WHITEPAPER

# Developing and deploying databases with SSMS and Visual Studio

A comparison between Microsoft SSDT and Redgate's Database DevOps solution

Compliant Database DevOps

redgate

# Developing and deploying databases with SSMS and Visual Studio

## Contents

# Introduction

The way applications and databases are developed has changed dramatically since the term “agile software development” was coined in 2001. To increase the rate at which organizations deliver value to their customers, more and more developers are expected to develop changes for databases as well as applications.

Organizations can now choose between multiple approaches to manage database changes as part of their overall development process:

## 1. State approach

Developers define the desired end state of a database schema, and do not control how the transition occurs. CREATE scripts defining the state of database objects are checked into a version control system. Code to generate the change is automatically scripted by comparing the desired state with the target database just prior to deployment.

The state approach is often preferred by teams who are experienced with comparison-based database tools, and teams where not everyone has expertise in the T-SQL language.

## 2. Migrations approach

Developers define migration scripts, which control exactly how the change for the database is executed. Migration scripts are typically ALTER statements but may also contain data modification statements. Migrations are applied in sequence against target databases to effect larger, iterative updates to a database.

The migrations approach is often preferred by teams who deploy to high-uptime environments, and by teams which include some expertise in the T-SQL language, as it offers the ability to control the exact code which is executed against the target database.

## 3. Hybrid approach

The state and migrations approaches are combined into a single solution which leverages the strengths of each approach.

Many organizations using Microsoft Visual Studio begin developing database changes with Database Projects in SQL Server Data Tools (SSDT), which offers a state approach.

# What sets Redgate's Database DevOps solution apart from SSDT?

Redgate offers a hybrid approach with SQL Change Automation.

SQL Change Automation gives teams the predictability and reliability of the migrations approach, while using the SQL Compare engine to generate code automatically and provide the productivity benefits of the state approach for team members who are not experts in T-SQL.

Redgate also provides plug-ins for this hybrid approach for both Visual Studio and SQL Server Management Studio (SSMS), while Database Projects in SSDT are only available in Visual Studio.

Additionally, Redgate's Database DevOps solution enables virtualized database copies to be deployed on-demand as part of each developer's workflow. SQL Provision gives developers the power to deploy a writeable, private, production-like environment as needed, while keeping DBAs in control of the data. Spinning up a new database or refreshing an existing environment is nearly instantaneous, even for Very Large Databases (VLDBs), and the copies are a fraction the size of the original – typically around 40MB, even for databases many TBs in size.

Redgate's solution offers six unique advantages:

## Cross-team collaboration for developers and database administrators

Developers on the Microsoft Data Platform often prefer to work in Visual Studio. Many database administrators, however, do not install or learn to use Visual Studio and are much more comfortable working in SQL Server Management Studio. Using Redgate's solution, each team member can collaborate on the same projects in the tool of their choice.

## Automatic provisioning of secure databases in the development workflow

Redgate's solution enables developers to automatically provision or reset a development database with de-identified ("masked") data when creating a branch in source control to develop a new feature. With one click or one line of code, a fresh sandbox environment is deployed and associated with the new branch, enabling the developer to work against a realistic environment while protecting sensitive information by default.

## Automatic code generation with easy customization

Redgate's solution empowers developers skilled in T-SQL to write custom code for their changes if they wish. Developers who prefer a simpler workflow, or who are less skilled with T-SQL, may use the graphical tools in Visual Studio or SQL Server Management Studio to bring their development database to the desired state for their feature. Redgate's solution will automatically generate the code for their change when they are ready to move forward. This code may be used as-is or can be easily customized.

## Provide custom resolutions for code conflicts

While Redgate's hybrid approach empowers developers to generate and customize migration scripts, it also automatically tracks the state of database objects behind the scenes. This enables developers to easily identify and resolve conflicts in objects which have been modified by other users when merging branches or creating pull requests.

While pure state-based approaches also help identify code conflicts, Redgate's Database DevOps solution gives more flexibility for custom resolution of these conflicts, such as the abilities to:

- Use dynamic logic in change scripts to determine if specific code should be used against individual targets, or to control the use of synonyms across various targets
- Use custom filters to prevent deployment of specified objects to some targets

## Early identification of production impact for changes

Redgate's data virtualization technology empowers developers to quickly and easily provision realistic, writeable datasets. When developers use such datasets for feature development, tests and the review of pull requests, problems in both functionality and performance are identified far earlier in the software development lifecycle, saving valuable work hours and preventing customer impact.

