Walkthrough of LocalDB

This is to demystify some of the stuff around using Sql Express Local DB. Local DB would be an ideal option for student projects, as the database can be included in github alongside the code, to make it easier to run a project.

The attempted structure of this walkthrough is:

- I'm going to try to go the long way around, to show all the little bits that make up the system
- Then, i'll go the short way at the end. At this point, it should make sense, there shouldn't be any "magic" to it.

This entire document is one person's opinion. Your mileage may vary, etc. Find me on slack @sunny.gulati if something's not right for you.

CONTENTS

What Is LocalDB?

Best explanation I found:

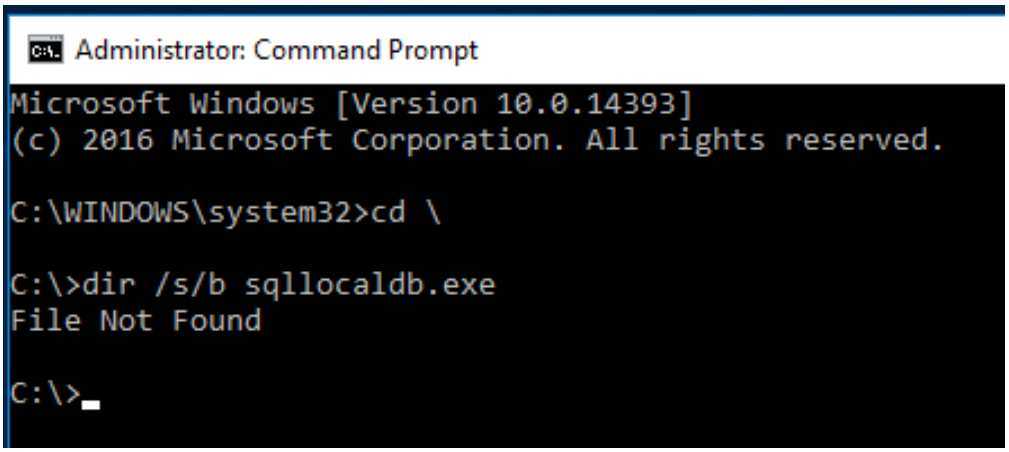
- https://blogs.msdn.microsoft.com/sqlexpress/2011/07/12/introducing-localdb-an-improved-sql-express/

Summary:

- In the past, setting up a database was a large detour that developers had to take in order to get an app up and running from source.
- LocalDB is an answer to that. It can:
    - Respond to the desire to have a database, and auto-create a database server for you
    - Load up a database provided in a file into an auto-created database instance.
- It does not
    - Work outside of your local machine. Can only use it for local development.
- Microsoft has started distributing it with Visual Studio.

Installing LocalDB

I have installed Windows 10 + all the updates (as of 9/30/2016) on a VM .. but have not yet installed visual studio. Let's see if SQLLOCALDB is installed. I'm going to use an administrator CMD.EXE (Win-X + A), and search the hard drive:



Installing via a Download

My research found a nice overview via stackoverflow:

- http://stackoverflow.com/questions/23320013/how-to-install-localdb-separately

Which points to an official Microsoft Download location:

- https://www.microsoft.com/en-us/download/details.aspx?id=29062

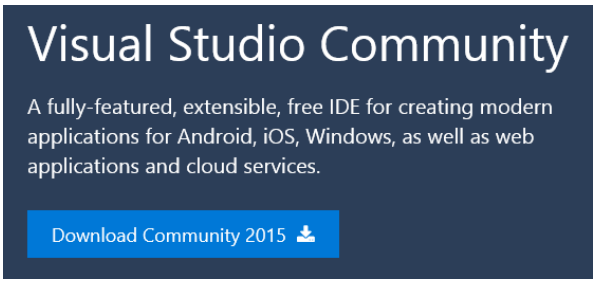
LocalDB is one of the (smallest) options in the SqlExpress download page:

