SSMS for Developers

Eric Selje, Geeks and Gurus, Inc.
Voice: 608/213-9567
Twitter: EricSelje
Email: eselje@geeksandgurus.com

You're a database application developer who's using SQL Server now, but you're no DBA. Sure you use SQL Server Management Studio (SSMS), and maybe you're even pretty good at designing tables and executing queries. But there's a whole lot more functionality in this seemingly simple program than you may realize. In this session we'll share features in SSMS that you may have overlooked but will make your life as a developer a lot easier.

Introduction

A lot of us database developers end up becoming “accidental” DBAs. We didn’t really plan this course of action. Sometimes it happens organically, our application continues to grow until we reach the point where we need to migrate it. Other times we were thrust into an existing project that already had all the data in a back-end database.

For .Net, Access, and Visual FoxPro developers who have grown comfortable in the Microsoft environment, SQL Server is the popular choice for the back-end database, especially since the free SQL Server Express version was released a few years ago. Its limitation of 10 GB per database is usually ample for our applications.

There are other options available to us, including the intriguing SQLAnywhere from Sybase, the open source MySQL and PostgresSQL, and of course the big boys Informix and Oracle. Some of the concepts presented in this paper may well be applicable to those database engines as well, but this guide is focused on SQL Server Management Studio (SSMS). It’s applicable to all versions of SQL Server including the Express version up through SQL Server 2012. This whitepaper is not a comprehensive how-to guide for SQL Server, or even how to get started with SQL Server. We’re going to assume you’ve already got SQL Server installed either locally or on a server (including Azure) and already have some databases installed. If you need a starting reference, Microsoft Press offers the free eBook “Introducing SQL Server 2012” at their site (http://www.microsoft.com/sqlserver/en/us/default.aspx). There are also sample databases there including the ubiquitous and oft-cited in samples “Northwind” database and the newer and more complex “AdventureWorks” database, which we’ll be using in this paper.

This document contains tips and tricks for database developers who are not full time DBAs but are now using SQL Server Management Studio regularly and want to go beyond the basics and explore some of the power of this seemingly-simple tool. These are tips that I’ve jotted down as I’ve used it as well as many that I’ve collected from other users on blogs, in user groups, and even while presenting this session at SQL Saturdays in my area.

So here then I present to you my Top Tips for the Accidental DBA.

Tip # 1: Get Organized Solutions and Projects

Using Solutions in SMSS is exactly like using Solutions in Visual Studio because SSMS is actually built using the same IDE as Visual Studio.

Surprisingly many users of SQL Server Management Studio, including full time DBAs, don't take advantage of Solutions and Projects in SSMS. Instead they rely on the filesystem to organize their odd collection of scripts that remain outside of the database proper. If you don't believe me just watch any of the training videos available on the SQL PASS (Professional Association for SQL Server) website and you'll see plenty of opportunities for the presenters to use Solutions to organize their scripts, and they don't do it.

Part of the problem may be that the Solution Explorer is not visible by default in SSMS, so let's start by making that visible by choosing View | Solution Explorer.

What Do Solutions Buy You?

Solutions offer you a way to organize all of the little snippets of SQL that you don't keep in your database, such as tests that you use to call your database's stored procedures (sp's) or utility scripts for wiping out or adding in test data.

It's important to note that Solutions are stored in the filesystem and not in the database, so if you want to share them with your codevelopers you need to save them in a shared space. However because they're stored locally, you can add them to a Source Control provider unlike everything that's in the database. [There are 3rd Party Tools to add your stored procedures, etc., into Source Control but that's beyond the scope of this paper].

To create a new Solution in SSMS, you naturally select File | New Project from the menu. On that dialog (figure 2) you'll have the option to create a new solution or add the new project to an existing Solution. You can also hook it up to Source Control.

A solution can contain many projects, and each project contains Connections (to a database), Queries (T-SQL scripts), and "Miscellaneous" (whatever you want).

Because you can only have one solution open at a time, and a solution can contain multiple projects, what works for me is to have one solution called "All Clients," and then have a separate project for each client. That way they're all available to me without opening and closing multiple solutions.

