Documentation

Code Analysis

Code Analysis > Code Analysis for SQL Server > Style rules

Style rules

Page last updated 25 July 2018

Code style issues.

ST001 – Old-style join is used (...from table1,table2...)

Use ANSI-style joins via JOIN.
This form of SQL is no longer used. It is always better to specify the type of join you are using, INNER, LEFT OUTER, LEFT OUTER, FULL OUTER and CROSS. If you are only using INNER JOIN then that is a far bigger problem.

Learn more about this rule

ST002 – Old-style column alias via EQUAL sign

It is recommended to specify alias via AS.
SQL uses the AS keyword to specify an alias. The assignment syntax is deprecated.

Learn more about this rule

ST003 – Procedure body not enclosed with BEGIN...END

It is recommended to enclose a routine body in a BEGIN...END block.
The BEGIN ...END block declaration is optional for stored procedure

though it is mandatory for multi-line user-defined functions. To avoid confusion it is best to be consistent.

Learn more about this rule

ST004 – SQL-92 style cursor declaration is used

Use extended syntax instead.

You use the DECLARE CURSOR command to define the attributes of a T-SQL cursor. DECLARE CURSOR can accept the SQL-92 standard syntax and the SQL Server Extended syntax. Only the extended syntax supports the many T-SQL extensions to cursors.

Learn more about this rule

ST005 – IF or ELSE without BEGIN...END block

Note: By default, this rule is set to "Ignore". To enable it, set it to "Warning" in the Rules dialog.

It is recommended to use BEGIN...END as a wrapper for a block of code in IF or ELSE statements.
Although, in theory you only need a block wrapper if more than one statement is used in an IF ELSE sequence It is easy to get statements wrong if you leave out the BEGIN…END

Learn more about this rule

ST006 – Old-style TOP clause is used

It is recommended to use the new style TOP clause - TOP(n).

In order to limit the results of a query, you can specify the number you need by using the TOP clause. The TOP syntax is clearer with the parentheses, though they are syntactic sugar since only one parameter can be within parentheses.

Learn more about this rule

ST007 – Cursor name is reused

More than one cursor has the same name.

Cursors are often thought to have the same scope as variables but this is not the case. It is a mistake to reuse the cursor name.

Learn more about this rule

ST008 – Non-named parameter style used

Use named parameters when calling procedure (exec dbo.Procedure @Parameter1=value,@parameter2=...). Issue registered once per procedure call.