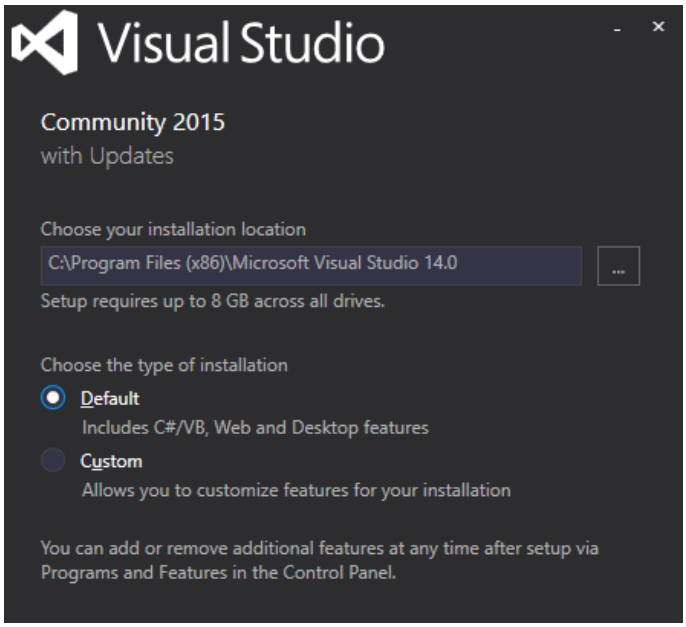
-

However, just installing Visual Studio 2015 Community Edition *should* do the trick as well. That's the route I'll take.

Installing via VS2015 Community Edition

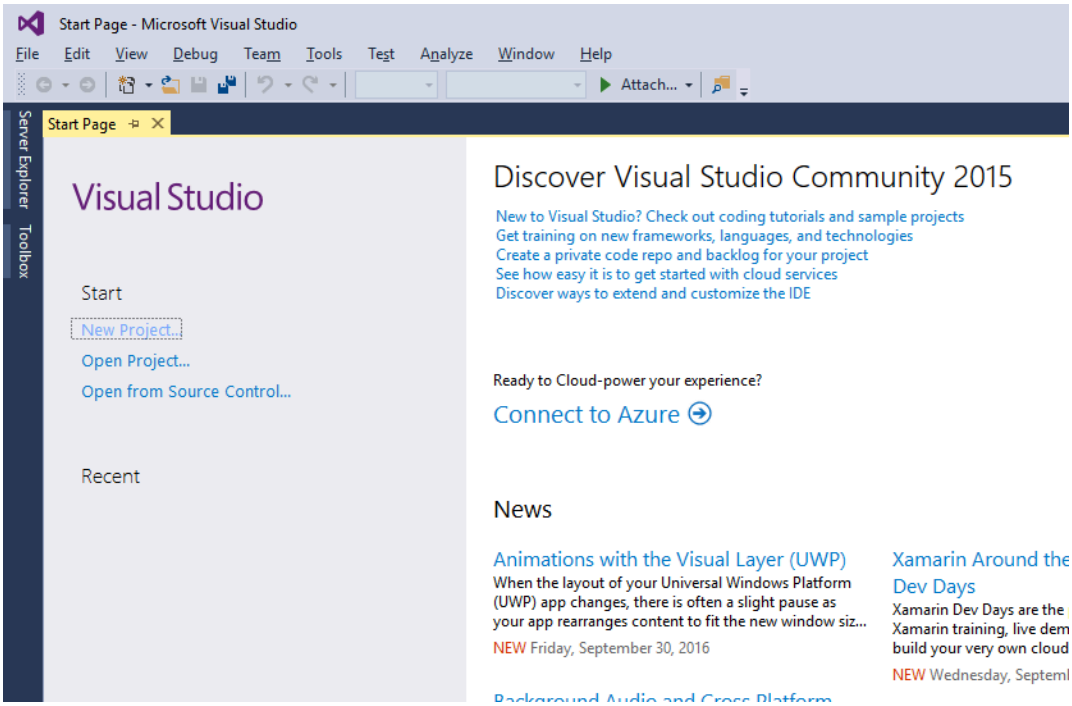
- https://www.visualstudio.com/vs/community/