Within each Project's Connections, I include connections to point to each client's Live Production server, their Test server, and my development database server. I can also set a default database for each connection. This saves me a ton of time over trying to find the server in my usual "Connect" dialog's long list of servers (figure 3).

What saves even more time however is that by right+clicking on the connection and choosing New Query, you're all set up to run the query on the right server/database without having to mess around with connecting to a server and setting the default database.

Adding queries to the project can be done one of two ways. If you simply right+click on the Queries folder and select "New Query" you'll get a blank script tab where you can start typing away. Once you're happy with your script you'll have to manually rename it to something meaningful, but be sure to keep the .SQL extension or SSMS will move it from Queries to Miscellaneous.

The other way to add a query to your project is to right+click on the Project itself and choose Add | New Item… This will bring up a dialog that is chock full of Query Templates that you can use to do routine tasks, such as Database Backup scripts. This is a great way to get a head start on any T-SQL script, because who can ever remember all that syntax?

The Miscellaneous group under the project holds any file that doesn't have a .SQL extension. That could be notes to yourself, task lists, Analysis Services Queries (if that's your thing, but way beyond the scope), etc. Don't put anything sensitive in here, as these are just files stored on your drive and aren't encrypted or protected in any way.

Solutions are an excellent organizational tool. Use them.

Tip # 2: Get Quickly to the Bottom

Developers, and consultants especially, are often called in to figure out a database they've never seen before. Time is always of the essence, so these tips will help you quickly discover what's going on inside the database.

Filter, Don't Search

Nothing will slow you down quite like scrolling through a long list of similarly named objects in the Object Explorer, trying to find the one thing that you're looking for. Instead, right+click on the category you're looking for and click "Filter Settings". You can then filter out the list not just by the name, but most usefully by the creation date. So you can show, for example, all the views created in the last month.

Search, Don't Filter

There are times when you might actually want a list of the views modified in the last 30 days. In that case you need the results in the Output window, so you're going to need to query the metadata of the database. For example, to get that aforementioned list, you can issue this command in the Query Window:

```
SELECT OBJECT_NAME(object_id), * FROM sys.views WHERE views.modify_date > GETDATE()-365
```

You can now export that output into any number of formats (more on that later).

Bonus Tip: Hey, that's a handy query! I think I'll save that one in my project.

Here's another handy search, which will list all of the objects in the database that have the string 'txtCounty', a field that I'm interested in because my application uses it:

```
SELECT OBJECT_NAME(object_id), * FROM sys.sql_modules WHERE definition LIKE '%txtCounty%'
```

What other metadata can you query? The sys schema can give up just about all the information about the database, including tables, columns, views, stored procedures, indexes, triggers, partitions, and so much more. Just try typing SELECT * FROM sys. and look at the long list in the Intellisense.

Free 3rd Party Tool Alert: RedGate SQL Search is available for free on RedGate.com that makes querying SQL Server as easy as GoFish makes querying Visual FoxPro.

The Devil's in the Details

Well, maybe. But you won't know unless you hit F7 (or View | Object Explorer Details). Then you can select the Tables category in Object Explorer and a list of all of the tables, along with the schema, row count, and creation date, will show in the details pane. But wait, there's more! Right+click on the column headings and you can add columns for Data Space Used (table size), a handy way to see how big your tables are. Unfortunately for views, there isn't a "Last Update" column which is something I could really use. There *is* a "Last Update" column for Stored Procedures though. Weird.

Another way to get some details about anything is to take advantage of the *sp_help* command. To see some details about a table in your database, type *sp_help* <tablename> and run the query (hint: F5). The output pane will now enumerate all the details about the element.

Depend on Dependencies

Ever wonder which views and stored procedures are going to be affected by your change to a table? Just right+click and View Dependencies to see. Conversely, if you don't want to parse through a large stored procedure to see which tables it uses, right+click and View Dependencies, and then toggle the option at the top to "Objects on which [procedure] depends.

Tip 3: Be Quick About It

There are an enormous number of shortcuts to get around SSMS quickly. The first thing a developer needs to know is that many of them are the same ones as available in Visual Studio (again, because they're based on the same IDE framework). So if you want to comment a block of code in a T-SQL query or stored procedure, just hit Ctrl+K, C. I could go on, but there's a full list of SSMS shortcuts available at http://msdn.microsoft.com/en-us/library/ms174205.aspx that you can peruse.

