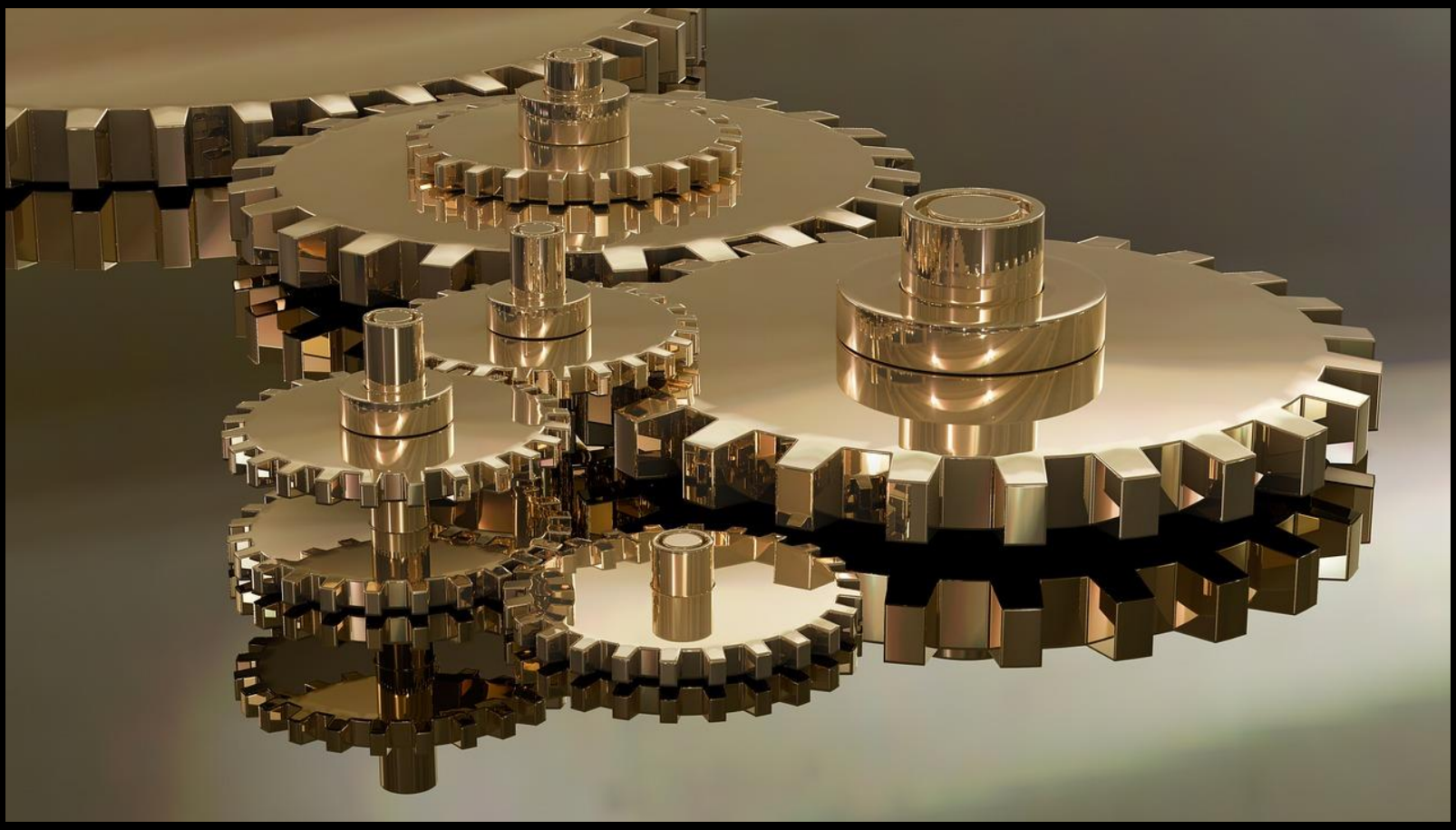
Second Edition

Administering SQL Server®



Get Useful Tips on SQL Server® Administration

Artemakis Artemiou

Administering SQL Server®

(Second Edition)

ARTEMAKIS ARTEMIOU

PUBLISHED BY
Artemakis Artemiou

Publication Date: July 5, 2017.

About the Author

Artemakis Artemiou is a Senior SQL Server and Software Architect, Author, and a former Microsoft Data Platform MVP (2009-2018). He has over 15 years of experience in the IT industry in various roles. Artemakis is the founder of SQLNetHub and TechHowTos.com. Artemakis is the creator of the well-known software tools Snippets Generator, DBA Security Advisor and In-Memory OLTP Simulator. Moreover, he is the author of many eBooks on SQL Server. Artemakis currently serves as the President of the Cyprus .NET User Group (CDNUG) and the International .NET Association Country Leader for Cyprus (INETA). Artemakis's official website can be found at aartemiou.com. You can follow Artemakis on Twitter at https://twitter.com/artemakis.

