedicated

However, SQL Source Control has first class support for shared development databases, including a nifty Object Locking feature. SSDT doesn't. That's not to say it's impossible with SSDT, but it's much more complicated.



Development model:

(●) Dedicated database
Each developer will work on their own copy of the database

( ) Shared database
All developers will work on a single shared database

Figure 10 SQL Source Control can support both a "Dedicated database" mode and a "Shared database" mode

While some might argue that SSDT is right to make it harder for people to use shared databases, that wouldn't be very useful to folks who are lumbered with one regardless.

While DLM Consultants remain opposed to shared databases, if you have a shared development database, and are not able to move to the dedicated model, SQL Source Control does provide better support. (Unless you use Git. But using a distributed source control system with a shared development database is a plain contradiction and should not be attempted.)

Winner: SQL Source Control – but try to avoid using a shared database if possible

Build

Before one can deploy either a SQL Source Control or SSDT project, it must be "built" or "compiled". This process validates the syntax and referential integrity of the source code and generates a package which can be used as a deployment artifact.

Running a build in SSDT is as simple as clicking the build button or running MSBuild against your project or solution file and a DACPAC package is created. This can be done on a developer's machine or a build server relatively easily. The DACPAC can later be deployed with SQLPackage.exe or PowerShell scripts, often using dbatools, both of which call the DacFx framework.

docs.microsoft.com/en-us/sql/tools/sqlpackage

With Redgate it's a bit more complicated. Developers can run a build using Redgate SQL Change Automation. (Previously known as DLM Automation – this gets a bit complicated because Redgate recently merged ReadyRoll, their migration-based Visual Studio plugin, and DLM Automation, the PowerShell deployment cmdlets to build and deploy model-based SQL Source Control projects. "SQL Change Automation" now refers to both - which makes technical conversations somewhat clunky.)

forum.red-gate.com/discussion/83543/sql-change-automation-has-been-released

Once you have run your build on your SQL Source Control project you can create a NuGet package which can be deployed using SQL Change Automation.

documentation.red-gate.com/sca3/reference/powershell-cmdlets

SQL Change Automation builds work differently to SSDT builds. Where SSDT is compiled in memory by MSBuild, SQL Change Automation fully deploys your source code against a temporary database. This database can either be created in localDB (if localDB can support your source code and you don’t have any dependency issues) or on a full SQL Server instance (if, for example, you have dependencies on other databases or you need to set up filegroups etc).

Some argue that the Redgate approach is more realistic, but it can also be more effort to set up. So long as your database can be built on localDB it might not be too challenging, but if you end up needing to support a complicated integration environment with many dependent databases then SSDT immediately becomes more attractive.

Additionally, SSDT makes it very easy to build your database from Visual Studio, but Redgate have not brought the build action into SSMS. This means developers are not encouraged to build their source code until it hits the CI/Build server. While I encourage SQL Source Control users to maintain a build.ps1 script in their source control repo to enable developers to compile their code on their dev machine, developers are far less likely to compile their code often if there isn’t a button right there in the IDE. (Yes, you could ask users to add their own buttons in SSMS to run a PowerShell script, but they shouldn’t have too.)

SSDT also provides a much richer set of warnings etc both within the IDE and when you run a build. It can even be set to support a particular SQL Server edition, so that if you start using SQL Server 2016 features in SSDT but your production databases are only at SQL Server 2012, your builds will start failing.

Redgate’s main redeeming factor is that there are various plugins available to the market leading build servers, including Azure DevOps Services, TeamCity, Bamboo and Jenkins. These do make it much easier to set up an automated build process.

Winner: SSDT

Deployment

Generally, the Redgate SQL Compare engine is more highly regarded than Microsoft's DacFx Framework and comes with more options and features – but there isn't a lot in it.

When you deploy a DACPAC you have limited options:

- Generate a script and save it for me to run manually
- Generate a script and run it automatically

Neither are really what is required. Instead, we should deploy first to a pre-production or staging environment. Then, if that goes well, we should run exactly the same script against production. The process should be smart enough to abort if the tool detects schema drift, where someone has made a "hotfix" change on the target database. Any differences between the production and pre-production environments would mean that re-using the same script will no longer get to the expected state.

