0.. SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

* [**NOT NULL**](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [**UNIQUE**](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [**PRIMARY KEY**](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [**FOREIGN KEY**](https://www.w3schools.com/sql/sql_foreignkey.asp) - Uniquely identifies a row/record in another table
* [**CHECK**](https://www.w3schools.com/sql/sql_check.asp) - Ensures that all values in a column satisfies a specific condition
* [**DEFAULT**](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column when no value is specified
* [**INDEX**](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly

1. What is the difference between Clustered and Non-Clustered Indexes in SQL Server?

<https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/>

Very good

<https://stackoverflow.com/questions/1251636/what-do-clustered-and-non-clustered-index-actually-mean>

2. SQL Injection

[❮ Previous](https://www.w3schools.com/sql/sql_view.asp)[Next ❯](https://www.w3schools.com/sql/sql_hosting.asp)

<https://www.w3schools.com/sql/sql_injection.asp>

SQL Injection

SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL in Web Pages

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will **unknowingly** run on your database.

Look at the following example which creates a SELECT statement by adding a variable (txtUserId) to a select string. The variable is fetched from user input (getRequestString):

Example

txtUserId = getRequestString("UserId");  
txtSQL = "SELECT \* FROM Users WHERE UserId = " + txtUserId;

The rest of this chapter describes the potential dangers of using user input in SQL statements.

SQL Injection Based on 1=1 is Always True

Look at the example above again. The original purpose of the code was to create an SQL statement to select a user, with a given user id.

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId:

Then, the SQL statement will look like this:

SELECT \* FROM Users WHERE UserId = 105 OR 1=1;

**How Can You Write Secure Code?**

The solution is often DBMS specific; that is, it varies between MySQL, Oracle or SQL Server. In PHP with MySQL, for example, it is usual to escape parameters using the function mysql\_real\_escape\_string before sending the SQL query. Alternatively, you can utilize [prepared statements](http://net.tutsplus.com/articles/news/crud-with-php-prepared-statements-new-plus-tutorial/) to "prepare" your queries. Make it your mission to understand the DBMS with which you are working and the inherent security issues.

SQL injection isn't the only security vulnerability for databases and developers to worry about, however, it is one of the most common methods of attack. It is important to test your code and be familiar with the latest security issues for your DBMS in order to protect against attacks.

I

———

4. 10 Essential SQL Tips for Developers

**4. Know Your Data Types**

In SQL, typically each table column has an associated data type. Text, Integer, VarChar, Date, and more, are typically available types for developers to choose from.

* [MySQL Data Types](http://dev.mysql.com/doc/refman/5.0/en/data-types.html)
* [Oracle Data Types](http://download.oracle.com/docs/cd/B10501_01/server.920/a96524/c13datyp.htm)
* [SQL Server Data Types](http://msdn.microsoft.com/en-us/library/ms187752.aspx)

When developing, make sure you choose the proper data type for the column. Dates should be DATE variables, numbers should be a numeric type, etc. This becomes especially important when we deal with a later topic: indexing; but I'll demonstrate an example of poor knowledge of data types below:

|  |  |
| --- | --- |
| 1  2  3 | SELECT employeeID, employeeName  FROM employee  WHERE employeeID = 112457891; |

Looks fine based on what we currently know, correct? However, what if employeeID is actually a string. Now we've got a problem, because the DBMS might not find a match (because string data types and integers are different types).

Therefore, if you're using indexing, you'll probably be perplexed as to why your query is taking forever, when it should be a simple index scan. This is the reason that developers need to pay special attention to data types and their applications. Non-key attributes which are IDs are often string types, as opposed to integers, because of the increased flexibility that is granted. However, this is also a trouble area for junior developers, who assume that ID fields will be integers.

*Properly utilizing data types is essential to proper database programming, as they directly lead to query efficiency. Efficient queries are essential to creating quality, scalable applications.*

**5. Write Compliant Code**

All programming languages have standards which web developers should be aware, and SQL isn't any different. SQL was standardized by ANSI and then ISO, with new revisions to the language being occasionally submitted. The latest revision is SQL:2008, although the most important revision that developers should be aware of is SQL:1999. The 1999 revision introduced recursive queries, triggers, support for PL/SQL and T-SQL, and a few newer features. It also defined that the JOIN statements be done in the FROM clause, as opposed to the WHERE clause.

When writing code, it is important to keep in mind why standards-compliant code is useful. There are two primary reasons why standards are used. The first is maintainability, and the second is cross-platform standardization. As with desktop applications, it is assumed that websites will have long lifespans, and will go through various updates to add new functionality and repair problems. As any systems analyst will tell you, systems spend a majority of their lifespan in the maintenance phase. When a different programmer accesses your code in 2, 5 or 10 years, will they still be able to understand what your code is doing? Standards and comments are designed to promote maintainability.

The other reason is cross-platform functionality. With CSS, there is currently an ongoing standards battle between Firefox, Internet Explorer, Chrome, and other browsers about the interpretation of code. The reason for the SQL standards is to prevent a similar situation between Oracle, Microsoft and other SQL variants such as MySQL.

———

5. Explanation of LIKE '%[0-9]%'? 5/11/17

Hi there. First things first: please define "numeric" and what you are REALLY checking for. Are you looking for:

Hi there. First things first: please define "numeric" and what you are REALLY checking for. Are you looking for:

 all digits

 is a valid number anywhere in the world

 is convertible to one of the available number datatypes in SQL Server  
  
These are all different concepts.

 12345 is all digits, a valid number, and convertible to most SQL Server number types.

 12345.00 is NOT all digits, but is still a valid number, and convertible to DECIMAL / FLOAT / REAL.

 12,345 is NOT all digits, but is still a valid number, and convertible to MONEY.

 123.45E+03 is NOT all digits, but is still a valid number, and convertible to FLOAT / REAL.

 12.345.678,9 is NOT all digits, is NOT convertible, but is still a valid number in some locales

 [12 345 678,9](tel:12%20345%20678,9) is NOT all digits, is NOT convertible, but is still a valid number in some locales (such as fr-FR)

 23847234872893475983479583749583749573945739 is all digits, is a valid number, but is NOT convertible to any SQL Server number types as it is larger than 38 digits  
  
So first you need to be clear on what you will accept as being a number and what is not valid. Then you have several options, namely:

 LIKE operator as you have used before: LIKE '%[^0-9]%'. This will find rows that are NOT all digits, but won't catch NULL or empty, which may or may not be acceptable so you might need additional WHERE clauses.

 If you are running SQL Server 2012 (or newer at some point) use the new [TRY\_PARSE()](http://msdn.microsoft.com/en-us/library/hh213126.aspx) function

 String\_IsNumeric function (free in [SQL# (SQLsharp)](http://www.sqlsharp.com/))

 RegEx functions (free in [SQL# (SQLsharp)](http://www.sqlsharp.com/))  
  
Take care,  
Solomon...    
  
SQL# - <https://SQLsharp.com/>  
Sql Quantum Lift - <http://SqlQuantumLift.com/>

# 7. Idera SQL Elements Review: A SQL Server Inventory Management Solution

<https://www.mytechmantra.com/Product-Reviews/Idera-SQL-Elements-Product-Review/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+SQLServerArticles+%28Latest+SQL+Server+Articles%29>

Jan 27 2014  
  
*Page 1/3*

## Introduction

One of the challenges faced while working as a Database Administrator is the management of the SQL Server environment effectively. The SQL Server Environment in my recent workplace was quite unknown because of no DBAs being employed earlier. There was no clarity on the number of servers that were very busy, which ones were backed, current configuration settings used by each instance, status of database maintenance, the ones that are down or slow to respond etc. Attempts were made at developing scripts to determine the usage of servers in the environment but it was proving to be a tedious task. In this process, the issues were being addressed with delay and a lot of confusion had crept into the environment. There was a need for a quick and a comprehensive SQL Server Inventory Management solution to help understand the environment and to get a broad view of all the SQL Servers. While searching for a commercial and broad solution for this issue, Idera SQL Elements turned out to be a viable solution.