Some of the online channels where you can find Artemakis are:

- SQLNetHub
- Official Website
- The SQL Server and .NET TV
- TechHowTos.com
- Twitter
- Cyprus .NET User Group

Contact the Author

You can contact the author using the contact methods on www.aartemiou.com and www.sqlnethub.com

To all of those who dare to experiment with new technology,
who constantly try to become better in what they do,
but never lose respect for their fellow human beings in the process.
-A.A.

SAMPLE BOOK CHAPTER

PURCHASE THE FULL EBOOK AT:

https://www.sqlnethub.com/sql-server-ebooks/

Introduction

I started working with SQL Server and .NET (C#) 15 years ago. Since then it has been quite a journey! Each release came - and still comes - with exciting new features enabling us to do more and more! Every time waiting for that new built, in order to start testing it, exploring it, learning it, as soon as it becomes available. The possibilities are endless! The only limit is your creativity!

The massive interaction with the SQL Server community started at about seven years ago. Blogging, organizing user group events, speaking in user group meetings, conferences and other events, open-source projects related to SQL Server, guest articles, discussions on message boards/forums, and much more!

The love for technical writing and knowledge sharing urged me for adding another activity to my interaction with the community that is ***book authoring***. In order to be able to write, you first need to acquire and comprehend the specific technical knowledge. You need to explore, to experiment, to test. You need to test the limits of each new technology or feature in order to be able to fully understand its nature and capabilities.

SQL Server is a powerful data platform that includes several data management and analysis technologies that allow you to do just about anything. Since 2002 I have been exploring SQL Server in many areas. I have been deep diving into various topics of SQL server having to do mainly with: ***administration***, ***development/data access***, ***performance tuning, In-Memory OLTP***. These are the four areas of SQL Server I mostly focus on and on which I base, but not limit, my interaction with SQL Server and acquisition of knowledge. To this end, I have decided to start writing eBooks mainly, but not limited to, on the above areas.

Who Should Read This Book?

This book is for database administrators and architects who monitor and administer SQL Server instances in order to keep them operating to the highest possible level of stability and performance. The book suggests several techniques that can be used for ensuring a healthy SQL Server instance. However, the book is not intended to be a step-by-step comprehensive guide. Additionally, it assumes at least intermediate-level experience with SQL Server administration.

Supported SQL Server Versions

Each article in this book has in the end an "***Applies to:***" note indicating the SQL Server versions against which the provided T-SQL scripts can be executed. There are also articles providing different T-SQL scripts for SQL Server 2000, SQL Server 2005 or later. Every time I write an article, I try to write code that can be executed in all versions of SQL Server unless stated otherwise. Even though SQL Server 2000 and 2005 are not officially supported anymore, I am quite sure that still there are many of you out there who still use these versions of SQL Server for various reasons (i.e. application compatibility issues, etc.). My

advice is to consider upgrading to a newer version of SQL Server the soonest possible, in order to be able to use the benefits of the latest SQL Server releases as well as being able to get official support and updates.

How Is This Book Organized?

This book is organized as follows. Chapter 1 discusses basic SQL Server maintenance tasks such as disk usage monitoring, history cleanup, backup-related topics, and more. Chapter 2 discusses security and compliance topics such as Policy-Based Management and encryption. Chapter 3 talks about different integration topics such as linked servers, use of proxies on SQL Server Agent job steps and Unicode support. Chapter 4 discusses special SQL Server topics like the Windows Internal Database (SSEE) and dynamic T-SQL generation. Chapter 5 discusses error handling and ways to overcome errors you might encounter due to external factors like permission issues, invalid user input, etc.

Feedback

I am a proud member of the worldwide SQL Server community. Feedback from you, my fellow community members, is more than welcome. Please feel free to visit the following link and provide your feedback:

http://www.sqlebooks.com/feedback/

The survey is short. It will only take 5 minutes of your valuable time.

Acknowledgments

Writing a book is not an easy thing. It doesn't really matter the length of the book. Even if you just write a few pages, it takes a considerable amount of time and energy. In the case where you are not a professional technical writer but just a technical community guy like me, this amount of time is valuable "free" time after work, time from your family and beloved ones. You may use this time because you are just passionate in what you do. However, this is not enough, at least to me. Without the support of your family it just doesn't feel right. During this process, I had all the support I needed. I am grateful for all the support my family gave me during writing this book and for all their support and love in everything I do. Without their support and encouragement none of this would have been possible. This book is dedicated to them with all my love.