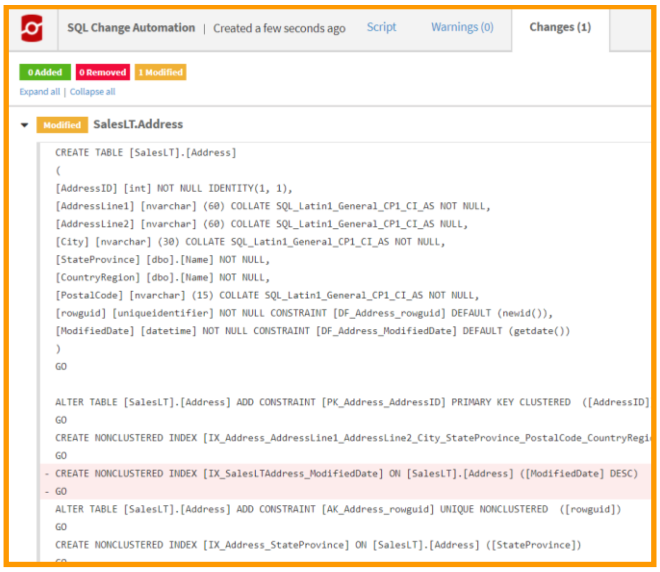
Redgate SQL Change Automation (the PowerShell deployment scripts, not the old ReadyRoll Visual Studio plugin) has a very impressive but often overlooked pair of cmdlets that help you do this:

New-DatabaseReleaseArtifact
documentation.red-gate.com/sca3/reference/powershell-cmdlets

Use-DatabaseReleaseArtifact
documentation.red-gate.com/sca3/reference/powershell-cmdlets/use-databasereleaseartifact

The release artifact that is created contains a copy of the before state, end state and the upgrade script to execute the deployment. This artifact can then be re-run on multiple databases, in each case verifying that the target database has not drifted from the before state, then re-using the same upgrade script to ensure reliable results. In contrast, the same behaviour in SSDT would require significant custom scripting and the development of custom release artifacts.

Also included in the Redgate release artifact is an intuitive HTML deploy report that can be used to sanity check deployments or to support approval gates etc:



```
  CREATE TABLE [SalesLT].[Address]
  (
  [AddressID] [int] NOT NULL IDENTITY(1, 1),
  [AddressLine1] [nvarchar] (60) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
  [AddressLine2] [nvarchar] (60) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
  [City] [nvarchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
  [StateProvince] [dbo].[Name] NOT NULL,
  [CountryRegion] [dbo].[Name] NOT NULL,
  [PostalCode] [nvarchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
  [rowguid] [uniqueidentifier] NOT NULL CONSTRAINT [DF_Address_rowguid] DEFAULT (newid()),
  [ModifiedDate] [datetime] NOT NULL CONSTRAINT [DF_Address_ModifiedDate] DEFAULT (getdate())
  )
  GO

  ALTER TABLE [SalesLT].[Address] ADD CONSTRAINT [PK_Address_AddressID] PRIMARY KEY CLUSTERED  ([AddressID]
  GO
  CREATE NONCLUSTERED INDEX [IX_Address_AddressLine1_AddressLine2_City_StateProvince_PostalCode_CountryRegi
  GO
- CREATE NONCLUSTERED INDEX [IX_SalesLTAddress_ModifiedDate] ON [SalesLT].[Address] ([ModifiedDate] DESC)
- GO
  ALTER TABLE [SalesLT].[Address] ADD CONSTRAINT [AK_Address_rowguid] UNIQUE NONCLUSTERED  ([rowguid])
  GO
  CREATE NONCLUSTERED INDEX [IX_Address_StateProvince] ON [SalesLT].[Address] ([StateProvince])
```

Figure 11 The Redgate SQL Change Automation deploy report artifact

While SSDT does allow users to generate deploy and a drift reports with SQLPackage.exe, they are neither as presentable nor as practical to integrate into a release pipeline as the Redgate equivalent.