## Why to Use Idera SQL Elements

SQL elements helps in the efficient discovery and exploration of one’s SQL Server Environment. The difficulty that an Administrator or an IT Team might face while tracking the large number of servers in an organization gets simplified. For instance, one could be working on a large environment and would need some help on classification of instances. SQL Elements is a perfect answer for this. Secondly, SQL Elements must be used because it helps an administrator to know the basics and hence it is an excellent tool for someone new to the environment. Thirdly, there are built-in default options to manage the servers and this can prevent the hassle of developing scripts as mentioned in the problem above. Fourthly, this tool does the health check and provides recommendations. Some of the recommendations are Volumes which are at the risk of filling up, TempDB configuration issues, Databases which are not backed up, Databases have not been checked for DBCC CHECKDB in the recent times, optimization for ad-hoc workloads etc.

Furthermore, SQL Elements must be used because it provides alerts about the various issues to the DBA and helps in fast identification and resolution of the issues. Finally, SQL Elements can help improve the efficiency and productivity of the administrator as well as the overall environment. This is precisely why the product deserves a review.

"This article has a total of 3 pages including this page. Click the links below to open all 3 pages in seperate windows to learn more about ***Idera SQL Elements.***[Page 1](http://www.mytechmantra.com/Product-Reviews/Idera-SQL-Elements-Product-Review/), [Page 2](http://www.mytechmantra.com/Product-Reviews/Idera-SQL-Elements-Product-Review/P2/)and [Page 3](http://www.mytechmantra.com/Product-Reviews/Idera-SQL-Elements-Product-Review/P3/)"

## Installation

Installing Idera SQL Elements is very easy. The installation wizard is self explanatory and the installation completes in less than 10 minutes.

8. Topics on Index

## Different Ways to Enable an Index in SQL Server

<https://www.mytechmantra.com/LearnSQLServer/Different-Ways-to-Enable-an-Index-in-SQL-Server/>

* Enable an Index Using ALTER INDEX...REBUILD Statement
* Enable an Index Using CREATE INDEX WITH DROP\_EXISTING Statement
* Enable an Index in SQL Server Using SQL Server Management Studio (SSMS)

# Using Check Constraints to Validate Data in SQL Server

<https://www.databasejournal.com/features/mssql/article.php/3811831/Using-Check-Constraints-to-Validate-Data-in-SQL-Server.htm>

# 9. Microsoft SQL Server Constraints – with good example

<http://odetocode.com/Articles/79.aspx>

# **10. Constraint Yourself!**

<https://www.red-gate.com/simple-talk/sql/t-sql-programming/constraint-yourself/>

11. CHECK Constraints

<https://technet.microsoft.com/en-us/library/ms188258(v=sql.105).aspx>

## **Limitations of CHECK Constraints**

CHECK constraints reject values that evaluate to FALSE. Because null values evaluate to UNKNOWN, their presence in expressions may override a constraint. For example, suppose you place a constraint on an **int** column **MyColumn** specifying that **MyColumn** can contain only the value 10 (**MyColumn** **=** **10**). If you insert the value NULL into **MyColumn**, the Database Engine inserts NULL and does not return an error.

A CHECK constraint returns TRUE when the condition it is checking is not FALSE for any row in the table. A CHECK constraint works at the row level. If a table that has just been created does not have any rows, any CHECK constraint on this table is considered valid. This situation can produce unexpected results, as in the following example.

The CHECK constraint being added specifies that there must be at least one row in table CheckTbl. However, because there are no rows in the table against which to check the condition of this constraint, the ALTER TABLE statement succeeds.

CHECK constraints are not validated during DELETE statements. Therefore, executing DELETE statements on tables with certain types of check constraints may produce unexpected results. For example, consider the following statements executed on table CheckTbl.

The DELETE statement succeeds, even though the CHECK constraint specifies that table CheckTbl must have at least 1 row.

# 12. Using Check Constraints to Validate Data in SQL Server – good

<https://www.databasejournal.com/features/mssql/article.php/3811831/Using-Check-Constraints-to-Validate-Data-in-SQL-Server.htm>

# 13. Move a Database Using Detach and Attach (Transact-SQL)

<https://technet.microsoft.com/en-us/library/ms187858(v=sql.110).aspx>

14. sql server import sqlite database – 12/11/2017

Creating a SQL Server Linked Server to SQLite to Import Data

<https://www.mssqltips.com/sqlservertip/3087/creating-a-sql-server-linked-server-to-sqlite-to-import-data/>

# Different Options for Importing Data into SQL Server -- good

<https://www.mssqltips.com/sqlservertip/1207/different-options-for-importing-data-into-sql-server/>

# Dynamically Import data from a foreign database using a SQL Server Linked Server

<https://www.mssqltips.com/sql-server-tip-category/56/linked-servers/> ---this website leads to below website

<https://www.mssqltips.com/sqlservertip/1896/dynamically-import-data-from-a-foreign-database-using-a-sql-server-linked-server/>

<https://www.mssqltips.com/sqlservertip/1896/dynamically-import-data-from-a-foreign-database-using-a-sql-server-linked-server/> ---this website leads to below website 🡪 MS Docus

# **Linked Servers (Database Engine)**

<https://docs.microsoft.com/en-us/sql/relational-databases/linked-servers/linked-servers-database-engine>

# **Create Linked Servers (SQL Server Database Engine)**

<https://docs.microsoft.com/en-us/sql/relational-databases/linked-servers/create-linked-servers-sql-server-database-engine>

# **sp\_addlinkedserver (Transact-SQL)**

<https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-addlinkedserver-transact-sql>

# Introduction to SQL Server Data Tools -- good

<https://www.mssqltips.com/sqlservertip/2804/introduction-to-sql-server-data-tools/>

Exporting from SQLite to SQL Server --- good

<https://www.mssqltips.com/sqlservertip/1896/dynamically-import-data-from-a-foreign-database-using-a-sql-server-linked-server/>

14. Copy/Move, backup/restore, detach/attach SQL database

# [SQL Server backup/restore vs. detach/attach](https://stackoverflow.com/questions/610113/sql-server-backup-restore-vs-detach-attach)

<https://stackoverflow.com/questions/610113/sql-server-backup-restore-vs-detach-attach>

15. running total using CTE

<http://blog.sqltechie.com/2009/04/how-to-do-running-sum-using-cte-sql.html>

### 16. B. Using the OVER clause with aggregate functions -- very good

<https://docs.microsoft.com/en-us/sql/t-sql/queries/select-over-clause-transact-sql>

17. CTE and Running TOT using Recursive query

How does a Recursive CTE run, line by line?

<https://stackoverflow.com/questions/3187850/how-does-a-recursive-cte-run-line-by-line>

explain recursive CTE

<https://technet.microsoft.com/en-us/library/ms186243(v=sql.105).aspx>

<https://www.essentialsql.com/recursive-ctes-explained/>

<https://www.mssqltips.com/sqlservertip/1520/recursive-queries-using-common-table-expressions-cte-in-sql-server/>

Running total – good example

<http://blog.sqltechie.com/2009/04/how-to-do-running-sum-using-cte-sql.html>

<http://bretstateham.com/using-ctes-to-build-running-sums/> --- good

<https://www.codeproject.com/Articles/300785/Calculating-simple-running-totals-in-SQL-Server>

<https://stackoverflow.com/questions/860966/calculate-a-running-total-in-sql-server>

18. Select … Over … clause --- goood 14/11/2017

<https://docs.microsoft.com/en-us/sql/t-sql/queries/select-over-clause-transact-sql>

19. ODBC for SQLite database Articles

good

<https://www.mssqltips.com/sqlservertip/3087/creating-a-sql-server-linked-server-to-sqlite-to-import-data/>

<http://web.synametrics.com/SQLite.htm>

20. WINSql software video

<http://web.synametrics.com/MovieFeatureList.htm>

21. x86 ( 32 bits ) vs x64 bits

<http://www.tomshardware.com/forum/28592-63-what>

22. Pivot or unPivot table

<https://blogs.msdn.microsoft.com/spike/2009/03/03/pivot-tables-in-sql-server-a-simple-sample/>