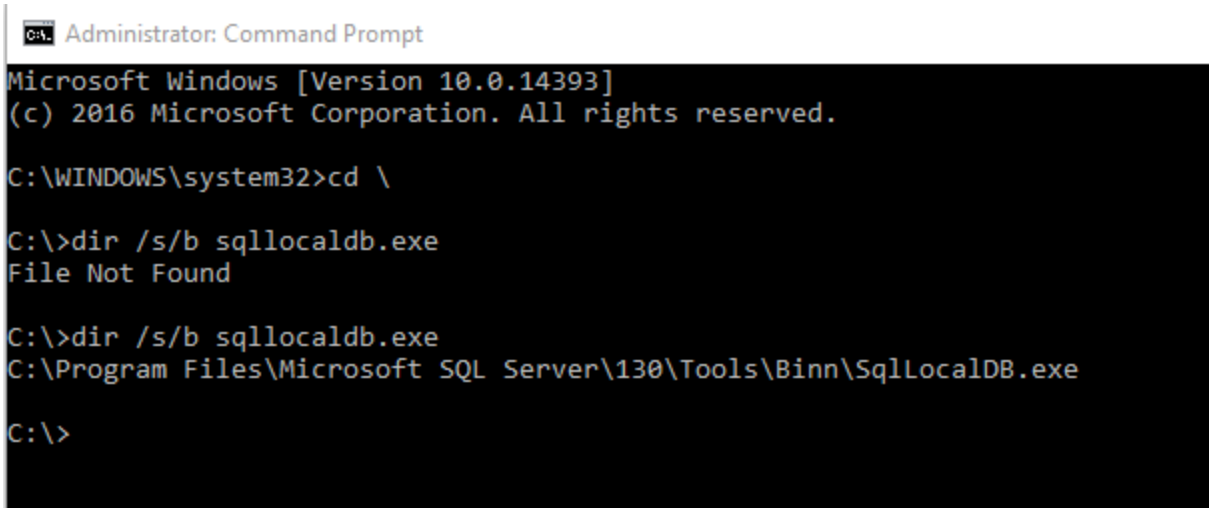


I'm choosing the Default option, which is what I'm hoping most people did.

1.5 hours later ..



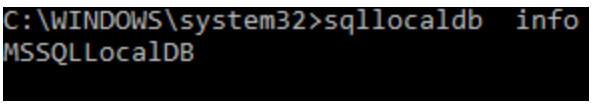
Its installed! And if I re-do my search for sqllocaldb on the hard drive:



A couple of things:

- The “130” there is effectively the SQL Server version. It doesn’t match the 2016 year that’s in the name of the product. Here’s a list of mappings: http://stackoverflow.com/a/18754188
- VS2015 Installation should have already added it to your path.
- If you run that executable without any parameters, it gives you a bunch of help.
- There’s not much you can do with it. It effectively starts and stops a database server for you, but that’s about all. It won’t run any SQL.

If I run it with “info” to list the databases servers / instances that exist, It shows just the default instance. This name is standard as of SQL Server 2016.



Clarifying Server, Instance, Database, Table, Owner, Schema, Login, User ...

There’s a lot of terms in database land.

What's worse is, some database systems use different terms for different things.

I'm going to refer you to this as an example of the complexity that you can get into: https://www.connectionstrings.com/sql-server/ And specifically call out this connection string

```
Server=myServerName\myInstanceName;Database=myDataBase;User Id=myUsername;
Password=myPassword;
```

And this other one:

```
Data Source=190.190.200.100,1433;Network Library=DBMSSOCN;
Initial Catalog=myDataBase;User ID=myUsername;Password=myPassword;
```