Also, Redgate has a pair of extensions for Octopus Deploy and Azure DevOps Services which make it relatively easy to set up a deployment pipeline – once you have got your head around the difference between your build artifacts (NuGet packages) and your release artifacts.

SSDT, in its defence, does have a powerful API which does allow users to put develop their own deployment routines exactly as they wish. This is popular with .NET developers but for most full time SQL Server professionals the Redgate tools make it easier to set-up and maintain deployment pipelines.

Winner: SQL Source Control

Versioning security settings

Security is a tricky one. Typically, it is required to set security up differently per environment. Both SQL Source Control and SSDT have settings to filter out users and roles etc – but that's skirting around the issue. It's best to source control which users should exist in which environment and to ensure they are deployed correctly every time. Auditors love to see this sort of "security by design and default".

The best solution for this is to:

- Ensure roles are consistent in all environments and are deployed through source control.
- Have a post deploy script to run conditional logic per environment to deploy the appropriate users.

The fact that SSDT Post-deployment Scripts support SQLCMD variables makes this is relatively easy to configure in SSDT, as Peter Schott explains here:

schottsql.blogspot.com/2013/05/ssdt-setting-different-permissions-per.html

With Redgate Post Scripts, which don't support CMD variables, you would be relying on using @@SERVERNAME in your deployment scripts, referencing a config table in the database or using a complicated configuration within your release management process – all more complicated options.

Winner: SSDT

Complex projects

So what about the big, ugly, real world, monolithic databases that we all have to look after from time to time? Are the tools going to handle them? Maybe.

Neither SSDT nor Redgate can do much if your source code is already fairly broken. Don't expect either to be a perfect solution to all the problems. Sometimes the tools won't be up to the job.

For example, if you have circular cross-database dependencies, neither SSDT nor Redgate will be very forgiving. You end up in a world of pain using SSDT partial projects with database references and littering your solution with stub projects. You also need to set up and maintain complicated build environments with complicated routines to deploy your databases in the right order.

One issue you will have with SSDT is that you might struggle to even get your project to compile, and you can't really do any meaningful development work until the project compiles. SQL Source Control is a little easier to work with in that regard.

Having said that, SSDT is more likely to support all the latest SQL Server features natively, and occasionally you'll find that Redgate doesn't support the long tail of SQL Server features that not everyone uses – which is fine, unless you rely on one of those features.

Also, since SQL Source Control is a plugin to SSMS (which is 32 bit), and SSDT is a plugin to Visual Studio (which is also 32 bit) both are limited to an allowance 2GB of RAM, which is shared with the IDE. This can be upped to 3GB using Large Address Aware, but often even 3GB just won't cut it. With particularly large and/or complex databases either SSMS or Visual Studio can grind to a halt or even start crashing with Out of Memory exceptions.

Winner: Draw / Neither

Price

Redgate costs money. SSDT is free.

You can buy SQL Source Control as a standalone product, but you also want SQL Compare and SQL Change Automation, and that increases the cost because each team member will need a licence for the full SQL Toolbelt.

“Free” is always going to be more attractive than “not free”. The decision you need to make is whether the benefits listed in this post are worth the additional expense.

Winner: SSDT

Support

The documentation for both SSDT and Redgate is a bit hit and miss.

SSDT is more widely used so it’s easier to hire developers who understand it well. There are also more blogs and other community resources available for SSDT as a result.

However, Redgate SQL Source Control isn’t that far behind and the Redgate support team is widely regarded as being excellent. After all, Redgate have a vested interest in you being successful because they only get paid if you use their stuff – but Microsoft don’t mind if you use SSDT, SQL Source Control or something else. They just want you to host your applications and databases in Azure and it makes no difference to them how you get it there.

Also, the folks at DLM Consultants are always happy to help regardless of whether you choose SSDT or SQL Source Control.