Here are some that I find myself using all the time.

Ctrl+E: Execute the current query. This one seems so obvious that I'm almost afraid to mention it, but I see enough people that do this the hard way (Query | Execute) that I feel I must mention it. You can also hit F5, which is the same as running your app in Visual Studio.

Ctrl+U: This one probably saves more total time than any other. How many times have you typed a query, say SELECT * FROM MyTable, only to have SQL Server come back and say "Could not find MyTable." Argh. Of course not! Because MyTable is actually in a database that's not the "current" database. You could type USE database before you're SELECT (that's the equivalent of SET DATABASE TO and a little counterintuitive for us FoxPro folks). Or you could spell out the entire name of the table, a la SELECT * FROM MyDatabase.MySchema.MyTable, but that's a lot of typing and you don't want to have to do that every time.

The quickest way is to just indicate to SSMS which database you'd like to be current, and there's a dropdown for that right in the toolbar. But rather than reach for that mouse, just hit Ctrl+U, start typing the name of the database you want to be current, and then hit Enter to jump back into your query.

Ctrl+W: Another one that's completely counterintuitive for me, Ctrl+W selects the word under the cursor. I find this handy for deleting fields, parameters, or any other extraneous optional fields in my code. Keep this one in mind and you'll find yourself using it quite a bit.

Alt+F1: We talked about *sp_help* in the previous section, and SSMS expects you to use it so often that it comes predefined with Alt+F1 as a shortcut for it. Simply highlight the name of an object (hint: Ctrl+W) in the query window and hit Alt+F1 to get the equivalent of a new query of sp_help <objectName>.

Note: For tables whose schema isn't explicitly defined, this will assume the 'dbo' schema (more on schemas later). If you want the same thing for a table that's not in the dbo schema, just type sp_help 'schema.objectname.' Also note that if you don't select any object, you'll get a very large resultset back, you got the schema for the entire database and not just the table you were looking for. That may be actually what you wanted, but just be aware of that.

If there's not a shortcut for your favorite stored procedure, you can create one yourself with Tools, Options | Keyboard | Query Shortcuts. Whatever items is highlighted becomes the parameter to your stored procedure.

There's a link at the end of this paper to a site that contains all SSMS shortcuts, but a nice way to deduce shortcuts is to turn on "ScreenTips.

- Open ***SSMS***
- Go to ***Tools*** | ***Customize…***
- In the ***Customize*** Window, ensure that ***Shot ScreenTips on toolbars*** is checked and ***Show shortcut keys in ScreenTips*** is unchecked

Ctrl+Shift+Q

When you're starting a new SELECT query, you can skip handcoding all of the details of your query by switching into the visual designer. This is the same designer that comes up automatically when you start a new view, and is especially handy for quickly assembling a collection of tables with lots of joins. You can then break out of the visual designer and continue with your handcoding.

Mouse Shortcuts

If you're a mouse person, you never have to type the name of a database, table or column again. Simply drag the name over from the Object Explorer and SSMS will insert the name for you. When you drag a table name, it doesn't put the database name in so you may have to use the Ctrl+U trick to set the database to the current table.

While you're looking at the table in Object Explorer, you can drag the word 'Columns' over to the query window and SSMS will insert the name of each column in the table. You can then use the Ctrl+W trick to remove any columns you don't want.

Another handy thing you can do with your mouse is Alt+Select, which allows you to select a rectangle of text and then manipulate within that rectangle. You can delete it, move it over, or search/replace within that selection. This is very handy when you want to remove tables aliases in front of a set of column fields, for instance.

Tip 4: Be Scientific

SQL Server Management Studio has amazing tools for analyzing what's going on behind the scenes. Learning to use those tools to their full potential is the subject of entire books, but here's a brief overview so you can be aware that they're there and what they do.