Thingy	Description	Connection String Component
Machine / Host	This is something you can ping, or connect a keyboard/monitor to. It will typically have multiple Instances of sql server installed on to it. The default instance is called "default" and listens on port 1433.	Server=**myServerName**
Instance	You could have 3 separate "instances" of SQL server installed on a single machine. They might even be different versions..one is 2016, one is 2014, for example. Each one has a name, and probably listens on a different port number.	Server=myServerName**myInstance Name**
Sql server Browser	This is a bit tricky. Every instance of sql server on a machine has a different name; however, because they all listen on different IP addresses, there's a separate thingy that listens on the default port. When somebody connects to it, if they ask for instance ABC, it knows "oh, that's on this other port", and a handshake happens and you end up connecting to the correct instance and port number without knowing about the port numbers. Sometimes, those port numbers are random, which is really fun when debugging firewall rules, but I digress.	This is how \instance1 can be differentiated from \instance2.
Port Number	Sometimes, if sqlserver browser is not an option, you might connect directly to a port number. In this case, its specified with a **comma (!)**	Data Source=190.190.200.100,1433 Turns out, Data Source and Server are aliases for each other. See the docs for a full list of aliases: https://msdn.microsoft.com/en-gb/library/system.data.sqlclient.sqlconnection.connectionstring.aspx
Database	Once you connect to an instance, within that instance you will find: • databases named "master", "model", "msdb", and "tempdb"	Database=myDatabase
Table	Within a database, you'll find multiple tables, which	

	belong to a schema (default schema = “dbo”)	
Schema	In SQL server, schema’s are like “categories” for things. Most folks just use dbo, but when you get really big databases with 100’s of tables, new schemas will be created to group the tables together. You can have two tables named the same if they’re in different schemas.	
Login	This is the username / password that goes in the connection string. This is the thing that lets you IN to the server.	User Id=myUsername; Password=myPassword
User	This is the thing defined per database. So for example, the login “Fred” might map to the user “Fred” in database1, but might map to the user “Frederick” in database2.	It gets very confusing. Usually it’s the same name all the time. But if you ever back up a database from one machine and restore it to another, the User’s come over with the backup, but the Login’s do not, and the mapping has to be set back up.

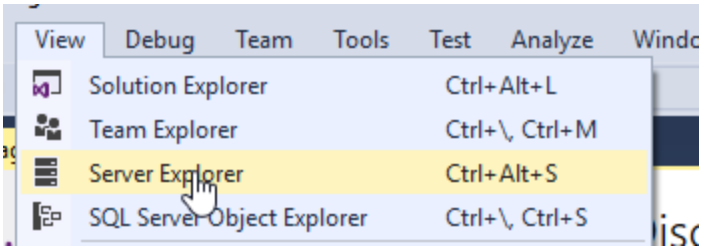
And how does this relate to LocalDB:

- The list of names from sqllocaldb.exe info is a list of INSTANCES on the machine.
- Each instance has its own master + list of databases.
- There’s no user / login stuff. If you’re on the machine, you’re allowed in.