Winner: Draw

Summary

Which is better? Well, like most things, it depends. Here are the winners of each category:

SSDT	**SQL Source Control**	**Draw**
Refactoring	Ease of use	Complex projects
Build	Filters	Support
Versioning security settings	Static data	
Price	Shared development databases	
	Deployment	

Redgate has a reputation for being the gold standard for database DevOps tooling, and on balance this reputation is probably deserved in most scenarios. While it is certainly not the best option in all areas, and while it does have some flaws, it is strong or the strongest in most.

SSDT on the other hand does have some advantages and, while it's not as polished as SQL Source Control, it can normally be engineered to work. For example, with Redgate, re-using a deployment script against multiple environments and checking for drift can be achieved in a few lines of PowerShell and an elegant drift report can be created. To achieve the same with SSDT requires more work, more scripting, and the report is not as attractive or detailed. Similarly, to set up a filter or to version control some data with Redgate it's a simple right-click option. To achieve the same in SSDT requires a much more convoluted solution and more developer skill.

However, it should not be ignored that Redgate isn't free. And SSDT is only free if your developers already have licences for Visual Studio (which sometimes database folks don't).

Different organisations have different budgets, and some do more SQL Server work than others, so whether your organisation can both afford the Redgate tools or the Visual Studio licences and whether it will see the return is not something this author can answer.

However, when making decisions about budgets and spending, it is important to consider the real cost of your development efforts. Don't just look at salaries. What is the consequence on employee retention and what are your recruitment and training overheads? What are the costs of managing the developers (and the managers - recursive)? What are the costs of failed deployments, bugs and delays? What are the benefits of getting to market more quickly and reliably?

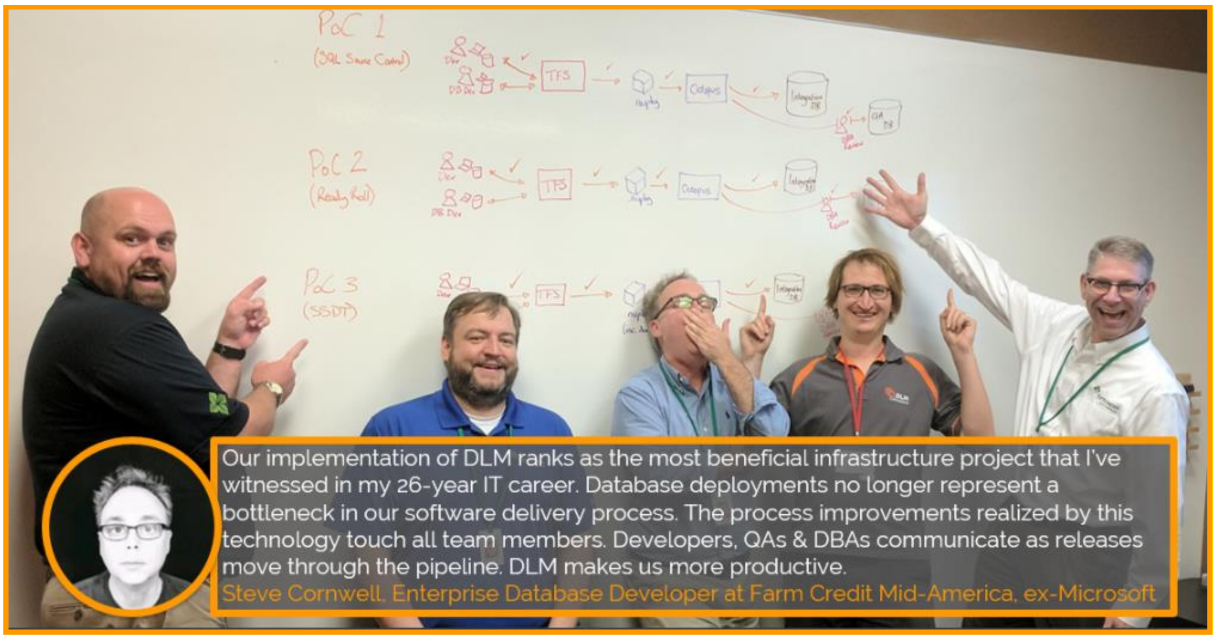
As stated on page one of this whitepaper, the Agile and DevOps movements have articulately explained the relationship between IT excellence and business success, and effective change management and deployment are crucial to achieving it. Hence, it's wise to invest in the optimal tooling for your team.