- Artemakis

Stay in Touch

Feel free to connect! You can find me on the following online channels:

- Official Website: https://www.aartemiou.com
- SQLEBooks.com: https://www.sqlebooks.com
- SQLArtBits: https://www.sqlartbits.com
- Twitter: https://twitter.com/artemakis
- The SQL Server and .NET Hub: http://www.sqlnethub.com
- The SQL Server and .NET TV: http://www.youtube.com/user/sqlserverdotnetblog
- Cyprus .NET User Group: http://www.cdnug.net
- CodePlex: http://www.codeplex.com/site/users/view/ArtSQL

CHAPTER 4

Special Topics

IN THIS CHAPTER:

- Windows Internal Database (SSEE)
- Dynamically Generating T-SQL Statements
- Massively Detaching and Re-attaching Databases in SQL Server
- Installing 32-bit SQL Server 2005 Reporting Services on a 64-bit machine/Windows OS
- Compatibility Levels Supported by Different SQL Server Versions
- Searching for Keywords in SQL Server Jobs
- Accessing Reporting Services Using a Fully Qualified Domain Name
- Useful SQL Server Knowledge

Have you ever wondered what Windows Internal Database is, what does exactly do and how you can connect to it? Have you ever wondered how you can massively manage databases and other SQL Server objects with just a single T-SQL script?

This chapter is dedicated to different special topics having to do with various administration-related tasks in SQL Server. Among other you will learn more about the Windows Internal Database (SSEE), how to dynamically generate T-SQL statements (i.e. how to massively detach and re-attach SQL Server databases) and how you can install 32-bit SQL Server Reporting Services on 64-bit machine.

Also, we will talk about SQL Server compatibility levels, SQL Server Reporting Services (SSRS) and use of fully qualified domain names.

Last but not least, in the last article of this chapter you will find a large set of different SQL-Server related tips focusing on the following areas:

- Solutions to common issues
- Basic string functions
- Performance-related tips
- Maintenance
- Miscellaneous

Windows Internal Database (SSEE)

The Windows Internal Database is a special variant of Microsoft SQL Server Express 2005-2012. It is included with Windows Server 2008/R2/2012 and later, Windows Server Update Services (WSUS), Windows SharePoint Services (WSS) and other Windows Server tools. It is also referenced as SQL Server Embedded Edition (SSEE).

The Windows Internal Database is used by Active Directory, WSS, WSUS and other Windows Server services as their data storage. Some of its main characteristics are:

- It cannot be used as a regular SQL Server instance as it is intended to be only used by Windows Services/Programs.
- It cannot be removed by using the "Add or Remove Programs" tool because it does not appear in the list of currently installed programs (more info here). Note that it is not recommended to uninstall SSEE as it might affect the operation of Windows Services that use it.
- It only supports Windows Authentication.
- You can only connect to the instance using Named Pipes.

In some cases where you might need to access this special instance of SQL Server (i.e. for reducing the transaction log size of a database) you must use the Named Pipes protocol for doing so. Named Pipes can be enabled from Network Configuration in SQL Server Configuration Manager.

First of all you will need to install SQL Server 2005 Express Edition Management Studio or later (link) and use it locally on the server. Then you will need to use the following connection properties:

- Server Type: **Database Engine**
- Server Name (Windows Server 2003-2008): **\\.\pipe\MSSQL$MICROSOFT##SSEE\sql\query**
- Server Name (Windows Server 2012): **\\.\pipe\MICROSOFT##WID\tsql\query**
- Authentication: **Windows Authentication**

***Note**: You have to be careful when accessing the Windows Internal Database as many Windows Services depend on it.

➔ ***Applies to: Windows Server 2003 / SQL Server 2005 or later.***

Dynamically Generating T-SQL Statements

In previous articles we talked about the undocumented stored procedures "sp_msforeachdb" and "sp_msforeachtable" which allow you to massively perform changes on all databases and tables within a SQL Server instance. However, massively performing changes on database objects does not give you much control over the process and many times, even the experienced DBA or Developer might encounter problems if for example a database object is mistakenly included in the above process.

A method that allows you to still massively manage database objects but also gives you more control is to dynamically generate T-SQL scripts. By using some of the catalog views in SQL Server 2005 or later you can easily do that.

The idea is to use a basic SELECT statement and then dynamically build the T-SQL expression that eventually will generate the desired T-SQL code.

Below you can find two examples (before running the code, right-click in the query window, then select "Results To" and finally "Results to Text"):

Example 1: See the physical files used for each database

```
SELECT
'USE '+ name +';'+' SELECT name,physical_name FROM SYS.DATABASE_FILES'
FROM SYS.databases;
GO
```

Listing 4.1: Generating a set of statements for retrieving physical files info for all DBs.

Example 2: See all the views for each database

```
SELECT
'USE '+ s.name +';'+' SELECT * FROM sys.views' FROM SYS.databases s;
GO
```

Listing 4.2: Generating a set of statements for viewing all view for all DBs.