Tutorial & knowledge base – very good

<https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx>

<https://www.red-gate.com/simple-talk/sql/t-sql-programming/questions-about-pivoting-data-in-sql-server-you-were-too-shy-to-ask/>

Example

<https://www.mssqltips.com/sqlservertip/3000/use-sql-servers-unpivot-operator-to-help-normalize-output/>

<https://www.codeproject.com/Tips/500811/Simple-Way-To-Use-Pivot-In-SQL-Query>

23. Find out sql server version

<https://www.mssqltips.com/sqlservertip/1140/how-to-tell-what-sql-server-version-you-are-running/>

# **24. SQL Server Best Practices – Implementation of Database Object Schemas – very good especially for interview and presentation for knowledge base 24/11/2017**

<https://technet.microsoft.com/en-us/library/dd283095(v=sql.100).aspx>

25. Remove UTF-8 byte order marks. 27/11/2017

<https://www.ablebits.com/office-addins-blog/2014/04/24/convert-excel-csv/>

https://stackoverflow.com/questions/2317605/escaping-non-ascii-characters-or-how-to-remove-the-bom

26. Copy a database to another computer 30/11/2017

<https://stackoverflow.com/questions/8715687/how-to-copy-a-database-from-one-computer-to-another>

<https://cybertext.wordpress.com/2008/02/05/restoring-an-sql-express-database-on-another-computer/>

<https://stackoverflow.com/questions/6183139/sql-server-database-backup-restore-on-lower-version>

<https://www.devart.com/dbforge/sql/studio/copy-database.html>

<https://docs.microsoft.com/en-us/sql/relational-databases/databases/copy-databases-with-backup-and-restore>

27. Compatibility 1/12/2017

<https://www.spiria.com/en/blog/web-applications/understanding-sql-server-compatibility-levels>

28. Migrate dB from higher version to lower version – very good

<https://www.mssqltips.com/sqlservertip/2810/how-to-migrate-a-sql-server-database-to-a-lower-version/>

29. Microsoft odbc driver system32 or SysWow64

<https://stackoverflow.com/questions/1134827/how-do-i-use-a-32-bit-odbc-driver-on-64-bit-server-2008-when-the-installer-doesn>

30. Excel VBA access to sqlite db 4/12/2017

<https://stackoverflow.com/questions/42509154/accessing-a-sqlite-database-in-vba-in-excel>

<https://stackoverflow.com/questions/1346979/how-do-i-access-sqlite-from-vba>

<http://pragmateek.com/using-an-access-mysql-postgresql-or-sqlite-database-from-vba/>

<https://www.codeproject.com/Articles/586361/UsingplusanplusAccess-cplusMySQL-cplusPostgreSQL>

<https://www.mrexcel.com/forum/excel-questions/435486-sqlite-excel-vba.html>

31. SQLite 3 to SQL Server 2008R2 --- 5/12/2017

<http://www.tek-tips.com/viewthread.cfm?qid=1425555>

<https://access-programmers.co.uk/forums/showthread.php?t=99429>

<https://stackoverflow.com/questions/37448636/excel-vba-insert-array-variable-into-single-field-in-access>

<https://social.msdn.microsoft.com/Forums/en-US/2a49fdd0-0b4a-4257-84e1-b471102bca21/vba-in-excel-for-inserting-and-updating-records-in-sql-server-table?forum=exceldev>

32. recordset

<https://www.reddit.com/r/excel/comments/2750w1/insert_into_sql_server_from_excel_with_vba/>

<http://www.tek-tips.com/viewthread.cfm?qid=1454845>

33. date in SQL Server – very good

<http://sqlhints.com/2013/07/14/how-to-get-date-part-only-from-datetime-in-sql-server/>

34. SQL tips and tricks – very good

<https://www.sqlshack.com/10-ssms-tips-and-tricks-to-boost-your-productivity/>

<http://www.sqlmatters.com/Articles/Top%2010%20SQL%20Server%20Management%20Studio%20(SSMS)%20Tips%20and%20Tricks.aspx>

35. Pass multi value to one parameter in store procedure. - 14/12/2017

<https://stackoverflow.com/questions/43249/t-sql-stored-procedure-that-accepts-multiple-id-values?rq=1>

<https://stackoverflow.com/questions/617706/passing-an-in-list-via-stored-procedure?noredirect=1&lq=1>

<https://raresql.com/tag/sql-server-passing-multiple-values-through-one-parameter-in-a-stored-procedure/>

* Good

<https://www.sqlservercentral.com/Forums/Topic1368492-391-1.aspx>

<https://sqlwithmanoj.com/2015/07/10/passing-multiple-values-with-a-single-parameter-in-a-stored-procedure-sql-server/> - very good

<https://raresql.com/tag/sql-server-passing-multiple-values-through-one-parameter-in-a-stored-procedure/>

36. Bcp terminator

<https://www.red-gate.com/simple-talk/sql/database-administration/creating-csv-files-using-bcp-and-stored-procedures/>

37. Order by in olap over() In SQL

<https://explainextended.com/2009/03/08/analytic-functions-sum-avg-row_number/> - very good

38. Use Views in Sql – use bcp to export or import data

<https://technet.microsoft.com/en-us/library/ms188250(v=sql.105).aspx>

<https://technet.microsoft.com/en-us/library/ms187086(v=sql.105).aspx>

<https://technet.microsoft.com/en-us/library/ms188250(v=sql.105).aspx>

<https://stackoverflow.com/questions/2680207/what-is-a-good-reason-to-use-sql-views>

39. Views Vs Store procedures – why use Views etc

<https://www.google.com.au/search?ei=bbE1WtLFHYn68gWIkrSwCg&q=sql+stored+procedure+vs+view&oq=sql+stored+procedure+vs+&gs_l=mobile-gws-serp.1.1.0l5.1587.5772.0.7217.4.4.0.0.0.0.1087.3813.6-3j1.4.0....0...1.1.64.mobile-gws-serp..1.3.2850...0i67k1.0.dZ5s9iSUM-A>

https://stackoverflow.com/questions/5194995/what-is-the-difference-between-a-stored-procedure-and-a-view

40. ---- Convert format Date in SQL server --- 16/12/2017

<http://www.sql-server-helper.com/tips/date-formats.aspx>

41. Sql calculate standard deviation - 7/1/2018

<https://www.tutorialgateway.org/sql-stdev-function/>

42. how-to-execute-stored-procedure-in-sql-with-multip 11/1/2018

<https://www.codeproject.com/Questions/800672/how-to-execute-stored-procedure-in-sql-with-multip>

43. SQL function

<https://www.techrepublic.com/blog/the-enterprise-cloud/understand-when-to-use-user-defined-functions-in-sql-server/>

<http://www.c-sharpcorner.com/UploadFile/37db1d/creating-and-managing-user-defined-functions-in-sql-server-2008/>

<http://www.informit.com/articles/article.aspx?p=31673&seqNum=4>

<https://stackoverflow.com/questions/26354172/how-to-pass-parameters-to-microsoft-sql-server-query>

44. SQLite update with CTE - 22/1/18

<http://sqlite.1065341.n5.nabble.com/Syntax-error-using-CTE-and-UPDATE-td97280.html>

<http://sqlite.1065341.n5.nabble.com/SQLite-Update-With-CTE-td97012.html>

<http://sqlite.1065341.n5.nabble.com/Syntax-error-using-CTE-and-UPDATE-td97280.html>

45. how to rename a column – 24/2/2018 – good for interview and 平时talking

<https://docs.microsoft.com/en-us/sql/relational-databases/tables/rename-columns-database-engine>

You can rename a table column in SQL Server 2017 by using SQL Server Management Studio or Transact-SQL.

## **Before You Begin**

### **Limitations and Restrictions**

Renaming a column will not automatically rename references to that column. You must modify any objects that reference the renamed column manually. For example, if you rename a table column and that column is referenced in a trigger, you must modify the trigger to reflect the new column name. Use [sys.sql\_expression\_dependencies](https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-sql-expression-dependencies-transact-sql) to list dependencies on the object before renaming it.