Set up a proof of concept

It is normally possible to set up a proof of concept (PoC) using both Redgate SQL Source Control and SSDT for no additional licencing costs because SSDT is available with Visual Studio and Redgate offers a free trial.

When DLM Consultants visited Farm Credit Mid-America we set up a cross functional team with a representative from each of the impacted job roles and together we set up three PoC solutions using SSDT, Redgate SQL Source Control and another tool.

After completing this exercise, the team voted unanimously for their preferred solution, but each member had their own reasons for doing so – many of which are reflected in this whitepaper.



Our implementation of DLM ranks as the most beneficial infrastructure project that I've witnessed in my 26-year IT career. Database deployments no longer represent a bottleneck in our software delivery process. The process improvements realized by this technology touch all team members. Developers, QAs & DBAs communicate as releases move through the pipeline. DLM makes us more productive.
Steve Cornwell, Enterprise Database Developer at Farm Credit Mid-America, ex-Microsoft

Figure 12 It took two days for FCMA, with the support of DLM Consultants, to set up three PoC solutions and to decide which to invest in

Ultimately, whether you choose SSDT or SQL Source Control, either product is an excellent investment of your time and/or money and either would be a significant improvement over home-grown solutions or a simple lack of database source control and deployment automation.

Other options

Migrations-based tools

To avoid getting tangled up in higher level debates about whether to adopt a migration-based or a model-based source control solution – or even a hybrid, this whitepaper is specifically scoped to model-based source control solutions. That's not to say model-based solutions are better or worse, but if you are using a model-based approach you are more reliant on tooling. For more information on the relative pros and cons of model-based and migration-based solutions:

workingwithdevs.com/delivering-databases-migrations-vs-state

That said, if you are interested in adopting a migrations-based approach various options exist. With so many third-party and open source database migration tools to choose from I will not try to produce a complete list. However, the following are all popular options that DLM Consultants have had experience with:

- **The Redgate SQL Change Automation Visual Studio Extension** (previously known as ReadyRoll) is a migrations-first hybrid and also requires a SQL Toolbelt licence. While it might be expensive, it offers various premium features that cement its place as the most mature migration-based tool for SQL Server. For example, the auto generation of update scripts saves significant amounts of time and the "Schema Model" and "Programmable Objects" features help to minimise the potential downsides of the migration-based approach. It is also a natural choice both for teams who have previously used SSDT (because it's built on top of SSDT and shares many features) and teams who have previously used Redgate SQL Source Control (because it is from the same family of tools and comes under the same licence):
  red-gate.com/products/sql-development/sql-change-automation

DLM Consultants

- **DbUp** is an open source project by Paul Stovell (who also created Octopus Deploy). DbUp is a C# project that you can add to a Visual Studio solution to run an ordered set of T-SQL scripts in sequence. DbUp is a good choice for folks who like their project to exist in Visual Studio (like SSDT and the Redgate SQL Change Automation Visual Studio extension) but who want to adopt a migration-based approach and don't have the budget for the Redgate SQL Toolbelt. While this tool won't auto-generate your update scripts, detect drift or match the SQL Change Automation hybrid features, it is certainly simple and effective and is a good choice for simple SQL Server projects:
  dbup.github.io
- **Flyway** is another open source option which, like DbUp, will run through an ordered list of .sql scripts and execute them in sequence against a target database. Unlike DbUp its written in Java. This may seem an unnatural choice for a SQL Server source control and deployment tool, and that's because Flyway was not written specifically with SQL Server in mind. Unlike the other tools mentioned in this whitepaper, Flyway supports many relational databases, including MySQL, Oracle and Maria DB. This makes Flyway a popular choice for teams who need to support multiple different databases technologies. Flyway also has a Pro and Enterprise edition with various additional features and support options available:
  flywaydb.org