When using this technique along with the information retrieved from the catalog views, the possibilities are endless.

Yes, in this case you still need to manually execute each generated T-SQL statement but the good thing is that the statements are being generated dynamically and by manually executing them, you have more control over the process.

*** Caution:** Whenever you massively modify database objects you need to be very careful. Always backup your data.

➔ ***Applies to: SQL Server 2005 or later.***

Massively Detaching and Re-attaching Databases in SQL Server

There are cases where you might need to massively detach and re-attach all the databases in a SQL Server instance for any reason. In the case where you have 2-3 user databases on the instance, it might not be an issue to detach and re-attach the databases manually. However, imagine having over 100 databases. It would be not the easiest thing to go and detach/re-attach each database one-by-one (keep in mind that you have to specify the data/log file names for each database when you are attaching it back to the instance).

In order to simplify things, I have developed a simple procedure that undertakes generating the necessary detach/attach scripts when you want to massively perform these actions on a SQL Server 2005 instance or later.

*** Caution:** This procedure is presented in this article only for showing a different way of doing things. Always take backups of your data prior to any changes or actions.

The procedure for generating the scripts is the following:

1. Run the "**Attach DDL Statements Generation Script**"
2. Run the "**Detach DDL Statements Generation Script"**

If you want to proceed and detach all user databases you can then execute the set of DDL statements generated by the "Detach DDL Statements Generation Script".

If you want to attach the databases on the same instance or any other SQL Server instance but on the same server, you can then execute the set of DDL statements generated by the "Attach DDL Statements Generation Script". On a different case, you will have to update the physical location paths in the scripts.

You can view the scripts below:

Attach DDL Statements Generation Script (on next page):

```
--
--Attach DDL Statements Generation Scrip
--
print '--'
print '--Script for Attaching all DBs in a SQL Server Instance'
print '--'
print ''
SET NOCOUNT ON

DECLARE @dbname nvarchar(128)

DECLARE DBList_cursor CURSOR FOR
select [name] from master.sys.databases where database_id > 4;

OPEN DBList_cursor

 FETCH NEXT FROM DBList_cursor
 INTO @dbname;

 WHILE @@FETCH_STATUS = 0
 BEGIN

  declare @attach_TSQL_script varchar(max);
  set @attach_TSQL_script='';
  set @attach_TSQL_script=@attach_TSQL_script+'CREATE DATABASE ' + @dbname +' ON ' ;

  declare @tsql varchar(max),@filename varchar(max);
  set @tsql='DECLARE DBFiles_cursor CURSOR FOR select [filename] from '+ @dbname +
'.sys.sysfiles';
  execute (@tsql);

  PRINT '--'+@dbname;

  OPEN DBFiles_cursor
  FETCH NEXT FROM DBFiles_cursor INTO @filename;

  WHILE @@FETCH_STATUS = 0
  BEGIN
   set @attach_TSQL_script=@attach_TSQL_script+' (FILENAME = '''+ @filename +'''),';
   FETCH NEXT FROM DBFiles_cursor INTO @filename
  END

 set @attach_TSQL_script=SUBSTRING(@attach_TSQL_script,0,len(@attach_TSQL_script));
 set @attach_TSQL_script=@attach_TSQL_script+' FOR ATTACH;';

 PRINT @attach_TSQL_script;
 PRINT '';

CLOSE DBFiles_cursor;
DEALLOCATE DBFiles_cursor;

FETCH NEXT FROM DBList_cursor
INTO @dbname;

END

CLOSE DBList_cursor;
DEALLOCATE DBList_cursor;
```

Listing 4.3: Generating attach DDL scripts for all DBs in a SQL Server Instance.

Detach DDL Statements Generation Scrip

```
--
--Detach DDL Statements Generation Scrip
--

PRINT '--';
PRINT '--Script for Detaching all DBs in a SQL Server Instance';
PRINT '--';
PRINT '';

SET nocount ON;

DECLARE @dbname nvarchar(128);
DECLARE dblist_cursor CURSOR FOR
  SELECT [name]
  FROM   master.sys.databases
  WHERE  database_id > 4;

OPEN dblist_cursor
FETCH next FROM dblist_cursor INTO @dbname;

WHILE @@FETCH_STATUS = 0
  BEGIN
    print 'EXEC sp_detach_db ''' + @dbname + ''', ''true'';';
    FETCH next FROM dblist_cursor INTO @dbname;
  END

CLOSE dblist_cursor;
DEALLOCATE dblist_cursor;
```

Listing 4.4: Generating detach DDL scripts for all DBs in a SQL Server Instance.

Applies to: SQL Server 2005 or later.