## Parallel approach for Oracle

Redgate's Database DevOps solution extends past the bounds of the Microsoft Data Platform and enables teams to develop and deploy changes against Oracle databases in much the same way.

# Overcoming the challenges of the state approach

Organizations that manage database code with the state approach, using a tool like Database Projects in Visual Studio's SSDT, commonly run into problems as teams grow, more frequent deployments are required, or their database code becomes more complex.

Redgate's solution overcomes these challenges in the following ways:

## Prevent deployment errors

At the point changes are deployed to a database, the existing data needs to be preserved. Some modifications, however, cannot be easily handled by the comparison approach because tools that use it don't understand the chronology of the change.

When asked to split or merge a table, for example, SSDT's data-loss safeguard will be triggered and the deployment will terminate with an error message. This is because the comparison method does not provide enough information to determine how the data in the existing database should be treated during the change. In this instance, pre- and post- deployment scripts will need to be created to move the data from the existing table to the new table to prevent data loss. As pre- and post- deployment scripts are run in each deployment, code added to them must be idempotent, which adds complexity to writing and maintaining the scripts.

Similarly, other changes like altering the data type or size of a column, adding a CHECK constraint to a table, adding a NOT NULL column to a large table, updating lots of rows at the same time, creating indexes online, or adding a column in the middle of a table may not be possible or can end up with unintended or confusing results.

Redgate's hybrid solution avoids these issues by generating numerically-ordered migration scripts automatically, using the SQL Compare engine, the industry standard for comparing and deploying SQL Server database schemas quickly and accurately. This makes it very easy to customize the script as required in order to handle delicate migrations so that deployments are predictable and error-free.

These changes are included in the normal development workflow, allowing pre- and post-deployment scripts to be purely dedicated to code which developers wish to be executed on every deployment (and not used for cumbersome workarounds).

## Deploy reliably from version control

The state approach used by SSDT dynamically generates a change script at the time of deployment, at which point errors and delays can occur.

Redgate's solution saves migration scripts for changes that require careful handling, which are checked into the same version control system as the application code during the ongoing development process. After optional review and customization, the migrations functionality then kicks in and the script becomes immutable. This means that, as expected with a migrations-based approach, users can have confidence that the change operation will be performed in exactly the same way across different environments such as QA and Staging. Before changes reach Production, the exact code which will run has been successfully deployed and validated in multiple environments.

Changes to programmable objects such as stored procedures, triggers, views, and functions are also version controlled in Redgate's solution, this time as individual T-SQL files, giving them the same level of visibility and speed of deployment that the state approach provides. This makes it much easier to resolve conflicts when developers are editing the objects concurrently, while the object level history it provides makes it easy to understand how an object has changed over time. As a result, the development team always knows what will be deployed, and can be confident there will be no surprises waiting at the end of the development process.

## Easily include static or reference data

Alongside transactional data, databases invariably contain static data. Common examples of such data include reference lists and codes for states, regions and countries, as well as industry-specific data like car models or supplier names. Static data needs to be reliably stored in version control and deployed to development and production environments.

To manage static data in SSDT, teams must write custom code maintained in pre- and post-deployment scripts which have to be maintained outside the main development pipeline. As these scripts are also used for other workarounds for challenges in the state approach, they quickly become complex, and they are a common headache for SSDT users.

As well as versioning and deploying schema changes, SQL Change Automation allows users to easily add static or reference data tables to the version control repository from the plug-ins in Visual Studio or SQL Server Management Studio. Redgate's solution understands that the data is now source controlled in these objects and will detect changes to the tables made in a development environment and suggest the changes be imported and committed to version control. Once committed, static data changes will be deployed as part of the same script as schema changes. A frequently seen problem thus becomes part of the normal workflow.

## Catch errors earlier in development

While SSDT leaves it to the point of deployment to generate the migration script, SQL Change Automation moves the review of database changes to the earliest stage of development. As soon as changes are imported into a project using Visual Studio or SQL Server Management Studio, a migration script is generated which can be checked and corrected if necessary.