Display Estimated/Actual Execution Plan

Estimate Execution Plan (Ctrl+L) will take a look at the query and diagram how SQL Server's is going to return the data you're asking for. You want to ensure indexes are being used for as much as possible, and common queries use indexes with contained fields so they don't have to return to the database to get the fields. Be sure to check the Actual Execution Plan (Ctrl+M) as well and compare. The 3rd Party tool *PlanExplorer* may help optimize your indices and queries.

View Client Statistics

Ever wonder if it's quicker to run a query one way or another? SSMS provides a fantastic tool for comparing the execution times of your queries so you can see exactly how long something took without resorting to putting custom code in to track the time manually. Just select Query | View Client Statistics and you'll see. Run a query once, change it around, then run it again. You'll see how long everything took, trends, and even an average column. No more guessing or going by feel.

SQL Server Profiler

While technically not part of SSMS, SQL Server Profiler is a tool that developers should keep in their belt. It's particularly useful for those "black box" applications you inherit and you have no idea what's going on inside (perhaps you don't even have the source code).

With SQL Server Profiler, you can set up "traces" that will log the inner happenings of the server. You can see in plain English the queries sent through ODBC commands. See also the new XEvents in SSMS 2008 R2.

Tip 5: Go Off Script

One of the most difficult things that we'll have to do as developers working with SQL Server is keep the changes we make to our development environment in sync with what's going on in the live application (as well as a test environment, if we're lucky enough to have one). This means changes to the database schema as well as changes to the "code" included in the database.

You can spend some money on great 3rd party tools to help you manage this, but here are a couple of tricks you can do using SSMS natively:

Script Your Databases

SSMS can generate all the code needed to recreate and and all of your database, including data, via scripting. You can then take the results and compare them with previous version of your database or store them in your version control system for an ad hoc database version control.

To do this effectively, you don't want to use the "Script Database As…" option however. There aren't enough options in there, you literally only get the script necessary to regenerate the database itself, not any tables, views, stored procedures, etc.

Instead, right+click on the database and select Tasks | Generate Scripts…

This will then walk you through a wizard that allows you to select which components of the database you want scripted and gives you all sorts of options.

Tip: One option you'll definitely want to turn *off* if you are going to compare database schemas is the *Include Descriptive Headers* option. If you leave this on, you get something like this at the top of each script:

```
/****** Object:  Table [xxx].[xxx]    Script Date: xxx  ******/
```

Because the script date will be different every time you generate the script, this will be flagged as a new version of the schema even if nothing else is changed, causing a bit of extra chaos in your version control system.

Tip: I find myself using Beyond Compare so often that I added it to the Tools menu.

Programatically Scripting

For views, stored procedures, and functions you can execute the *sp_HelpText* stored procedure to have SSMS output the script used to create the object. You can then save those results to a text file, but I'm not aware of a way to directly send the results to a file non-interactively.

Another way to script programmatically is to use the SQLPubWiz.exe utility. This command-line tool allows you to generate scripts from the command line, and I find it very helpful to integrate this into my build process in order to generate a text file that I can check into my source control system.

Go Commando

Speaking of command-line, you don't have to drop to the SQL Command Line in order to run scripts. By going into SQLCmd mode (Query | SQLCmd Mode), you can run commands right from within the query window by starting the line with *:R*.

For example, sometimes you get sent a script (.sql) file that's too large for SSMS to load into the editor window. (The limit of script size is available memory, so that may be a pretty large script.) You can enter the command

:R <scriptname>

And it will execute that script name. I find this useful for those times when someone sends me a large set of INSERT INTO commands in order to populate the database (an alternative way of "backing up" I suppose, and one you can emulate by choosing the "Script Data" option when you generate scripts).

Tip 6: Viewing and Changing Your Data

Eventually it all comes down to the data itself, and sometimes the data isn't quite right. What do we do then? Here's a couple of tips.

Take a look at your data

The first thing we often do when we want to get a look at what kind of data is in our tables is to right+click on the table and choose "SELECT TOP 1000 Rows" (by the way, that '1000' is configurable in Tools | Options | SQL Server Object Explorer).

The problem with this approach is that the first 1,000 records in our table may not accurately represent the data that's in our table. After all it's likely that those were the very first records inserted into the table, perhaps even as an import from another system altogether.

