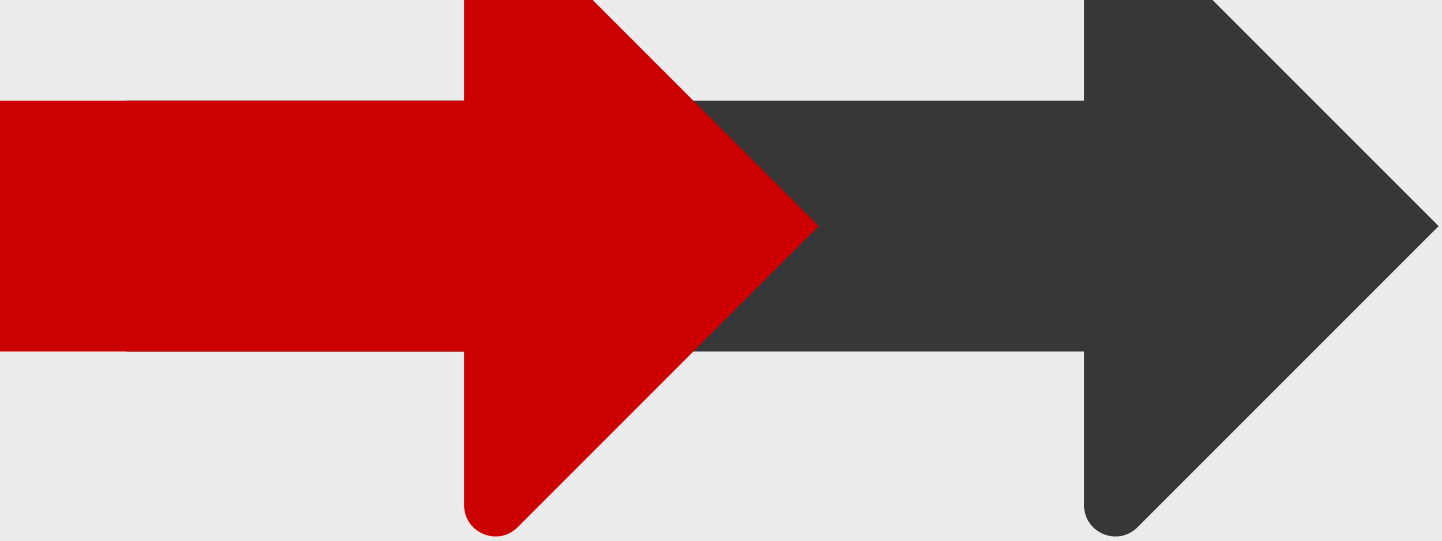


State-based or migrations-based database development

Compliant
Database DevOps

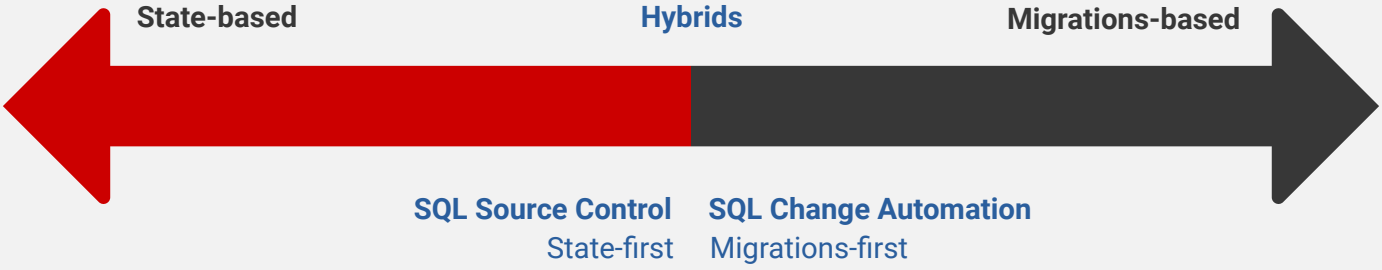
redgate

State-based or migrations-based database development

While those of us in the SQL Server world have benefited from tools which help us design the end-state of the database and work out how to make the changes correctly, script-based processes have moved into the background. However, Redgate SQL Change Automation now gives new life to script-based deployment processes.

This paper outlines the differences between what's referred to as the model or state-based approach and the migrations-based approach, and the corresponding Redgate tools. Both approaches, at their root, are about which is your source of truth: the definition of how you want the database to look, or the scripts you produce to handle the upgrade.

State-based or migrations-based tools typically sit at either end of a spectrum and are exclusive in their approach. Redgate's tools primarily follow one approach or the other, but both SQL Source Control and SQL Change Automation benefit from hybrid features, explained below, designed to address the shortcomings associated with the extreme ends of the spectrum:



SQL Source Control is primarily state-based, but pre- and post- deployment scripts enable flexibility for deployments where custom coding is needed to effect the desired change.

SQL Change Automation is primarily migrations-based, but features state-based programmable objects and an offline schema model to reduce the number of migration scripts needed, enable users to compare source code with databases, and support identifying merge conflicts.

	SQL Source Control the state-first approach	SQL Change Automation the migrations-first approach
Definition	Aims to enable a single step between the current database state and the desired database state. The current state of each object in the database is versioned as a CREATE script. **SQL Source Control** compares the source to the target and auto-generates a script to synchronize the two states using the SQL Compare engine.	Captures individual change scripts during development to effect larger, iterative database migrations. **SQL Change Automation** auto-generates numerically ordered migration scripts using the SQL Compare engine. Changes (often ALTER commands) are organized in SQL scripts and run in order to migrate a database from one version to the next.
Development model	Both **SQL Source Control** and **SQL Change Automation** support the preferred connected development model, allowing users to modify the database directly in SQL Server Management Studio (SSMS).	
IDE	**SQL Source Control** has an extension for SSMS, allowing you to commit the state-first representation of your database to the version control system (VCS) of your choice.	**SQL Change Automation** has extensions for both Microsoft Visual Studio (VS) and SSMS. You may use either IDE to commit the migrations-first representation of your database to the VCS of your choice. Team members may collaborate on the same project across both IDEs.
Scenario	A simple approach to get started with standardizing development practices, especially for teams who are experienced with SQL Compare and other state-based development tools.	Ideal for complex database changes and environments with high uptime requirements, as there is no ambiguity in the code which will be executed in a deployment.
Deployment	Deployment scripts are inferred at deployment time using the SQL Compare engine.	Deployment scripts are built from the building blocks of individual migration scripts.
	Assisted deployments can be carried out via the SQL Compare UI, with full transparency to review changes prior to deployment.	Manual deployments can be executed via a package sql script.
Automation	Both **SQL Source Control** and **SQL Change Automation** may use SQL Change Automation's PowerShell cmdlets and add-ons for automation to implement continuous integration and automated deployments for your database changes.	
Drift	Using SQL Source Control or SQL Change Automation, developers benefit from drift detection functionality.	
Hybrid features	Some changes such as column splits or similar data motion scenarios can't be easily managed via a pure state-based approach because the tools don't understand the context of the change. **SQL Source Control** offers pre- and post- deployment scripts that enable users to hard code certain parts of the deployment process.	Migrations-based approaches can introduce a "last one in wins" scenario, in which the last check-in could overwrite previous migrations. To avoid this, users can take advantage of **SQL Change Automation's** Programmable Objects feature for a state-based approach for managing functions and stored procedures. An offline schema model enables you to identify merge conflicts and perform comparisons between source code and databases.

www.red-gate.com/sql-source-control

www.red-gate.com/sql-change-automation