46. what is trigger – 24/2/2018 -- good for interview and 平时talking

<https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql>

# **CREATE TRIGGER (Transact-SQL)**

Creates a DML, DDL, or logon trigger. A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view. These triggers fire when any valid event is fired, regardless of whether or not any table rows are affected. For more information, see [DML Triggers](https://docs.microsoft.com/en-us/sql/relational-databases/triggers/dml-triggers).1

DDL triggers execute in response to a variety of data definition language (DDL) events. These events primarily correspond to Transact-SQL CREATE, ALTER, and DROP statements, and certain system stored procedures that perform DDL-like operations. Logon triggers fire in response to the LOGON event that is raised when a user sessions is being established. Triggers can be created directly from Transact-SQL statements or from methods of assemblies that are created in the Microsoft .NET Framework common language runtime (CLR) and uploaded to an instance of SQL Server. SQL Server allows for creating multiple triggers for any specific statement.

# **CREATE TRIGGER (Transact-SQL)**

Creates a DML, DDL, or logon trigger. A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view. These triggers fire when any valid event is fired, regardless of whether or not any table rows are affected. For more information, see [DML Triggers](https://docs.microsoft.com/en-us/sql/relational-databases/triggers/dml-triggers).1

DDL triggers execute in response to a variety of data definition language (DDL) events. These events primarily correspond to Transact-SQL CREATE, ALTER, and DROP statements, and certain system stored procedures that perform DDL-like operations. Logon triggers fire in response to the LOGON event that is raised when a user sessions is being established. Triggers can be created directly from Transact-SQL statements or from methods of assemblies that are created in the Microsoft .NET Framework common language runtime (CLR) and uploaded to an instance of SQL Server. SQL Server allows for creating multiple triggers for any specific statement.

47. Split a string using Function – 27/2/2018

<https://community.spiceworks.com/topic/1822586-sql-split-string-to-columns-in-function>

Good

<http://sqlhints.com/tag/split-comma-separated-values-in-sql/>

Good

<https://www.aspsnippets.com/Articles/Split-function-in-SQL-Server-Example-Function-to-Split-Comma-separated-Delimited-string-in-SQL-Server-2005-2008-and-2012.aspx>

Good

<https://stackoverflow.com/questions/20512855/split-function-by-comma-in-sql-server-2008>

<https://codeworldtechnology.wordpress.com/2016/09/05/how-to-split-comma-separate-string-in-sql-server/>

Good

<https://stackoverflow.com/questions/10581772/how-to-split-a-comma-separated-value-to-columns>

48. how to execute View – 9/3/2018

<https://stackoverflow.com/questions/3533757/how-to-execute-all-views-in-database-through-stored-procedure>

<https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql>

<https://www.codeproject.com/Articles/266985/Views-In-SQL-Server>

49. temp table Vs table variables - 16/3/218

<https://www.red-gate.com/simple-talk/sql/t-sql-programming/temporary-tables-in-sql-server/>

<https://www.mssqltips.com/sqlservertip/1556/differences-between-sql-server-temporary-tables-and-table-variables/>

50. Except – Exist – In operator 5/4/2018

<https://stackoverflow.com/questions/1662902/when-to-use-except-as-opposed-to-not-exists-in-transact-sql>

EXISTS is much faster than IN when the subquery results is very large.  
IN is faster than EXISTS when the subquery results is very small.

51. not exists – not in – except 5/4/2018

<https://stackoverflow.com/questions/14479005/not-exists-vs-not-in>

You are correct that there is a big difference with nulls. A NOT IN query checks that each element definitively does not match. A comparison with null doesn't produce a definitive result. Thus if your subquery contains a null, nothing will be considered "NOT IN" it.

[See this SQL Fiddle example.](http://www.sqlfiddle.com/#!3/08bf7/1)

The unintuitive side effect of this behavior is that NOT IN is not actually the opposite of IN.

A NOT EXISTS query does not have this problem.

As you have rightly said the two are different things. If the subquery of items to not be IN contains NULL no results will be returned because nothing equals NULL and nothing does not equal NULL(Not even NULL).

Assuming you are using the two to achieve the same result, there is no difference between the two as long as you handle NULL values in your IN statement. The optimiser is clever enough to know that with NULL values eliminated, or with non nullable columns the two are the same, so use the same ANTI SEMI JOIN.

Consider these two tables:

## 52.

## [SQL Server: JOIN vs IN vs EXISTS - the logical difference](http://weblogs.sqlteam.com/mladenp/archive/2007/05/18/60210.aspx) -- good

There is a common misconception that IN behaves equaliy to EXISTS or JOIN in terms of returned results.

This is simply not true. To see why not, let's review what each statement does.

**IN:**

Returns true if a specified value matches any value in a subquery or a list.

**Exists:**

Returns true if a subquery contains any rows.

**Join:**

Joins 2 resultsets on the joining column.

If not read carefully it looks pretty same so far, doesn't it.

**The difference** comes when you take into account the "dreaded" **THREE-VALUED LOGIC**.

Let's review that also:

-- this is true

SELECT 1 WHERE 1 = 1

-- this is false

SELECT 1 WHERE 1 = 0

-- this is unknown - it is usually expected to be false, but that only shows

-- misunderstanding of nulls. It's not false it's only treated as false in the filter

SELECT 1 WHERE 1 = NULL

-- this is also unknown - but logicaly it would seem it would be true

-- but unknown compared to unknown equals uknown and it is treated as false in the filter

SELECT 1 WHERE NULL = NULL

The where will return a row only if the condition evaluates to true which UNKNOWN isn't.

<http://weblogs.sqlteam.com/mladenp/archive/2007/05/18/60210.aspx>

53. SQL in the Wild > SQL Server > T-SQL > NOT EXISTS vs NOT IN -- good

<https://www.sqlinthewild.co.za/index.php/2010/02/18/not-exists-vs-not-in/>

--Let’s say, for illustration purposes that there are 4 rows in the table called t, there’s a column called ID with values 1..4

WHERE SomeValue NOT IN (SELECT AVal FROM t)

is equivalent to

WHERE (

SomeValue != (SELECT AVal FROM t WHERE ID=1)

AND

SomeValue != (SELECT AVal FROM t WHERE ID=2)

AND

SomeValue != (SELECT AVal FROM t WHERE ID=3)

AND

SomeValue != (SELECT AVal FROM t WHERE ID=4)

)

Let’s further say that AVal is NULL where ID = 4. Hence that != comparison returns UNKNOWN. The logical truth table for AND states that UNKNOWN and TRUE is UNKNOWN, UNKNOWN and FALSE is FALSE. There is no value that can be AND’d with UNKNOWN to produce the result TRUE

Hence, if any row of that subquery returns NULL, the entire NOT IN operator will evaluate to either FALSE or UNKNOWN and no records will be returned

So what about EXISTS?

Exists cannot return NULL. It’s checking solely for the presence or absence of a row in the subquery and, hence, it can only return true or false. Since it cannot return NULL, there’s no possibility of a single NULL resulting in the entire expression evaluating to UNKNOWN.

Hence, when the column in the subquery that’s used for comparison with the outer table can have nulls in it, consider carefully which of NOT EXISTS or NOT IN you want to use.

# 55. [**When to use EXCEPT as opposed to NOT EXISTS in Transact SQL?**](https://stackoverflow.com/questions/1662902/when-to-use-except-as-opposed-to-not-exists-in-transact-sql)

Very good !