Importantly, the scripts can be immediately verified with SQL Change Automation's shadow database capability, which creates a disposable copy of the database to check the scripts migrate it to the required state. Redgate's solution does not merely validate syntax logically: verification executes the exact code from the migration scripts and gives immediate feedback to the user if a syntax error is in place.

This verification functionality empowers developers to customize code and validate it, all before they commit the code to version control. Developers who choose to may work iteratively and commit changes to their code in a feature branch, iteratively making changes and validating them. This workflow gives them the freedom to experiment, easily undoing any commits which they no longer need, and verifying code on-demand.

In this way, bugs are caught early when they are still easy to fix, the fixes can be applied and tested again while still in development, and the likelihood of experiencing major issues at deployment, or downtime afterward, is greatly reduced.

## Conquer circular cross-database dependencies

While developers never set out to intentionally create circular dependencies between different databases, it's common to find this type of dependency in real-world environments.

For example, two databases may each have a view which depends on objects in the other database. SSDT does not support this type of dependency and removing them altogether often requires significant refactoring of applications, which consumes significant developer time and doesn't result in new functionality for users. Working around the problem in SSDT results in a confusing project configuration which may lead to developer error.

Redgate's solution eliminates this problem simply by leveraging data virtualization: you may quickly provision databases for development, build, and test environments, satisfying any dependency on demand.

## Simplify and source control object filters

It's common for non-production databases to contain objects which you never wish to deploy to production. While SSDT provides some functionality to filter items, it's tricky to manage these filters and ensure they are used consistently throughout your pipelines.

Redgate's solution makes it easy to generate filters using SQL Compare technology, associate the filters with your project, and commit the filter into version control. This filter will now automatically be used for the project by all developers and all deployments with no extra knowledge needed.

If there are times when you wish to override this filter for selected deployment targets in your pipelines, a filter override can be specified easily using Redgate's graphic plug-ins or PowerShell cmdlets.

## Provide true visibility across deployments

Redgate's Database DevOps solution for the Microsoft Data Platform was designed from the ground up to facilitate a continuous integration and continuous delivery approach to database development. Tightly integrated with build and release tools including Azure Pipelines in Azure DevOps Services (formerly Visual Studio Team Services) and Azure DevOps Server (formerly Team Foundation Server), builds can be triggered whenever changes are committed, or at critical points defined in a pipeline.

One popular pattern with Redgate's solution uses the following configuration:

- The master branch has a policy requiring that changes be made via pull request functionality (may not be directly committed to master)
- This branch policy specifies that pull requests will automatically trigger a build
- Following a successful build, a pipeline is run which provisions an environment, then generates a release artefact containing a code summary and static code analysis against the environment
- Finally, the code in the pull request is deployed to the environment

Each piece of functionality in Redgate's solution may be deployed using a graphical plug-in for popular orchestration tools, or a simple PowerShell script.

With Redgate's solution, database administrators or other change reviewers are provided with a summary of the code which has already been deployed to an environment which was created and updated using automation.

None of these artefacts or environments are generated automatically by SSDT, and the only way to review the deployment script is to generate it in its entirety for each and every environment deployment, even if a single incremental change was made on top of previously reviewed changes.

By showing code reviewers exactly what is being deployed, and by reducing the amount of time they spend on manual tasks during the deployment process, Redgate's solution gives complete visibility, encourages collaboration with development teams, and improves efficiency.

## Why should organizations use Redgate's Database DevOps solution?

Redgate's solution offers a hybrid approach to database development that combines the strength and reliability of the migrations approach, while also taking advantage of the benefits of the state approach.

This solution brings your team together across long-standing organizational silos: developers who prefer working in Visual Studio can collaborate on the same project with database administrators who prefer working in SQL Server Management Studio.

Finally, Redgate brings data into the software development lifecycle by integrating data virtualization into existing development practices. This empowers developers to effortlessly provision production-like databases as part of their regular workflow. Leveraging automatic code creation and the ability to easily customize code as needed, teams can automatically validate and test their code, identifying problems early in the software development lifecycle and preventing wasted time and customer impact.

Quite simply, Redgate's solution enables your team to deliver value to your customers more quickly.



Organizations wanting to explore the capabilities described in this paper can download 14-day fully functional trials of **SQL Toolbelt** and **SQL Provision**.

For further information and resources, visit:

**www.redgate.com/solutions**