If you would like to discuss the various migrations-based solutions further, send us an email and we'll be very happy to talk you through the options:

enquiries@dlmconsultants.com

Other model-based tools

While SSDT and SQL Source Control, between them, account for the way the majority of teams source control and deploy SQL Server databases if using model-based approach, others do exist. For example see:

- ApexSQL source control:
  apexsql.com/sql-tools-source-control.aspx
- DBmaestro:
  dbmaestro.com/products/database-release-automation/
- dbForge Source Control:
  devart.com/dbforge/sql/source-control/

However, none are anywhere near as widely used or trusted for SQL Server databases as either SSDT or Redgate SQL Source Control.

More information and training resources

Online training

If you would like DLM Consultants' support with Database DevOps, sign up for one of our live "Database DevOps: What, Why and How" instructor-led classes. Every attendee will be provisioned with their own VM on which they will build their own source control and deployment PoC using Git and Azure DevOps Services and either Redgate SQL Source Control or SSDT. Classes are held online so attendees can tune in from their home or office and save on travel costs.

"Got all I wanted from this training and so much more. Alex is an excellent teacher and explained all the processes very clearly. If you want to know about Database DevOps, Alex is the man to listen to."

★★★★★

Nick Withey

"Alex has helped me wrap my mind around database DevOps some tools and approaches that can help me begin my journey into DLM. Highly recommended course from one of the experts in the field."

★★★★★

Bruce Wilson

All classes are delivered online during either an EU or US friendly hours.

The current class schedule (at time of writing) is as follows:

- **Jan 21 - 22 2019:** SSDT (US hours)
- **Jan 24 - 25 2019:** Redgate (EU hours)
- **Jan 28 - 29 2019:** Redgate (US hours) – ***Hosted by Brent Ozar Unlimited!***
- **Jan 31 - Feb 1 2019:** SSDT (EU hours)

You can see our live schedule and book your place here:

dlmconsultants.com/dlm-workshops

"Great class, fulfilled all my expectations and Alex was a great instructor! I now feel confident with DevOps, but the hard part lies ahead in implementing Continuous Integration and Continous Delivery for my team."

★★★★★

Sebastian Habeker

"Excellent Workshop. Covered all parts of SSDT that I can think of. The exercises helped to develop Database DevOps skills using SSDT. I would definitely recommend."

★★★★★

Shayan Iftekhar

Consulting services

DLM Consultants are based in the UK and offer two off-the-shelf consulting packages: **DLM Health Check** and **DLM Kick-starter**. We are also happy to discuss bespoke options upon request.

We specialise in Database DevOps and Data Privacy and are happy to work on-site or remotely. (Based on current exchange rates our prices are surprisingly competitive in non-UK currencies!) To date, DLM Consultants have visited clients in person on three continents and served customers remotely on five.

"It's been an incredible journey for us at Greentube. It wouldn't have been possible without Alex and Rob. They've been absolutely awesome whenever we required their help!"

Manoj Lona, Database Team Lead at Greentube

"Within a few minutes, it was apparent Alex knew his stuff. Each company he has had to work with is unique. He knew how to apply the Redgate and Octopus tooling to solve our problems. Alex knew to ask the right questions to ensure the solution would work. He could easily have force fed a solution, but he didn't. DLM Consultants' Kick-starter package is based on our engagement. Highly recommended"

Bob Walker, Former Lead Developer at Farm Credit Services of America

Currently Solution Architect at Octopus Deploy

"DLM Consultants helped us take our first steps with Database DevOps. We produced a POC and presented it to the wider team. Even the sceptics could see the value. We had a roadmap and could taste our reward. They also warned that the journey will be hard, which demonstrates DLM Consultants' integrity. We highly recommend DLM Consultants to anyone who wants highly professional database DevOps consulting"

Cihan Ucar, Senior Software Architekt at VAT Vakuumventile AG

"The truth is that after previous attempts to implement DLM we'd begun to believe the possibility was just fantasy. Your style of including the team in the design process enabled us to not only meet our goal but exceed it greatly, and the way you rationally challenged our release timeline led to an eye-opening realization. This is an event that rarely occurs without external influence and I appreciate yours greatly."