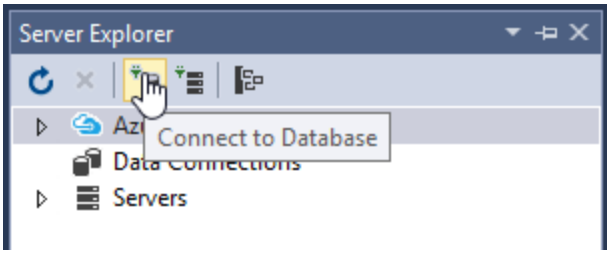
How to use this MSSQLLocalDB

Connecting via Visual Studio

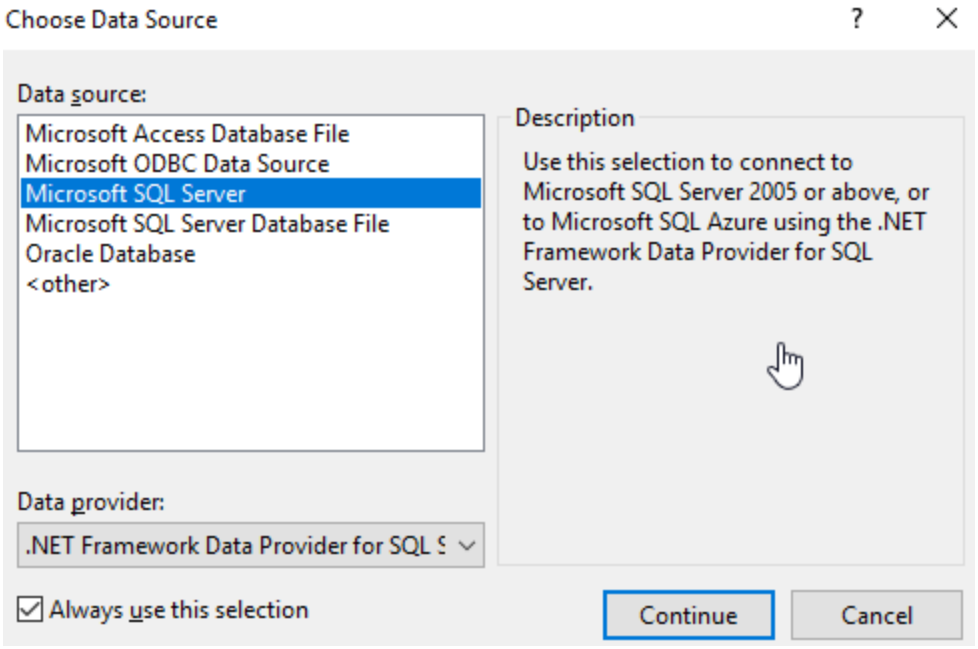
Open up Server Explorer:



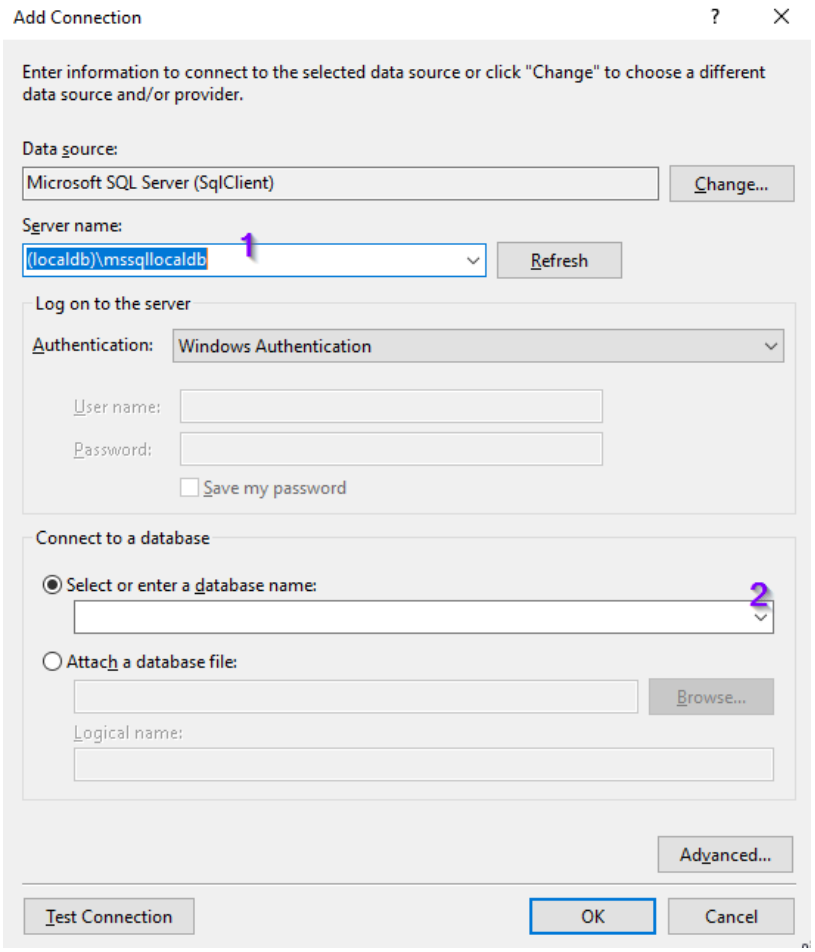
Click “Connect to Database”:



Choose “Microsoft SQL Server” .. but note that “Microsoft SQL Server Database File” for later…