▪ Installing 32-bit SQL Server 2005 Reporting Services on a 64-bit machine/Windows OS

Many fellow community members ask whether it is possible to install 32-bit SQL Server 2005 Reporting Services on a 64-bit Computer/Windows OS while keeping the 64-bit version of the rest of SQL Server features. The answer is yes.

Consider the following real-life scenario that happened to me some years ago.

I had installed SQL Server 2005 on a 64-bit Windows Server 2003 OS. This was done within the context of setting up a test environment for an ASP .NET application. As the OS was a 64-bit architecture, I had installed the 64-bit version of SQL Server as well.

Though, the ASP .NET application was compiled on .NET Framework 1.1 and IIS 6.0 was set up to

use this version of the .NET Framework. To this end, when the installation of the reporting services started, I got the message that it was not possible to install the 64-bit version of the SQL Server 2005 Reporting Services because of using .NET Framework 1.1 for IIS 6.0.

Though, SQL Server did not disappoint me. With some simple steps, it was very easy to install the 32-bit version of the Reporting Services while keeping installed the 64-bit version of the rest of SQL Server features (i.e. Database Engine, Analysis Services, etc.).

The following MSDN Link provides excellent instructions on how to perform this:
http://msdn.microsoft.com/en-us/library/ms143293(SQL.90).aspx

*** Note:** The procedure is almost identical for **SQL Server 2008** with the difference that when you are going to install Reporting Services, before starting the installation process, from within SQL Server 2008 Installation Center, click on "**Options**" and select the "**x86" Processor Type**", then click on "**Installation**" and after selecting "**New SQL Server stand-alone installation or add features to an existing installation**" install only Reporting Services.

➔ ***Applies to: SQL Server 2005/2008/2008R2 or later.***

▪ Compatibility Levels Supported by Different SQL Server Versions

Software is evolving and certainly SQL Server is not an exception to the rule. Every few years we need to upgrade our databases in order to run on a newer version of SQL Server and take advantage of significant new features that enhance the operations of our organization.

However, upgrading a database to a newer compatibility level of any DBMS involves evaluating not just the database, but also how the application supported by that database will behave.

The easiest thing for many people is to move the database to the new version of SQL Server but make use of the backwards compatibility support of the Database Engine. This is however not a recommended approach, because this way you cannot utilize all the features provided by the Database Engine of that new version of SQL Server.

A recommended high-level practice for migrating a database to a newer version of SQL Server is:

- Analyze your database with SQL Server Upgrade Advisor (for each SQL Server version there is the corresponding version of the Upgrade Advisor) in order to find any incompatibilities (i.e. the usage of deprecated features, etc.).
- Resolve any compatibility issues that might be reported by the tool.
- Move the database to the desired version of SQL Server on a **Test Environment**.
- Test your database and the supported application to check if everything works well (this step involves extensive testing by all involved parties).

- Resolve any issues that might be raised.
- Only if you are sure that everything works well after the testing process and any issues have been resolved, as well as you acquired all the necessary approvals, then you can consider proceeding to the actual migration.

Of course, it goes without saying that you always need to take backups of your databases, not only during their operational cycle but also always before making any changes to their configuration.

Now, you might encounter certain cases where you deal with a "legacy" application and it might be nearly impossible to upgrade its supporting database to a newer version, but at the same time you want to move it to a newer version of SQL Server. In such cases you can make use of the backwards compatibility option the Database Engine of SQL Server provides. However, each version of SQL Server supports up to a specific compatibility level. For example if you have a SQL Server 2000 database you cannot migrate it to a SQL Server 2012 or 2014 instance because there is no backwards compatibility support for such an old version of SQL Server. You can however migrate it to a SQL Server 2008 R2 instance.

Below you can find the supported compatibility modes for SQL Server versions 2008/R2, 2012, 2014 and 2016:

SQL Server 2008

80: SQL Server 2000

90: SQL Server 2005

100: SQL Server 2008

SQL Server 2008 R2

80: SQL Server 2000

90: SQL Server 2005

100: SQL Server 2008 (and 2008 R2 - it is the same compatibility level)

SQL Server 2012

90: SQL Server 2005

100: SQL Server 2008 and SQL Server 2008 R2

110: SQL Server 2012

SQL Server 2014

100: SQL Server 2008 and SQL Server 2008 R2

110: SQL Server 2012

120: SQL Server 2014

SQL Server 2016

100: SQL Server 2008/R2

110: SQL Server 2012
120: SQL Server 2014
130: SQL Server 2016

***Tip:** There is a free online SQL Server service developed by SQLArtBits called "SQL Server Backward Compatibility Checker" which allows you to get backward compatibility information for the selected SQL Server version. You can access this free service here.