Brian Locke, Director of Systems Development at DataScan

"We booked a DLM Kickstarter with DLM Consultants. It was tremendously valuable to have Alex pair with me as I got my head around Bamboo, tSQLt and the various DB automation tools. He was passionate about getting our PoC working and clearly explained the pros/cons of different strategies. Without his support it would have taken us much longer to put together a working solution. I highly recommend DLM Consultants."

Pencho Belneyski, Release Manager at the United Nations Office for Project Services

"Adopting DevOps involves making difficult, hard-to-reverse changes. You'll change dev processes, introduce new tooling, reform database processes, and train your staff on things that may not seem beneficial at first. You don't want to do this twice. You need people who have done this before, listen well, and communicate the right information to your team at the right time. Those people are the DLM Consultants."

Brent Ozar, SQL Server DBA Consultant and Brent Ozar Unlimited

To contact us regarding a consulting engagement, send us an email:

enquiries@dlmconsultants.com

About the author

Microsoft MVP, Alex Yates

Alex Yates is the director of DLM Consultants Ltd, an independent and specialist Database DevOps training and consulting company based in Cambridge, UK. Alex started his IT career at Redgate and spent six years between 2010 and 2016 in various roles, including SQL Source Control Product Champion and Pre-sales Engineer.

In his time at Redgate Alex helped some of Redgate's largest customers to roll out SQL Source Control across their businesses and he also co-authored a three-day database DevOps training class with Grant Fritchey, Steve Jones and a couple of others.

Alex loves solving problems and doesn't like selling, so in 2016 he quit his job at Redgate and founded DLM Consultants so that he could continue to help customers to solve problems but be impartial with regards strategy and tool choice. His integrity and impartiality are important to him:

workingwithdevs.com/introducing-dlm-consultants-mission

Through DLM Consultants, Alex has helped customers to implement database source control and automated deployment solutions using SSDT, SQL Source Control and many other tools, including Redgate SQL Change Automation (previously known as ReadyRoll), DbUp, Flyway and all the major source control, CI and release automation tools including Git, TFS, Azure DevOps Services, TeamCity, Octopus Deploy and many others.

At the time of writing Alex has an existing relationship with both Microsoft and Redgate. He is a "Microsoft MVP" and a "Friend of Redgate". These awards recognise Alex's deep understanding of the Microsoft and Redgate technologies featured in this whitepaper, as well as his community contributions. Also, Redgate are an official partner of DLM Consultants. However, any financial incentives that DLM Consultants receives from any of our partners are donated to charity.

dlmconsultants.com/charity

Alex is a co-organiser of Data Relay, an annual Microsoft Data Platform conference that welcomes 1,000 attendees through the door to see 100 sessions and workshops across 5 cities during a single week in October each year. He is also the founder of speakingmentors.com.

Acknowledgement to peer reviewers

As a defence against the author's unconscious biases, this whitepaper has been peer reviewed by various community members who have experience with either Redgate SQL Source Control, SSDT or both.

With that in mind, the author would like to take this opportunity to sincerely thank each of the following people for their time and feedback. All of which has been read carefully, acknowledged and acted on:

Brent Ozar
SQL Server DBA Consultant at Brent Ozar Unlimited

Erik Ejlskov Jensen
Cloud Data and DevOps Specialist at Cloudeon A/S
Microsoft Data Platform MVP

Kamil Nowinski
Data Engineer at ASOS
Microsoft Data Platform MVP

Drew Furgiuele
Senior Database Administrator at IGS
Microsoft Data Platform MVP

Bob Walker
Solutions Architect at Octopus Deploy

Dustin Shoup
Sr. Database Engineer at Edwards Lifesciences

Denis McDowell
Senior Manager of Application Services at AvidXchange, Inc.