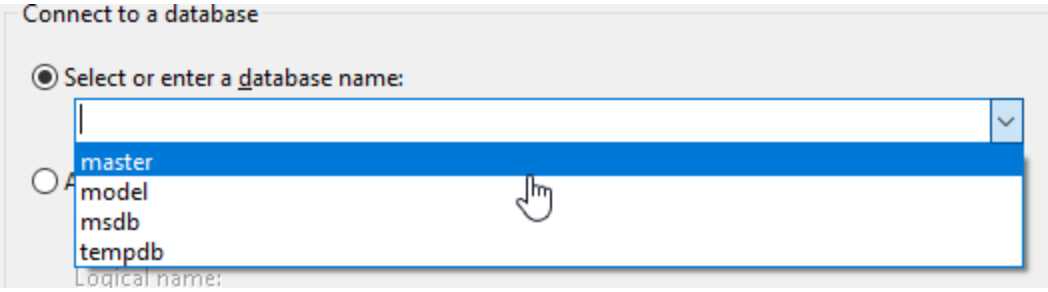


You'll get a dialog asking for the details of the SQL server to connect to. Put in (localdb)\mssqllocaldb as the server to connect to, and then click on the down arrow by "select or enter database name"



- The (localdb) means, "Yo, LocalDB, i want to connect to one of your instances"
- The mssqllocaldb was the name returned from the command line sqllocaldb info that we ran earlier.
- Windows Authentication is (AFAIK) the only thing that works here. It means "trust me, I am who I am".
- There might be a 5 second delay as your machine spins up that database (it only runs when connected to).

What should happen is the list of database names drops down, and you find a list like this:

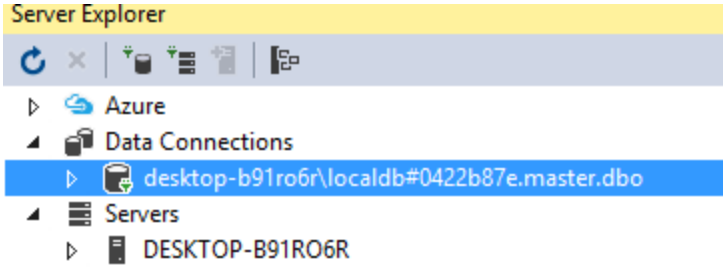


Choose master, and click OK. We'll create a database to work with shortly.

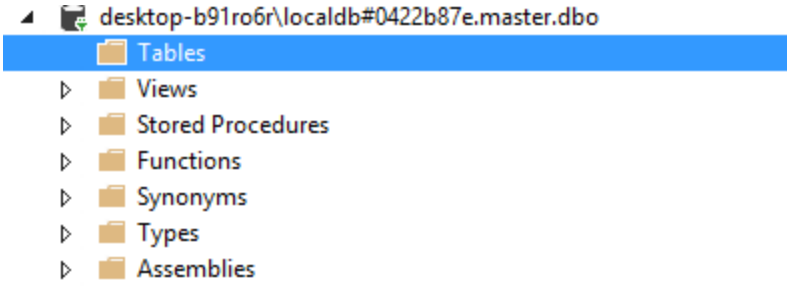
Quick note about master

master is an automatically created database that always always exists. Its job: its a database which contains the list of other databases, what tables they have, where on disk they are, all that kind of stuff. In a regular database environment, only the super-duper-dba's (sa's, system administrator accounts) have access to master.

We now get a connect to that database in the server explorer list:



If you expand the triangle, you get access to all kinds of goodies in there:



From here you can right click to get access to many other things, including a query window from which to run SQL.