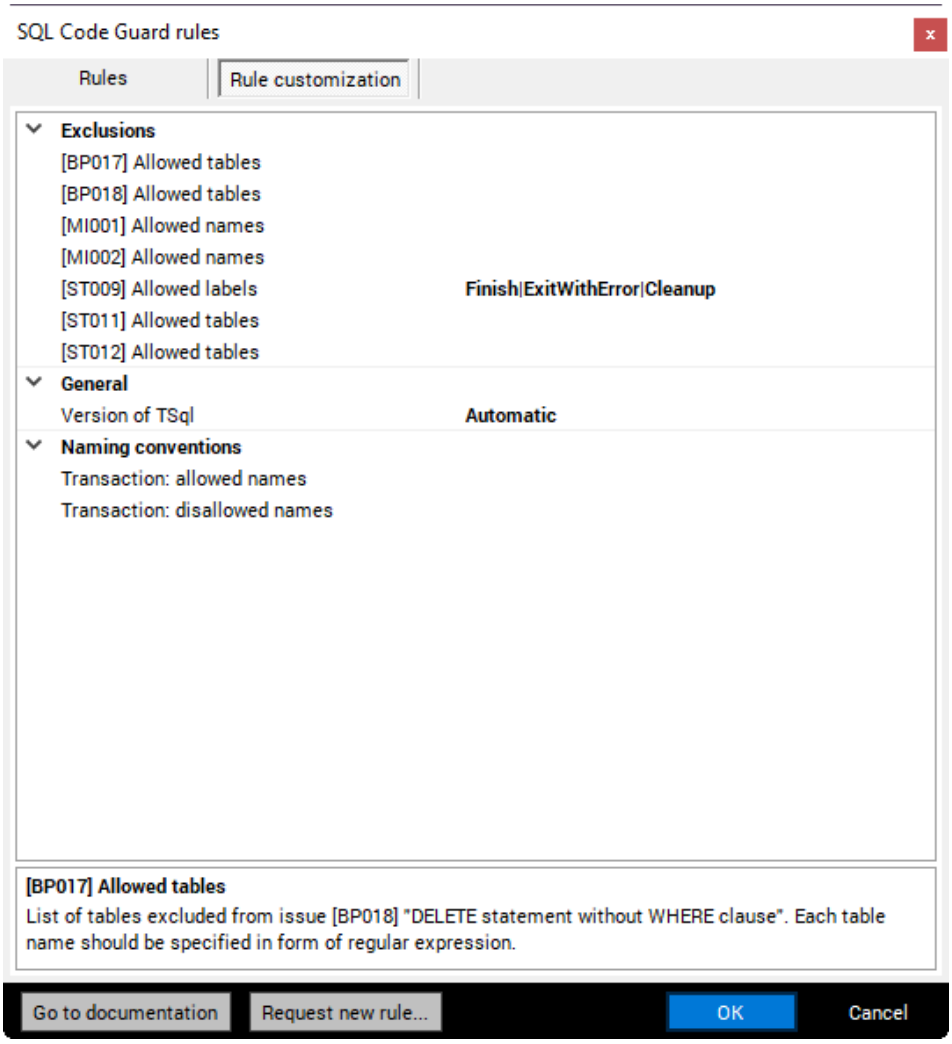
It is better to provide parameters by name, particularly during active development as otherwise, as procedures are improved, parameters have to be added at the end of the list rather than in logical order, and parameters cannot easily be deleted without causing several mysterious problems.

Learn more about this rule

ST009 – Avoid using GOTO to improve readability

GOTO is perfectly legal but it becomes extraordinarily difficult to maintain code that uses GOTOs rather than conventional block structures. It is very rare to find any circumstances where a GOTO is necessary.

GOTO checks for specific labels can be ignored. You may, for example, have a corporate convention to use labels with standard names such as 'Finish', 'ExitWithError', 'Cleanup', and so on. In this instance, you can add the labels to ***ST009 Allowed Labels*** *in the* ***Rules / Customizable Rules*** *dialog using standard regular expression syntax.*



Learn more about this rule

ST010 – Use alias for all table sources

Aliases are designed to improve readability, not to save typing.
You only need aliases where you are joining tables that, together with their schemas, have long names.
For consistency, it is often best to use aliases for every SQL expression that involves more than one table source.
There is no performance advantage to using aliases.

Learn more about this rule

ST011 – Consider using table variable instead of temporary table

These are much easier to work with, are pretty secure, and they trigger fewer recompiles in the routines where they're used than if you were to use temporary tables.

Learn more about this rule

ST012 – Consider using temporary table instead of table variable

If you are doing more complex processing on temporary data or likely to use more than reasonably small amounts of data in them, then local temporary tables are likely to be a better choice than a table variable.

Learn more about this rule

ST013 – Non-ANSI NOT_EQUAL operator used (!=)

Use ANSI-style NOT_EQUAL operator (<>).

The '!=' symbol for not equal is not part of the SQL language. It is understood but only for the sake of backward-compatibility.

Learn more about this rule

ST014 - Procedure name - pattern is not found in allowed patterns

Learn more about this rule

ST015 - Procedure name - pattern is found in disallowed patterns

Learn more about this rule

ST016 - Function name started with fn_

Learn more about this rule

ST017 - Using numbers in table names

Learn more about this rule

Didn't find what you were looking for?

Visit the Redgate forum | Contact Support

Product Articles

Tips and how-to guides for Redgate products

University

Easy to follow video courses

Community Forums

Ask, discuss, and solve questions about Redgate's tools

Events & Friends

Meet us at an event, get sponsored, and join our Friends of Redgate

Simple Talk

In-depth articles and opinion from Redgate's technical journal