<https://stackoverflow.com/questions/1662902/when-to-use-except-as-opposed-to-not-exists-in-transact-sql>

# 56. [**Difference between View and table in sql**](https://stackoverflow.com/questions/6015175/difference-between-view-and-table-in-sql) - 9/5/2018

<https://stackoverflow.com/questions/6015175/difference-between-view-and-table-in-sql>

57. Database VIEW does not reflect the data in the underying TABLE

<https://stackoverflow.com/questions/1013711/database-view-does-not-reflect-the-data-in-the-underying-table>

# 58. Table changes not automatically reflected in a SQL Server View – very good

When a view is created in SQL Server, metadata for the referenced table columns (column name and ordinal position) is persisted in the database. Any change to the referenced base table(s) (column re-ordering, new column addition, etc) will not be reflected in the view until the view is either:

 Altered with an ALTER VIEW statement

 Recreated with DROP VIEW/CREATE VIEW statements

 Refreshed using system stored procedure sp\_refreshview

<https://www.mssqltips.com/sqlservertip/1427/table-changes-not-automatically-reflected-in-a-sql-server-view/>

# 59. Refresh all views in a database – very good

<http://www.sqlservercentral.com/scripts/Views/147316/>

# 60. [**Are Views automatically updated**](https://stackoverflow.com/questions/7755358/are-views-automatically-updated) -- good

<https://stackoverflow.com/questions/7755358/are-views-automatically-updated>

61. **Do SQL views update automatically? - this is best answer**

<https://www.quora.com/Do-SQL-views-update-automatically>

In SQL Server a standard view is nothing but a query. As the underlying data changes, when that query gets accessed, it reflects the changes to the data. It's automatic, but not because some additional action is occurring to refresh the data in the view. Rather, the view, being just a query, retrieved the data from the tables in question.

Grant Fritchey provides a great answer, I would add one thing though. While a view will show changes to the data modified via inserts, updates, or deletes, The same is not always true in regards to changes to the schema of the underlying tables referenced in the view. Consider the following.

# 62. Difference Between Table and View

<http://www.differencebetween.net/technology/hardware-technology/difference-between-table-and-view/>

63. <https://www.quora.com/What-is-the-difference-between-tables-and-views-in-SQL-How-are-views-useful>

# 64. **When to Use Temporary Tables vs. Table Variables -- good**

<https://www.sqlshack.com/when-to-use-temporary-tables-vs-table-variables/>

# 65. **A DBA guide to SQL Server performance troubleshooting – Part 1 – Problems and performance metrics - very good ( even for interview )**

And follow the link to read all ‘4 parts’ of guide to Sql Server Performance troubleshooting by [Milena Petrovic](https://www.sqlshack.com/author/milenapetrovic/) !

<https://www.sqlshack.com/dba-guide-sql-server-performance-troubleshooting-part-1-problems-performance-metrics/>

# 66. Benefits of SCHEMABINDING in SQL Server – good english

<https://www.mssqltips.com/sqlservertip/4673/benefits-of-schemabinding-in-sql-server/>

##### Solution

In addition to functions, WITH SCHEMABINDING can also be used with views and stored procedures, and there can be tangible benefits there that don't involve performance. Mainly these revolve around either preventing unauthorized or inadvertent changes, or preventing unexpected results.

## Performance Troubleshooting

Okay, I said non-performance-related benefits, but there may be cases where Halloween protection is in place, causing a more expensive DML plan, but it is not obvious to the person trying to troubleshoot performance. So it is important to be able to identify that this might be the cause - as Paul White explained at the end of [his last post on the Halloween problem](https://sqlperformance.com/2013/02/sql-plan/halloween-problem-part-4), this protection "will not always show up as an extra Eager Table Spool, and scalar function calls may be hidden in views or computed column definitions, for example."

##### Summary

I am a big fan of SCHEMABINDING, and there are no valid downsides I can think of to always use WITH SCHEMABINDING on relevant functions and views. Some will say that having to touch additional modules to make changes to a single object might be a downside, but I don't see it that way at all. In fact I think it is essential to be forced to take all of these dependent objects into account before making a schema change of any kind. And in some cases, you will just happen to enjoy performance benefits as well.

# 67. [**NOT IN clause and NULL values**](https://stackoverflow.com/questions/129077/not-in-clause-and-null-values) -- Very good --- 22/6/2018

# <https://stackoverflow.com/questions/129077/not-in-clause-and-null-values>

## 68. SQL best practice – very good --- 4/7/2018

<https://data36.com/sql-best-practices-data-analysts/>

This is not even a best practice, this is a must.

## SQL best practice #4: Comments

Commenting your code is a highly recommended in any programming languages. For instance if you are working in a team, it helps the team to understand your code. But you should comment for your own best interest too! If you write a query today, I guarantee that in 1 year you won’t remember at all, what the heck that COUNT(origin) function was good for… So the next SQL best practice is about helping yourself remember stuff!  
I like to look at commenting as sending messages to myself in the future or in an alternate dimension. And it’s always really nice to receive these kind messages from my past-self (of course most of the times I’m also very angry, why past me couldn’t be more detailed).

Anyway. Commenting is important, no question around that. But how can you do it in SQL?  
Very simple. Type this: --. After double-dash, nothing in that line will be executed.

## SQL best practice #5: ORDER BY column name

There is a little “trick” you will meet sooner or later. So I want you to hear it first from me – to avoid any trouble with it in the future! When you use ORDER BY and GROUP BY, you can of course use column names (as we have just done that so far) or instead of that you can use column numbers (that’s the new thing).

Try it yourself… and then **never use it again**!  
Wait, what? Why?  
Imagine a very common scenario: you tweak your query and you add an extra column (eg. SUM(depdelays)) as first in your SELECT statement… If you use column names, everything will just work fine. If you use column numbers instead, your query will order by the wrong column (because you should have changed the column number in the ORDER BY clause too, but you might have forgotten it).  
So do yourself a favour and don’t add unnecessary points of failure to your query – your job will be complicated enough without them too!  
So the fifth SQL best practice is: Always use the actual name of the columns, when you refer to them (either in ORDER BY or GROUP BY) and never use the number.

## SQL best practice #6: Avoid SELECT \*

Remember to my first [SQL article](https://data36.com/sql-for-data-analysis-tutorial-beginners/) ? I said, this is the base query, you have to learn, use and expand most of the time:



Now you are good enough in SQL to learn the terrible truth: we don’t use \* in real data projects’ real queries – at all.  
There are a tons of [good reasons](https://stackoverflow.com/questions/3639861/why-is-select-considered-harmful) for that, but the top two are: efficiency and readability. If you need one column from a table, why would you print all of them? It would mean you have to move more data from your SQL server to your computer – slowing down processes unnecessarily. And at the same time, if you add \* and not column names in your query, you won’t have any clue, what columns you have in your data table, when you want to change something…  
Use column names, not \*.

69..

# The Seven Sins against TSQL Performance -- good -- 4/7/2018

<https://www.red-gate.com/simple-talk/sql/performance/the-seven-sins-against-tsql-performance/>

There are three very basic rules for writing T-SQL that performs. They are cumulative. Doing all of these will have a positive impact. Skipping or changing any of these is likely to have a negative impact on your code’s performance.

* **Write to your data structures**: If you are storing a datetime data type, use that, not a varchar or something else. Also take advantage of the foreign key constraints and other structures that you have in place when you’re writing your T-SQL code
* **Write for your indexes**: If there are indexes on your tables, and there should be, write your code so that it can take advantage of those indexes. Make sure that your clustered index, you only get one, is working well to help drive your queries to your data in an efficient manner.
* **Write for the optimizer**: The query optimizer is an amazing piece of software. But, you can overwhelm it by writing code that isn’t configured best to support it, such as nesting views within views within views, etc. Take the time to understand how the optimizer works and write your code in such a way that you help it, not hurt it.

Then, there are some very specific and common mistakes that people make in their T-SQL code. Just don’t do these and you’ll be much better off.

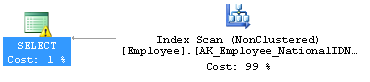
## **Using the wrong data types**

This is surprisingly simple in concept, but seems to be incredibly difficult in practice. Here you go… use the data type that is in your database. Use it in your parameters and in your variables. I know that SQL Server can implicitly convert from one to another. But when you get implicit conversions, or you have to put in explicit conversions, you’re performing a function on your columns. When you perform a function on your columns in any of the filtering scenarios, that’s a WHERE clause or JOIN criteria, you’re looking at generating table scans. You may have a perfectly good index, but because you’re doing a CAST on the column in order to compare to a character type that you passed in instead of a date, that index won’t get used.