Indeed it would be better to pull 1,000 *random* records out of the table, and there are a couple of ways to do this. The easiest would be to append an ORDER BY NewId() clause to the end of the query. Another way is to replace the TOP clause with the TABLESAMPLE clause of the SELECT command, like this

```
SELECT * [AdventureWorks].[Person].[Contact] TABLESAMPLE (45 ROWS)
```

The TABLESAMPLE clause isn't very precise though. It won't likely return exactly how many rows you ask for whether you ask for a specific number of rows or a percent of the table.

***Note: Generating* Random Data** If you need to populate your development databases with some random data and can't get a copy of your live database, there are tools available that can help you do that. It's also possible to write scripts that can evaluate the schema of tables and come up with suitable test data – hey you're a developer, why not try to write something like that?

Editing Specific Rows

If you want to change some data in your table, you could of course issue the UPDATE command but nothing's quite as convenient as the editable grid you get when you right+click on a table and click EDIT TOP 200 ROWS.

But what do you do in the (likely) event that the data you want to change isn't in the TOP 200? It'd be great if you could just run any SELECT in the query window and then toggle the output window into an Editable grid, but that just isn't possible.

You could go to Tools | Options | SQL Server Object Explorer and enlarge the sample size (or change it to 0 to get the entire table) but then you'd have to scroll down among all the rows to find the rows you're looking for. Eesh.

A better solution is to click on the Show SQL Pane (Ctrl+3) and add a WHERE clause to the SQL that pulled the data. Rerun the query (Ctrl+R this time, not Ctrl+E or F5) and you can now edit a nice subset of records.

Tip 7: Debugging

First tip for debgging: You should never debug on a live server. In fact you as a developer shouldn't even have the *sys_admin* rights necessary to debug on a live server. Even if you are the de facto sys admin, don't use that login when you're on the server as you're opening up yourself to the possibility of locking up that server while you figure out what's wrong with your code. Debug your scripts on a development server and deploy it live only after it's been tested, just like your regular code.

Debugging in SQL Server works just like debugging in Visual Studio. You can set breakpoints, step through code, inspect variables, etc. And just like your regular code, sometimes your scripts will throw an error. You could turn your line numbering on (Tools | Options | Text Editor | Transact-SQL) and scroll your way down to the line with the error, but SSMS makes it easier than that. Simply double+click on the error message and SSMS will take you right to the line with the error.

Tip 8: Know Thyself

Take the time to learn about what SQL Server provides for you in terms of built-in stored procedures and functions. We already talked about *sp_Help* and *sp_HelpText*, but there are a long list of stored procedures available. There are also a ton of built-in global variables that start with @@ that are useful in your scripts (eg. @@ROWCOUNT is useful for returning the number of rows a query produced).

On the flip side, know what SSMS does not include and which 3rd Party tools fill in the gaps. There are some great free utilities as well as commercial ones. RedGate of course is the first name in 3rd party utilities for SQL Server.

Speaking of RedGate, they provide free eBooks for all sorts of topics at http://www.red-gate.com/community/books/. Check that out along with some of these other links that I found helpful when putting this list together.

There's so much more to SSMS than I could cover here, but I hope this session opened up some possibilities for making you more efficient when using this tool. This list gets larger every time I give this

session, so if you have ideas that you think I should add to this session please get in touch via email or Twitter and I'll definitely include them.

References and Tools

Full List of Keyboard Shortcuts: http://msdn.microsoft.com/en-us/library/ms174205.aspx

http://www.mssqltips.com/sqlservertip/2415/ssms-keyboard-shortcuts-part-2-of-2/

http://www.mssqltips.com/sqlservertip/2145/starting-your-sql-server-career-path/?utm_source=dailynewsletter&utm_medium=email&utm_content=headline&utm_campaign=20122411

http://www.databasejournal.com/features/article.php/3593466/MS-SQL-Series.htm

SQLServerCentral.com

Blog.SQLAuthority.com

http://www.mssqltips.com/

http://www.codeproject.com/Articles/39131/Global-Variables-in-SQL-Server

T-SQL Formatter: http://www.architectshack.com/PoorMansTSqlFormatter.ashx and PoorSQL.com