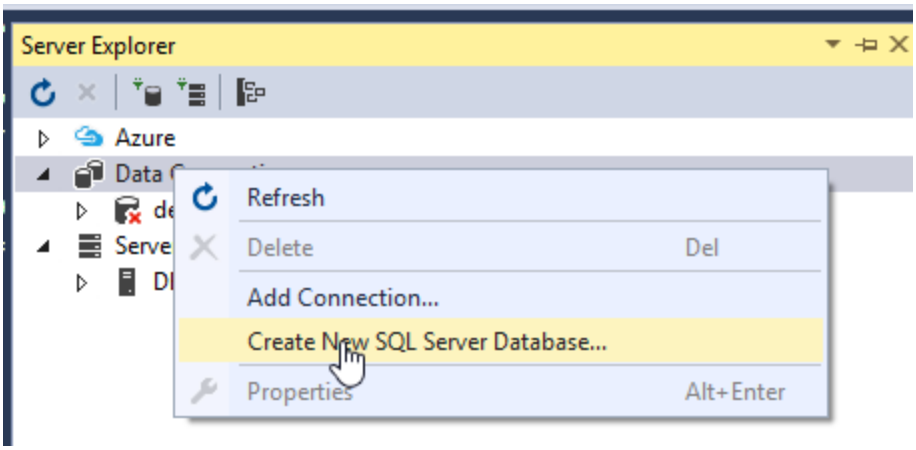
CRUD using Visual Studio + Designers

First, lets create a database to work with. Due to the nature of this walkthrough, i'm probably going to need several database names -- i'm going to go with Star Wars, Game of Thrones, and Pokemon, as my topics of choice.

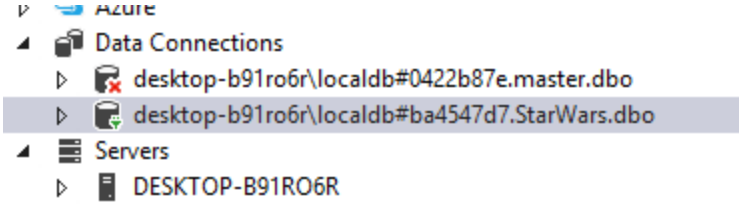
So, lets work with Star Wars. I want a Star Wars database. With a Planets table, and a People table. (I'm getting my inspiration from SWAPI - https://swapi.co/documentation)

Create Database

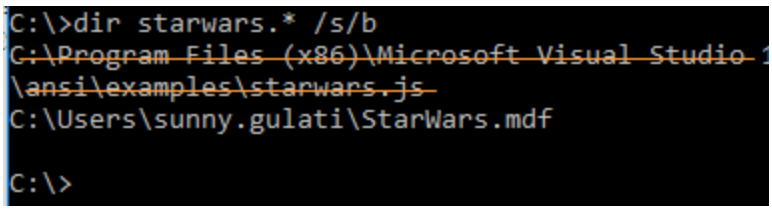
Start by right clicking "Data Connections" and choosing "Create new SQL Server Database…"



Provide the same connection string as before, and your new database name. Click OK.
After a bit of spinning, you get a connection to the new database:

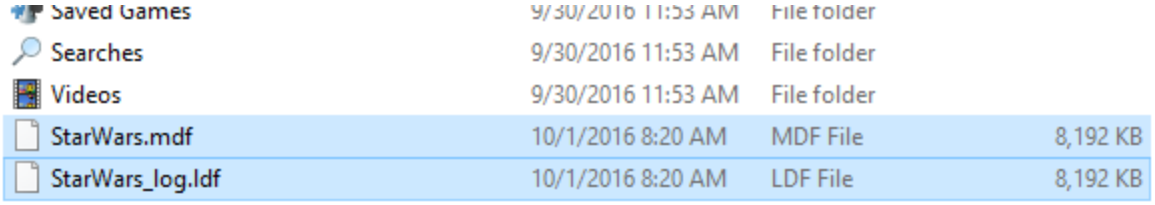


And where did this get created?



- Ignore the starwars.js. I had no idea that visual studio had some sample node code that had a starwars.js file in it when I picked the name StarWars
- The database was created in our home directory.