▪ Searching for Keywords in SQL Server Jobs

This article is actually a T-SQL tip that can become quite handy. It tackles the issue of finding which SQL Server Agent jobs are affected due to schema changes (i.e. table rename, etc.).

Consider the following example where you have three tables:
- table1
- table2
- table3

Also consider that you have a SQL Server Agent jobs that reference the above tables.

Then, for any reason, you want to rename '**table2**' to '**table2New**'

What happens to the SQL Server Agent job the next time it runs? Of course it fails.

When you rename an object in SQL Server using the sp_rename stored procedure, the Database Engine warns you with the below message:

Caution: Changing any part of an object name could break scripts and stored procedures.

In order not to get into problems with such actions, especially in the case of large databases, prior to renaming, you must first research if there are any functions, stored procedures, jobs, as well as other database objects that might be affected.

The below script returns **all jobs** that include one or more steps that reference a given string pattern (this can be a table name, stored procedure name, function name, etc.). In this case we search for **'table2'**:

```
DECLARE @pattern NVARCHAR(MAX);
SET @pattern = 'table2';

SELECT  j.[job_id] ,
        j.[name] ,
        j.[enabled] ,
        j.[description] ,
        s.command ,
        j.[date_created] ,
        j.[date_modified]
FROM    msdb.dbo.sysjobs j
        INNER JOIN msdb.dbo.[sysjobsteps] s ON j.job_id = s.job_id
WHERE   s.command LIKE '%' + @pattern + '%';
```

Listing 4.5: Search for Keywords in SQL Server Jobs.

***Note:** Prior to renaming a database object, you need to check if it is referenced by any other object such as functions, views, stored procedures, jobs, etc.

➔ *Applies to: SQL Server 2005 or later*

▪ Accessing Reporting Services Using a Fully Qualified Domain Name

If you installed SQL Server Reporting Services (SSRS) on a server in a domain and you use a domain user to start the service and did not perform any further configuration, then you most probably can only access the Report Manager using an IP and not the Fully Qualified Domain Name (FQDN) of the server (if an SPN is not set).

If you try to use the Fully Qualified Domain Name (FQDN) to access reporting services then you will most probably be prompted for username password several times ending with an empty page.

In order to be able to access Reporting Services using a Fully Qualified Domain Name you will need to perform the following actions:

1. Register a Service Principal Name (SPN) for the Domain User Account that Runs SSRS

Consider the following example:

sample computer name: **reportsrv01**
sample domain: **example.com**

sample domain account: **example\ssrsuser**

Then on the Domain Controller Server in a command prompt with elevated rights, you can Run as Administrator:

example 1: If SSRS are on port 80 (no need to specify 80 as it is the default http port):
Setspn -s http/reportsrv01.example.com example\ssrsuser

example 2: If SSRS are on any other port (i.e. 2430):
Setspn -s http/reportsrv01.example.com:2430 example\ssrsuser

2. Edit the RsReportServer.config File
On the Reporting Services server, in the virtual directory of SSRS, edit the "**RsReportServer.config**" file and locate the **authenticationtypes** section.

Then add **/rswindowsnegotiate** as the first entry in the **authenticationtypes** section.

The above step will actually enable NTLM.

For more information please visit this MSDN article.

➔ *Applies to: SQL Server 2008 or later*

▪ Useful SQL Server Knowledge

In this article you can find different tips and T-SQL scripts organized in the following categories:

A. Solutions to common issues
B. Basic string functions
C. Performance-related tips
D. Maintenance
E. Miscellaneous

A: Solutions to common issues

1. Resolving the "Divide by zero" error (by example)

```
DECLARE @denominator INT
SET @denominator = 0
SELECT 1 / ISNULL(NULLIF(@denominator, 0), 1);
GO
```

Listing 4.6: Resolving the "Divide by zero" error.

2. Handling NULL and empty values

```
----Step 1: Create the IsEmpty user-defined function
CREATE FUNCTION IsEmpty
(
@input AS VARCHAR(250),
@newValue VARCHAR(250)
)
RETURNS VARCHAR(250)
AS
BEGIN
-- First handle the case where the input value is a NULL
DECLARE @inputFiltered AS VARCHAR(250);
SET @inputFiltered = ISNULL(@input, '');


-- The main logic goes here
RETURN (CASE RTRIM(LTRIM(@inputFiltered)) WHEN '' THEN RTRIM(LTRIM(@newValue))
ELSE RTRIM(LTRIM(@inputFiltered)) END);


END
GO

----Step 2: Usage
SELECT dbo.IsEmpty(@column_to_check, @new_value);
GO
```

Listing 4.7: Handling NULL and empty values.

3. Handling the error "A transport-level error has occurred when sending the request to the server"

If the problem occurs in a SSMS Query Window, just try again to run your set of statements and a new connection will be re-established.