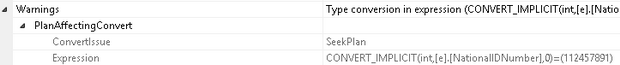
Don’t believe me? Check this query:

|  |  |
| --- | --- |
|  | SELECT  e.BusinessEntityID,          e.NationalIDNumber  FROM    HumanResources.Employee AS e  WHERE   e.NationalIDNumber = 112457891; |

Nice & simple. Should be covered by an index on the table. But here’s the execution plan:



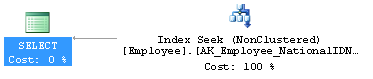
This query ran fast enough, and the table is small, so there were only 4 reads to scan the entire thing. Notice that little exclamation point on the SELECT operator though. What do the properties tell us:



That’s right. It’s a warning (new in SQL Server 2012) that you have a “PlanAffectingConvert.” In short, because I didn’t use the correct data type:

|  |  |
| --- | --- |
|  | SELECT  e.BusinessEntityID,          e.NationalIDNumber  FROM    HumanResources.Employee AS e  WHERE   e.NationalIDNumber = '112457891'; |

I didn’t get a plan that looked like this:



And there were only 2 reads this time instead of 4. And yes, I realize I only cut performance from a little to a very little, but what if that table had millions of rows instead of only a few? Ah, now I’m suddenly a hero.

Use the correct data types.

# 70. **Tips and Best Practices for SQL Programmers -- good**

<https://www.upwork.com/hiring/data/tips-and-best-practices-for-sql-programmers/>



[Structured query language (SQL)](https://www.upwork.com/hiring/for-clients/sql-developer-job-description/) is the primary language for databases. Whatever [relational database](https://www.upwork.com/hiring/data/a-guide-to-database-technology/) platform you decide to use, you’ll need to use SQL to edit, insert, delete, and query records within the database. Each platform has small differences between their respective SQL syntax, but logic and development are still similar. The language is different than [NoSQL](https://www.upwork.com/hiring/data/sql-vs-nosql-databases-whats-the-difference/), which has similarities but mainly different syntax. Here are some tips and best practices for any SQL development project.

[Join Upwork](https://www.upwork.com/signup/?dest=home&forceHipo=1)

### **1. THINK ABOUT RELATIONSHIPS**

The SQL language is the language of relational databases. You need to build tables based on relationships without creating “orphans.” Orphans are tables that don’t have a relationship with any other table and are poor database design. Each table should have a relationship with another, so when you query the database for records, you can join the tables together through their designated relationship.

You build a relationship through **primary** and **foreign** **keys**. For instance, suppose you have a customer table and an order table. Without the customer ID in the order table, the order table is orphaned. By placing the customer ID in the order table, you now have a relationship between the two tables that you can use to create queries.

### **2. ALWAYS CONSIDER PERFORMANCE, EVEN IF THE QUERY SEEMS SIMPLE**

Just because your SQL data set has few records currently doesn’t mean the data set won’t grow significantly after you move it to production. You should join tables on indexes and always have a primary key in each table.

One common mistake is to run a poorly performing query in development when the system has no users querying and very few records. The query runs fast, because there’s no load on the database server. Then, when the query is promoted to production and run in a busy environment, the query performs poorly and undermines site performance. Always consider performance even if the query seems like it won’t need many resources from the database server.

### **3. USE TABLE ALIASES FOR EASIER READING**

Let’s say you’re a developer who needs to maintain someone else’s code. Or, you’re a business owner who needs to [hire a SQL developer](https://www.upwork.com/hire/sql-programmers/) to fill in the gaps with a specific project. The SQL developer can pick up where another developer left off much quicker if the code uses aliases and is well formatted.

**Aliases** shorten the name of a table and make it easier to read and determine the logic in a SQL statement. When you build a database, always consider the way the code is written. It should be easy for you and another developer to determine its logic and data set. Poorly coded SQL code can lead to bugs should another developer need to edit it.

### **4. BE SPECIFIC IN YOUR SELECT CLAUSES**

In any common SQL language, the asterisk ( \* ) tells the database engine to return all columns within a table. There are several issues with this habit. The first issue is security. If a hacker is able to use a SQL injection attack on your database, it could leave every column available for theft. If you have customer passwords stored in the table, the attacker can expose passwords.

The second issue is performance. If you have a million records returned from a query, you affect performance when you return a couple dozen columns rather than the few that you need.

The third issue is that it’s difficult to determine what data will be returned. If you specify columns in your SELECT queries, you know exactly which ones are returned to the front-end application. For these three reasons, always specify the columns you want to return and avoid using the “all” or asterisk character in your queries.

### **7. DON’T FORGET OBJECT-LEVEL SECURITY**

Security should always be a priority when designing a database regardless of the platform you use. Object-level permissions provide security based on the user and what the account can access. For instance, you want to use a separate user name for all of your databases and give them access to the database accordingly. By using this method, if a hacker gains access to the account, they’ll only be able to access one database and not the entire server.

You can be even more specific with object-level permissions. This tells the database the exact tables and even columns that the user can access. By using object-level permissions, you limit the amount of exposure if an attacker is successful at SQL injection.

For instance, suppose you store social security numbers in the database. A public-facing user application should never have access to this information. You only want internal employees to have access. So, you create a public-facing user account and give it object-level permissions to the table, then access rights to specific columns, excluding the social security column. If a hacker gains access to this account, they won’t be able to view social security numbers, which limits your data breach exposure, risk, and damages.

### **6. AVOID DYNAMIC SQL**

SQL injection is one of the most common attacks on the web. This type of attack leads to severe data breaches that expose millions of records to an attacker. Extremely strategic SQL injection attacks can even elevate permissions for the attacker to give them administrative rights on the database server.

Dynamic SQL is a type of coding that builds a SQL statement based on input from a user. Typically, it’s done on the front-end application side, but some SQL developers use it too. Dynamic SQL should be avoided at all costs—it’s this type of development that leads to SQL injection exposure.

If you absolutely need to use dynamic SQL, you should “scrub” data input and make good attempts to detect malformed SQL statements. You can use third-party tools or integrated tools available with some languages. For instance, the C# language has the [TSqlParser](https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.transactsql.scriptdom.tsqlparser(v=sql.120).aspx) class to help programmers identify SQL injection in a dynamic SQL environment.

# 71. T – SQL Best Practices

# T-SQL Best Practices, Part 1 -- good

<http://www.itprotoday.com/software-development/t-sql-best-practices-part-1>

Every programming language has best and worst practices, and T-SQL is no exception. Some best practices in T-SQL are similar to those applicable to procedural languages, but most actually have to do with the fact that T-SQL is based on the relational model. This article is the first in a two-part series that highlights several important T-SQL best practices. (However, please don't consider my list as complete.)

**Related:** [T-SQL Best Practices, Part 2](http://sqlmag.com/t-sql/t-sql-best-practices-part-2)

### Think in Sets

Thinking in sets, or more precisely, in relational terms, is probably the most important best practice when writing T-SQL code. Many people start coding in T-SQL after having some background in procedural programming. Often, at least at the early stages of coding in this new environment, you don't really think in relational terms, but rather in procedural terms. That's because it's easier to think of a new language as an extension to what you already know as opposed to thinking of it as a different thing, which requires adopting the correct mindset.

If that's not enough, there are two levels of adaptation you need to go through. The first is not thinking in procedural terms, but rather in T-SQL terms. The second is realizing that even T-SQL isn't perfect as a relational language and understanding how to use it relationally.

Here I'll just lay out the key elements you need to keep in mind. For deeper coverage of this topic, see "[T-SQL Foundations: Thinking in Sets](http://www.itprotoday.com/article/tsql/thinking-in-sets-129772)."

When I say "not thinking in procedural terms," I'm talking about generally avoiding iterative constructs such as cursors and loops, in which you process data one row at a time in specific order. Remember that tables in T-SQL are supposed to represent sets of rows; sets need to be handled as a whole, and they have no particular order to their elements. I won't tell you to never touch a cursor, because I recognize that there are cases in which, for practical reasons, it's difficult to avoid them -- however, you should consider a set-based approach by default.

