Terminology Reference

SQL Change Automation terminology for developing database code with a migrations-first approach

SQL Change Automation (SCA) Visual Studio project

A *SQL Change Automation project* contains SQL scripts and configuration files. It contains both the state of a database, represented in an offline schema model and programmable objects, and a set of migration scripts that describe how to get to that state. The set of migration scripts may contain baseline scripts.

Development database

A *development database* is linked to the SQL Change Automation project for the purpose of generating migration scripts and updating programmable objects and the offline schema model. In the initial setup wizard for SQL Change Automation, this database is selected as the development database. You may wish to occasionally change the connection to your development database. For example you might want to use a newly provisioned database with new data in it for development. The development database may be changed for the project in Visual Studio by clicking 'Configure database connection' in the SQL Change Automation window. Read more about development and shadow databases.

Shadow database

The *shadow database* is controlled by the SQL Change Automation Visual Studio Extension. SQL Change Automation keeps the shadow database consistent with all the migration scripts currently in the project as needed, and uses it to verify scripts to detect problems in your code. Read more about development and shadow databases.

Target database

A *target database* is a database in a continuous deployment pipeline which is not a development database. One example of a target database is the production database. In the initial setup wizard for SQL Change Automation, an existing production database may be specified as the development target, for the purpose of creating an initial baseline. Automation components, such as release artifacts, may be deployed to target databases later with the automation components in SQL Change Automation, detailed below.

Baseline schema

A *baseline schema* captures changes that have already been deployed to your production database. We recommend creating an initial baseline schema for databases with an existing production environment when you first create your SQL Change Automation project. Read more about baselines.

Online edits

Online edits are made by running T-SQL commands or using a graphic interface to execute commands directly against the development database. This can be done using Visual Studio or other client tools such as SQL Server Management Studio or Azure Data Studio. These changes are then imported into the SQL Change Automation project.

For an example of how to make an online edit style change in Visual Studio, see the 'Create a view using the online-editing method' section in 'Getting started with SQL Change Automation projects in Visual Studio'.

Importing online changes

After a change is made against a development database online, a developer or DBA refreshes the list of database objects with pending changes in the SQL Change Automation project. The developer or DBA may then view object differences for those changes, which allows them to compare the state of the object in the project (if present) to the modified state of the development database. The developer or DBA then imports SQL scripts for the pending changes. The import may result in new migration scripts as well as changes to programmable objects and the offline schema model, depending on the nature of the change.

Refresh (Verify Script)

After creating an initial baseline or importing online changes made directly against the developer database, the 'Refresh (Verify Script)' button appears in SQL Change Automation in Visual Studio. Clicking this button runs this script against your shadow database. This script verification process identifies problems before the script is deployed elsewhere.

Offline edits

Offline edits are made by changing files in the Visual Studio project. These changes are then applied to the development database.

If you have the programmable objects feature enabled, offline additions or modifications to DDL Triggers, user defined functions, stored procedures, and views should be made directly to the scripts in the 'Programmable Objects' folder in the Visual Studio Solution Explorer pane.

Offline edits for other types of objects, such as tables, may be made by manually writing migration scripts or generating migration scripts from the SQL Server Object Explorer pane in Visual Studio. For an example of how to make this type of change in Visual Studio, see the 'Design a table and generate your first migration script' section in 'Getting started with SQL Change Automation projects in Visual Studio'.

Note: Both online and offline style changes may be made at any time within a single SQL Change Automation project, based on the preferences of the user for that change. If a user is most comfortable with SQL Server Management Studio, for example, they may use that for most of their work. That user would use Visual Studio only to import, verify, and commit their changes. Similarly, users may work exclusively in Visual Studio using a combination of online and offline edits, if desired.

Applying offline changes

When a change is made against a development database offline, such as making a change to a table with SQL Server Data Tools inside Visual Studio, the developer or DBA generates a script in Visual Studio. This script is added to the SQL Change Automation project as a migration script. The migration script may be modified as needed. Once the developer or DBA is ready to *apply* changes to the database, they click 'Deploy Project'. This deploys the change to the development database. Following this, refreshing the list of database objects pending import will update the offline schema model.

Note that refreshing the list of database objects will *not* update programmable objects. When programmable objects are enabled, offline modifications should be made directly to the programmable object files in the Solution Explorer pane. Alternately, an online edit approach may be used for programmable objects (see 'Online edits' above).

Offline Schema Model

SQL Change Automation will maintain a set of CREATE scripts written in SQL that define the desired state of a database's schema when the *offline schema model* is enabled. INSERT scripts are maintained for static data. The offline schema model helps you quickly review the state of your database as well as view the history of objects over time.

Migration script

A *migration script* moves a database schema from one version to another. It is only executed when deploying to a target database if it has not already been run on that database. See migrations and migration grouping.

Pre- and Post- Deployment scripts, also called Pre/Post Scripts

Pre- and post- deployment scripts are run before (Pre-Deployment script) or after (Post-Deployment script) the main block of scripts to be run against a target database. Unlike a migration script, a pre- or post-deployment script is run every time a project is deployed. These can be used for creating the database if it doesn't exist, altering database settings, or for static data. See transaction handling for extra considerations for these scripts.

Terminology for SQL Change Automation's automation-specific components

These components in SQL Change Automation may be used to automate deployment for SQL Change Automation projects. They may also be used to deploy SQL Source Control projects, for situations where a state-first approach to developing code is preferable.

In this reference, we link directly to the documentation on SQL Change Automation's PowerShell cmdlets. You may use these cmdlets directly to automate your database changes, or use SQL Change Automation's add-ons for Continuous Integration and Continuous Deployment Servers.

Build

The primary task of a *build* is to validate the contents of a SQL Change Automation project by checking that the scripts and the configuration files in the project can be used to build a database from scratch. This validates that the SQL is valid, that all dependencies are correct, etc. A build is done inside SQL Change Automation when validating code against the shadow database. SQL Change Automation also has components to allow you to invoke a full build against an empty database in a location of your choice, and you may produce a build artifact. The Invoke-DatabaseBuild PowerShell cmdlet initiates a build.

Build artifact

A *build artifact* is the output of a build, which can then be used to create a release artifact. Build artifacts may be exported and stored in a file system as either NuGet packages or zip files. Optionally, database documentation may be generated in an automated fashion and included in the build artifact by SQL Change Automation if the New-DatabaseDocumentation PowerShell cmdlet is integrated into the build process. The New-DatabaseBuildArtifact Powe rShell cmdlet creates a build artifact, which may then be exported by the Export-DatabaseBuildArtifact PowerShell cmdlet.

Release artifact

A *release artifact* is the result of customizing a build artifact for a specific target database. A release artifact can be deployed to a target database, and also contains reports of the changes that deploying the release will make to the target database and static code analysis of those changes. A release artifact is intended for human consumption as well as programmatic consumption. The New-DatabaseReleaseArtifact PowerShell cmdlet creates a database deployment resource specific to the target database specified, and it may be exported for review with the Export-DatabaseReleaseArtifact cmdlet.

Release

To *release* is to actually deploy a release artifact to a target database. The Use-DatabaseReleaseArtifact PowerShell cmdlet applies a release artifact to the target database.