4. Handling the error "The conversion of a char data type to a datetime data type resulted in an out-of-range datetime value"

Change the default language to "us_english" for the given SQL Server login:

```
USE [master];
GO
ALTER LOGIN "LOGIN_NAME" WITH DEFAULT_LANGUAGE = us_english;
GO
```

Listing 4.8: Handling the out-of-range datetime value error.

Best practice: Always use the ISO date format in your data applications/T-SQL scripts: **YYYY-MM-DD**

5. Handling the error "The multi-part identifier ... could not be bound"

Be careful with the use of subqueries and table aliases. Don't forget to reference the correct

table aliases in your T-SQL code.

Also, keep in mind that subqueries can only provide their results to their outer queries and not references to the subqueries' tables.

6. Handling the error "String or binary data would be truncated"
Either use an adequate size for the table columns in which the data is inserted or cast the data by removing redundant characters. I suggest the first approach.

7. Handling the error "Error converting data type varchar to float"

```
----Step 1: Create the Varchar2Float user-defined function
CREATE FUNCTION [dbo].[Varchar2Float]( @inputString VARCHAR(50) )
RETURNS FLOAT
AS
BEGIN
--Prepare the string for casting/conversion
SET @inputString = REPLACE(@inputString, '.', '');
SET @inputString = REPLACE(@inputString, ',', '.');

--Perform the conversion and return the result
RETURN CAST(@inputString AS FLOAT);
END
GO

----Step 2: Usage
SELECT dbo.Varchar2Float(@value);
GO
```

Listing 4.9: Handling the error "Error converting data type varchar to float".

8. Handling the error "Database [Database_Name] cannot be upgraded because it is read-only or has read-only files"
Make sure that the service account on which the SQL Server instance database engine is running has full access to the database files. Also, if you are working in Windows Vista or later, try running SSMS as Administrator.

B. Basic string functions

1. Returns @length characters from @expression starting from @start_index

```
SELECT SUBSTRING(@expression, @start_index, @length);
GO
```

Listing 4.10: Using the SUBSTRING string function.

2. Finds the given @pattern in the @string and replaces it with the @replacement_string

```
SELECT REPLACE(@string, @pattern, @replacement_string);
GO
```

Listing 4.11: Using the REPLACE string function.

3. Returns the size of @string in terms of number of characters

```
SELECT LEN(@string);
GO
```

Listing 4.12: Getting the length of a string.

4. Returns the first @num_chars characters of the @string counting from the left

```
SELECT LEFT(@string, @num_chars);
GO
```

Listing 4.13: Using the LEFT string function.

5: Returns the first @num_chars characters of the @string counting from the right

```
SELECT RIGHT(@string, @num_chars);
GO
```

Listing 4.14: Using the RIGHT string function.

6: Removes the leading blank spaces

```
SELECT LTRIM(@expression);
GO
```

Listing 4.15: Removing leading blank spaces in a string.

7: Removes the trailing blank spaces

```
SELECT RTRIM(@expression);
GO
```

Listing 4.16: Removing trailing blank spaces in a string.

C. Performance-related tips

1. Avoiding locking when reading data (however, dirty reads are allowed)

```
SELECT [columnName]
FROM [tableName] WITH (NOLOCK);
GO
```

Listing 4.17: Using the NOLOCK table hint.

2. Rebuilding indexes in SQL Server 2005 or later

```
USE [DATABASE_NAME];
ALTER INDEX [INDEX_NAME] ON [SCHEMA.TABLE]
REBUILD WITH (FILLFACTOR=[FILL_FACTOR_VALUE_BETWEEN_0_100], ONLINE=[ON|OFF]);
GO
```

Listing 4.18: Rebuilding a specific index with using parameters.

```
USE [DATABASE_NAME];
ALTER INDEX ALL ON [SCHEMA.TABLE]
REBUILD WITH (FILLFACTOR=[FILL_FACTOR_VALUE_BETWEEN_0_100], ONLINE=[ON|OFF]);
GO
```

Listing 4.19: Rebuilding all indexes in a table with using parameters.

3. Rebuilding all the indexes in a database

```
USE [DATABASE_NAME];
GO
EXEC sp_MSforeachtable @command1="print '?'", @command2="ALTER INDEX ALL ON ?
REBUILD WITH (ONLINE=ON)";
GO
```

Listing 4.20: Rebuilding all indexes online with keeping the default fill factor for each index.

```
USE [DATABASE_NAME];
GO
EXEC sp_MSforeachtable @command1="print '?'", @command2="ALTER INDEX ALL ON ?
REBUILD WITH (ONLINE=OFF)";
GO
```

Listing 4.21: Rebuilding all indexes offline with keeping the default fill factor for each index.

```
USE [DATABASE_NAME];
GO
EXEC sp_MSforeachtable @command1="print '?'", @command2="ALTER INDEX ALL ON ?
REBUILD WITH (FILLFACTOR=[FILL_FACTOR_PERC],ONLINE=ON)";
GO
```

Listing 4.22: Rebuilding all indexes online with specifying the fill factor.

```
USE [DATABASE_NAME];
GO
EXEC sp_MSforeachtable @command1="print '?'", @command2="ALTER INDEX ALL ON ?
REBUILD WITH (FILLFACTOR=[FILL_FACTOR_PERC],ONLINE=OFF)";
GO
```

Listing 4.23: Rebuilding all indexes offline with specifying the fill factor.