As I mentioned, a set has no order, so you need to remember that neither tables nor views are ordered. This sounds easy, but in reality it's very difficult for some people to accept this fact. You need to realize that regardless of what kind of physical structures exist, such as indexing, when you query a table or a view, you aren't supposed to assume any particular physical order of processing the rows. Also, unless you add an ORDER BY clause to the outermost query, the order of the rows returned isn't guaranteed. For these reasons, attempts to create a "sorted view" -- for example, by adding TOP (100) PERCENT to the query -- are wrong from a relational perspective and aren't guaranteed to return rows from a view in any particular order when the outer query against the view doesn't have an ORDER BY clause.

You must understand that the T-SQL concepts of table, row, and column have very different meanings from file, record, and field, which people often use mistakenly.

After you stop thinking in iterative, ordered, one-row-at-a-time terms and start thinking in terms of sets, you've taken the most important leap. A second level of maturity in programming with T-SQL is to realize that T-SQL isn't perfect as a relational language, but it provides you with enough tools that you can use it relationally if you understand the relational model. Chris Date does a great job explaining this topic in his book [*SQL and Relational Theory*](http://shop.oreilly.com/product/0636920022879.do), which I strongly recommend that all T-SQL practitioners read. You need to understand the difference between the T-SQL concepts of table, column, and row and their relational counterparts: relation, attribute, and tuple. (A full discussion of this topic is beyond the scope of this article.)

### **Think of NULLs and the Three-Valued Logic**

NULLs are a source of quite a lot of complexity and confusion in T-SQL. In essence, a NULL is supposed to represent a missing value that could be either applicable (exists but we don't know what it is) or inapplicable (doesn't exist, or is irrelevant). To start with terminology, NULL is a mark for a missing value -- not a value -- therefore, saying "NULL value" is incorrect.

The specific example is less important than the general recommended best practice: Be mindful of NULLs for every piece of code you write in T-SQL. If NULLs are possible in the data, make sure you include them in your tests. And if the default language treatment isn't the one you need, make sure you make the correct adjustments to make it behave the way you need.

### Adopt Good Coding Practices

There are some coding best practices that experienced database practitioners have heard a thousand times, but they're still so important that I want to make sure I mention them.

**Terminating statements with a semicolon.** In the past, T-SQL didn't have a requirement to terminate all statements with a semicolon. However, standard SQL requires it. Also, with each new version of SQL Server, this becomes a requirement for more and more specific cases to avoid parsing ambiguity. SQL Server 2008 already added "not terminating statements with a semicolon" to the list of deprecated features. So do yourself a favor and if you haven't done so already, start getting used to terminating all of your statements.

**Avoid \*.** Using \* in the SELECT list is a bad practice in most cases. Changes in the underlying table definition might cause problems for applications that don't expect them and in table expressions such as views defined based on the query. You also want to be able to rely efficiently on covering indexes when you're truly interested in only a subset of the columns.

**Schema-qualified object names.** Always use schema-qualified object names (Sales.Orders as opposed to just Orders) both when creating and when querying them. Different users can have different default schemas. Also, even though the savings is very minor and probably negligible, explicitly indicating a schema means you save the cost of the instructions involved in implicit resolution. Furthermore, a query plan produced by a user with one default schema can't be reused by a user with a different default schema. You'll get multiple optimizations with multiple unnecessary execution plans in cache.

**Have a style and use correct casing.** I find it very important to use the correct case for identifiers of objects and columns. I've seen a lot of code in which people weren't mindful of this rule; in case-insensitive environments, it typically doesn't matter much. However, what if at some point in the future you need to make your databases case-sensitive? Why not write in a way that won't get you into trouble if such changes are required in the future? Also, code that uses different casing for the same identifiers in different places doesn't look good. Consistency can go a long way in code readability and maintainability.

For similar reasons, I find it important to develop some style for your code. Have you seen queries that look like this:

There's not a specific style that's better than the rest, but it's important to have a style that's readable and natural, and to be consistent with it. Doing so improves the readability and maintainability of your code.

***Avoid the old-style join syntax.*** Some people find it difficult to adopt the ANSI SQL-92 syntax for joins with the JOIN keyword and the ON clause (e.g., FROM T1 INNER JOIN T2 ON T1.keycol = T2.keycol), as opposed to using a comma between the table names and then all predicates in the WHERE clause (e.g., FROM T1, T2 WHERE T1.keycol = T2.keycol).

At any rate, there are two main reasons why you should stick to the newer syntax for all joins. One is consistency; it simply doesn't look good when you try mixing the different styles. Another is the fact that if you use the old-style syntax and forget to indicate the WHERE clause with the join predicate, you get an unintentional cross join. With the newer syntax, if you use an inner join and forget the ON clause, you get a parsing error that forces you to fix the code.

# T-SQL Best Practices, Part 2

<http://www.itprotoday.com/software-development/t-sql-best-practices-part-2>

T-SQL, probably more than any other programming language, is prone to misuse. Last month I started a two-part series about following T-SQL best practices and avoiding bad practices (see "[T-SQL Best Practices, Part 1](http://www.itprotoday.com/article/tsql/tsql-best-practices-141536)").  I provided recommendations that have to do with the mathematical foundations to the language as well as coding best practices. This month I continue the coverage by discussing cloud compatibility, date and time best practices, the importance of writing in a standard way, and performance-related best practices.

### **Think Cloud**

Microsoft is investing a lot in cloud technology, making it richer and more powerful. For some applications, it's inevitable that the database will reside in the cloud at one point or another. A lot of effort is spent on having one code base for box (on-premises) and cloud versions (SQL Azure), but there are some differences you should be aware of. Understanding the differences can affect the way you write your code if you want a smooth migration (if and when you migrate).

### **Date and Time Best Practices**

There are many best practices related to working with date and time data. However, I want to mention two particularly common bad habits that I recommend avoiding.

**Use language-neutral literals.**

**Be careful about rounding errors.**

### Beware of Rules of Thumb

People like rules of thumb, but they don't always exist. I've seen many such rules that were probably created based on very limited observations and then made a rule. Have you heard rules of thumb such as "Use joins because they're more efficient than subqueries," or "If you have more than n tables in a single query, break it to multiple queries"? The reality is that sometimes joins are more efficient than subqueries, sometimes vice versa, and sometimes they're optimized the same. Sometimes a single query with 20 joins will perform just fine, and a query with 4 joins will actually perform better if split into steps that fill temp tables that are then joined. Optimization of T-SQL code is such a dynamic and sophisticated process that rules of thumb such as these won't help but will instead mislead you. If you insist on adopting a rule of thumb, make it "Keep an open mind, and be careful about rules of thumb." This should always be true...maybe....

### Use Search Arguments

When optimizing a query and considering whether and how to use indexes, one of the key elements the optimizer evaluates is the predicates in the query filters. However, you need to be aware that in order to rely on index ordering (e.g., perform an index seek or ordered scan), the predicate must be of a particular form. Specifically, except for very few exceptions, you should try to avoid manipulation of the index key columns. In most cases, after you apply manipulation the optimizer can't trust that the outcome of the manipulation preserves index ordering. Sometimes there's not much you can do to avoid manipulation, but sometimes you can. When the optimizer can rely on index ordering for a given predicate, the predicate is said to be a search argument, or SARG for short. Here are a few examples:

* The predicate COALESCE(T1.col1, -1) = COALESCE(T2.col1, -1) isn't a SARG, whereas the predicate T1.col1 = T2.col1 OR (T1.col1 IS NULL and T2.col1 IS NULL) is.
* The predicate col1 - 1 <= @n isn't a SARG, whereas the predicate col1 <= @n + 1 is.
* The predicate LEFT(col1, 3) isn't a SARG, whereas the predicate col1 LIKE 'abc%' is.

As you can see, sometimes simple revisions to your predicates can retain the meaning and allow efficient use of indexing.