4. Updating database tables without causing blocking

```
UPDATE [TABLE_NAME] WITH (READPAST)
SET ...
WHERE ...
GO
```

Listing 4.24: Updating tables with using table hints.

D. Maintenance

1. Shrinking an entire database

```
DBCC SHRINKDATABASE([DBName],[PercentageOfFreeSpace]);
GO
```

Listing 4.25: Shrinking a database.

2. Truncating an entire database

```
DBCC SHRINKDATABASE([DBName],TRUNCATEONLY);
GO
```

Listing 4.26: Releasing back to the OS the occupied space at the end of the DB files.

3. Shrinking a data/log file

```
USE [DBName];
GO
DBCC SHRINKFILE ([Data_Log_LogicalName],[TargetMBSize]);
GO
```

Listing 4.27: Shrinking a data or log file with specific target in MB.

4. Release free space from the end of the file back to OS

```
USE [DBName];
GO
DBCC SHRINKFILE ([Data_Log_LogicalName],TRUNCATEONLY);
GO
```

Listing 4.28: Releasing back to the OS the occupied space at the end of the data or log file.

5. Renaming a Windows login

```
ALTER LOGIN "[Domain or Server Name]\[Windows Username]"
WITH NAME="[New Domain or New Server Name]\[Windows Username]";
GO
```

Listing 4.29: Renaming a Windows login.

6. Renaming a SQL Server login

```
ALTER LOGIN "[SQL Server Login Name]"
WITH NAME="[New SQL Server Login Name]";
GO
```

Listing 4.30: Renaming a SQL Server login.

7. Creating Logins for orphaned SQL Server users

```
USE [DBName];
GO
EXEC sp_change_users_login 'Auto_Fix', '[UserName]', NULL, '[Password]';
GO
```

Listing 4.31: Creating Logins for orphaned SQL Server users.

8. Changing the Database Owner in a SQL Server Database (SQL Login)

```
USE [DBName];
GO
EXEC sp_changedbowner '[SQL_Login_Name]';
GO
```

Listing 4.32: Changing the Database Owner in a SQL Server Database (SQL Login).

9. Changing the Database Owner in a SQL Server Database (Windows Login)

```
USE [DBName];
GO
EXEC sp_changedbowner '[DomainName\UserName]';
GO
```

Listing 4.33: Changing the Database Owner in a SQL Server Database (Windows Login).

10. Backing up a Database in a Network Folder (creating the destination media)

```
USE [master];
GO
EXEC sp_addumpdevice 'disk',
'NetworkDeviceName','\\serverName\backupFolder\BackupFileName.bak';
GO
```

Listing 4.34: Creating the destination media for backing up a database in a network folder.

Now you can use the above "disk" device to backup databases onto the network location you specified.

E. Miscellaneous

1. Gets basic information on the current SQL Server instance

```
SELECT
SERVERPROPERTY('ProductVersion') AS ProductVersion,
SERVERPROPERTY ('ProductLevel') AS ProductLevel,
SERVERPROPERTY ('Edition') AS Edition,
SERVERPROPERTY('MachineName') AS ServerName,
SERVERPROPERTY('ServerName') AS Server_and_Instance_Names;
GO
```

Listing 4.35: Retrieving SQL Server instance-related information.

2. Gets basic table index information

```
EXEC sp_helpindex 'schema.table_name';
GO
```

Listing 4.36: Retrieving table index information.

Applies to: SQL Server 2000 (partially), 2005 or later (fully).

▪ Summary

In this chapter, you learned what the Windows Internal Database (SSEE) is and how you can access it. Additionally, you learned how to dynamically generate T-SQL statements as well as how you can install 32-bit SQL Server Reporting Services on 64-bit machine. Furthermore, you learned various T-SQL tips having to do with different SQL Server topics such as: solutions to common issues, basic string functions, performance tuning and maintenance. The next chapter discusses how you can troubleshoot specific errors you might get due to network problems, permissions, missing service packs/patches etc.

SAMPLE BOOK CHAPTER

PURCHASE THE FULL EBOOK AT:

https://www.sqlnethub.com/sql-server-ebooks/