### Practices to Use or Avoid

In this two-part series, I discussed several important areas in T-SQL where you often see people implementing bad practices. I provided what I find to be best practices that can help you write maintainable, correct, and efficient code. Some best practices involve understanding the mathematical foundations of the language and require time and effort on your part, whereas others are very simple and straightforward and only require you to be aware of them. The set of best practices I provided isn't exhaustive, but it does include what I find to be particularly important.

# 72. **SQL Injection – good**

<https://www.w3schools.com/sql/sql_injection.asp>

# 73. [SQL Server - stop or break execution of a SQL script](https://stackoverflow.com/questions/659188/sql-server-stop-or-break-execution-of-a-sql-script) -- 30/7/2018

<https://stackoverflow.com/questions/659188/sql-server-stop-or-break-execution-of-a-sql-script?noredirect=1&lq=1>

# [**SQL Server Management Studio Query Stop/Exit [duplicate]**](https://stackoverflow.com/questions/21061965/sql-server-management-studio-query-stop-exit)

<https://stackoverflow.com/questions/21061965/sql-server-management-studio-query-stop-exit>

# 74. **Different Options for Query Results in SQL Server Management Studio**

<https://www.mssqltips.com/sqlservertip/2346/different-options-for-query-results-in-sql-server-management-studio/>

75. difference between temp table and table variable --29/8/18

<https://www.sqlshack.com/when-to-use-temporary-tables-vs-table-variables/>

# **When to Use Temporary Tables vs. Table Variables**

February 21, 2017 by [Aamir Syed](https://www.sqlshack.com/author/aamir-syed/)

It is very beneficial to store data in temporary tables rather than manipulate or work with permanent tables. Let’s say you want full DDL or DML access to a table, but don’t have it. You can use your existing read access to pull the data into a temporary table and make adjustments from there. Or you don’t have permissions to create a table in the existing database, you can create a temporary table that you can manipulate. Finally, you might be in a situation where you need the data to be visible only in the current session.

<https://stackoverflow.com/questions/27894/whats-the-difference-between-a-temp-table-and-table-variable-in-sql-server>

There are a few differences between Temporary Tables (#tmp) and Table Variables (@tmp), although using tempdb isn't one of them, as spelt out in the MSDN link below.

As a rule of thumb, for small to medium volumes of data and simple usage scenarios you should use table variables. (This is an overly broad guideline with of course lots of exceptions - see below and following articles.)

Some points to consider when choosing between them:

* Temporary Tables are real tables so you can do things like CREATE INDEXes, etc. If you have large amounts of data for which accessing by index will be faster then temporary tables are a good option.
* Table variables can have indexes by using PRIMARY KEY or UNIQUE constraints. (If you want a non-unique index just include the primary key column as the last column in the unique constraint. If you don't have a unique column, you can use an identity column.) [SQL 2014 has non-unique indexes too](https://stackoverflow.com/questions/886050/sql-server-creating-an-index-on-a-table-variable/17385085#17385085).
* Table variables don't participate in transactions and SELECTs are implicitly with NOLOCK. The transaction behaviour can be very helpful, for instance if you want to ROLLBACK midway through a procedure then table variables populated during that transaction will still be populated!
* Temp tables might result in stored procedures being recompiled, perhaps often. Table variables will not.
* You can create a temp table using SELECT INTO, which can be quicker to write (good for ad-hoc querying) and may allow you to deal with changing datatypes over time, since you don't need to define your temp table structure upfront.
* You can pass table variables back from functions, enabling you to encapsulate and reuse logic much easier (eg make a function to split a string into a table of values on some arbitrary delimiter).
* Using Table Variables within user-defined functions enables those functions to be used more widely (see CREATE FUNCTION documentation for details). If you're writing a function you should use table variables over temp tables unless there's a compelling need otherwise.
* Both table variables and temp tables are stored in tempdb. But table variables (since 2005) default to the collation of the current database versus temp tables which take the default collation of tempdb ([ref](https://technet.microsoft.com/en-us/library/ms188927.aspx)). This means you should be aware of collation issues if using temp tables and your db collation is different to tempdb's, causing problems if you want to compare data in the temp table with data in your database.
* Global Temp Tables (##tmp) are another type of temp table available to all sessions and users.

Some further reading:

* [Martin Smith's great answer](https://dba.stackexchange.com/a/16386) on dba.stackexchange.com
* MSDN FAQ on difference between the two: <https://support.microsoft.com/en-gb/kb/305977>
* MDSN blog article: <http://blogs.msdn.com/sqlserverstorageengine/archive/2008/03/30/sql-server-table-variable-vs-local-temporary-table.aspx>
* Article: [http://searchsqlserver.techtarget.com/tip/0,289483,sid87\_gci1267047,00.html#](http://searchsqlserver.techtarget.com/tip/0,289483,sid87_gci1267047,00.html)
* Unexpected behaviors and performance implications of temp tables and temp variables: [Paul White on SQLblog.com](http://sqlblog.com/blogs/paul_white/archive/2012/08/15/temporary-tables-in-stored-procedures.aspx)

# 76. Creating a date dimension or calendar table in SQL Server – very good

<https://www.mssqltips.com/sqlservertip/4054/creating-a-date-dimension-or-calendar-table-in-sql-server/>

77. Schema binding -- 31/8/218

# Benefits of SCHEMABINDING in SQL Server

<https://www.mssqltips.com/sqlservertip/4673/benefits-of-schemabinding-in-sql-server/>

Aside from improved performance gained from avoiding [Halloween protection](https://en.wikipedia.org/wiki/Halloween_Problem), what other benefits can I get by using WITH SCHEMABINDING?

You can use WITH SCHEMABINDING on any module to prevent any inadvertent modifications to the objects referenced by the module. So, for example, if you have a function that relies on the data type of a column in a table, you can use WITH SCHEMDABINDING to ensure that the table doesn't change. Without SCHEMABINDING, the following scenario is possible:

Halloween Problem

<https://en.wikipedia.org/wiki/Halloween_Problem>

<https://sqlstudies.com/2014/08/06/schemabinding-what-why/> --good

# [SchemaBinding – What & Why](https://sqlstudies.com/2014/08/06/schemabinding-what-why/)

[24](https://sqlstudies.com/2014/08/06/schemabinding-what-why/#comments) [August 6, 2014](https://sqlstudies.com/2014/08/06/schemabinding-what-why/) by Kenneth Fisher

##### What

When you use the SchemaBinding keyword while creating a [view](http://msdn.microsoft.com/en-us/library/ms187956.aspx) or [function](http://msdn.microsoft.com/en-us/library/ms186755.aspx) you bind the structure of any underlying tables or views. So what does that mean? It means that as long as that schemabound object exists as a schemabound object (ie you don’t remove schemabinding) you are limited in changes that can be made to the tables or views that it refers to.

And here are a couple of other restrictions & factoids.

* You cannot change the collation of a database with schemabound objects.
* You cannot use *SELECT \** in a schemabound view.
* You cannot run sp\_refreshview on a schemabound view. You do get a rather unhelpful error though.
* You *can* make any change to the table that do not affect the structure of the bound columns.
* You can find out if an object is schemabound by looking at the column is\_schema\_bound in sys.sql\_modules or the system function OBJECTPROPERTY(object\_id, ‘is\_schema\_bound’).
* If you reference a view or function in a schemabound view or function then that view or function must also be schemabound.
* Objects that are bound (tables/views) cannot be dropped while a schemabound object references them

##### Why

Schemabinding isn’t a commonly used tool unless you are setting up an [indexed view](http://msdn.microsoft.com/en-us/library/ms191432.aspx) and then it can get lost in the crowd of other required restrictions. It does have uses outside of indexed views however. I could see using it if there is a mission critical view/function that just CAN’T break. By including the SCHEMABINDING clause you protect the view/function from unexpected changes to the tables underneath them. In fact if all of the data access in an application is through views and TVFs I might consider schemabinding all of them. It might be less important in a small shop with only a couple of developers and/or DBAs where everyone knows what changes are being made and what effect they will have. However if you are in a big shop with dozens of applications, may of which use the same databases, you can easily make a change to a table that breaks code in another application that you were completely unaware of.

So in the end SCHEMABINDING isn’t a world changing clause but still one that you should be aware of.