If we look in our home directory, it's not just the .mdf file; instead there are two files that make up the database - the "data" file (mdf) and the "log" file (_log.ldf):



What is the log file? https://en.wikipedia.org/wiki/Transaction_log

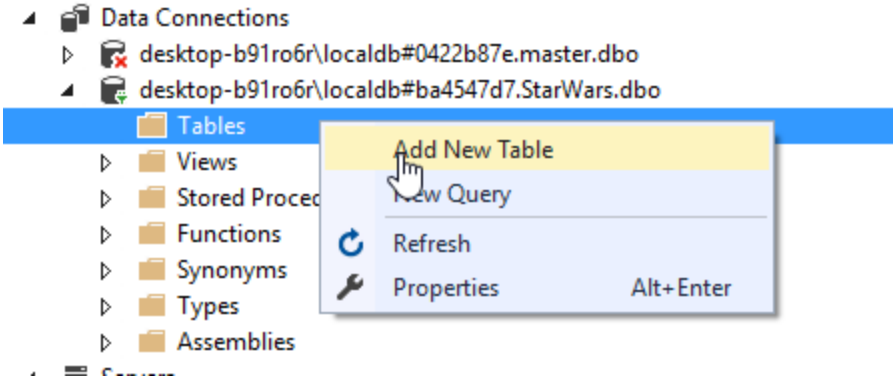
How to move this into your project?

I'll be going into that detail in a later document/video, but the answer is **`~/App_Data`**, and connection strings with a clause of "**`AttachDbFilename=|App_Data|\MyDataFile.mdf`**" in your project file

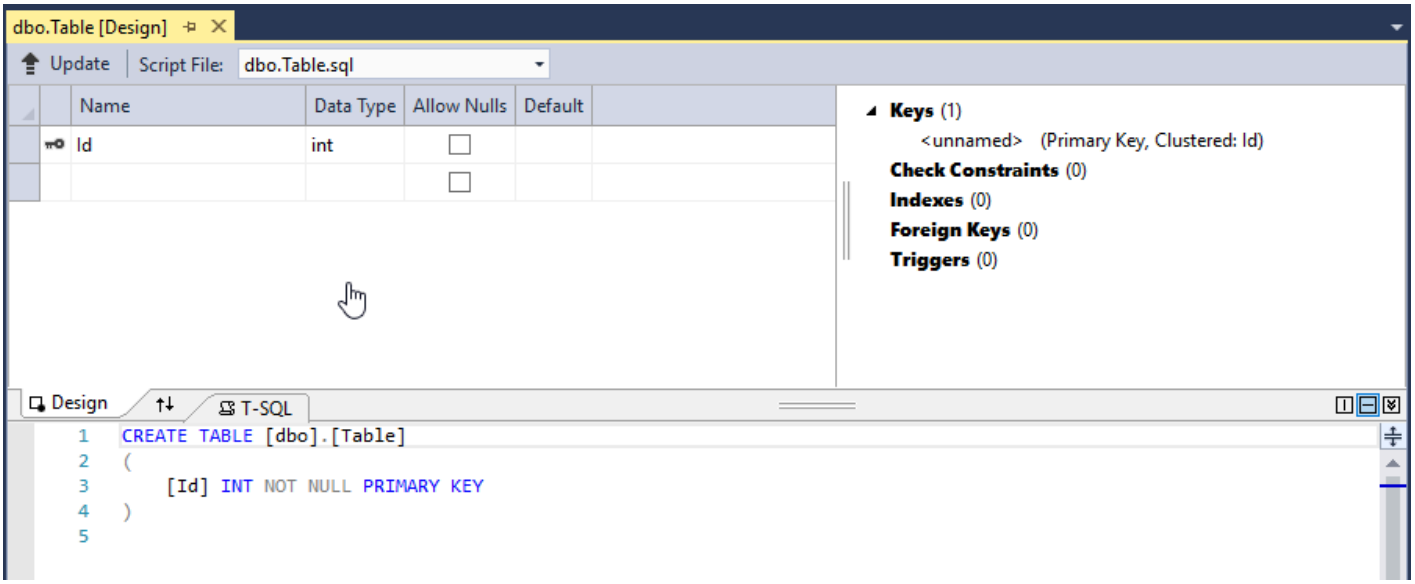
Ok, good enough. Lets make a table, and work with some data.

Create Table

Expand the triangle for the database, get down to tables, and right click, Add new Table.



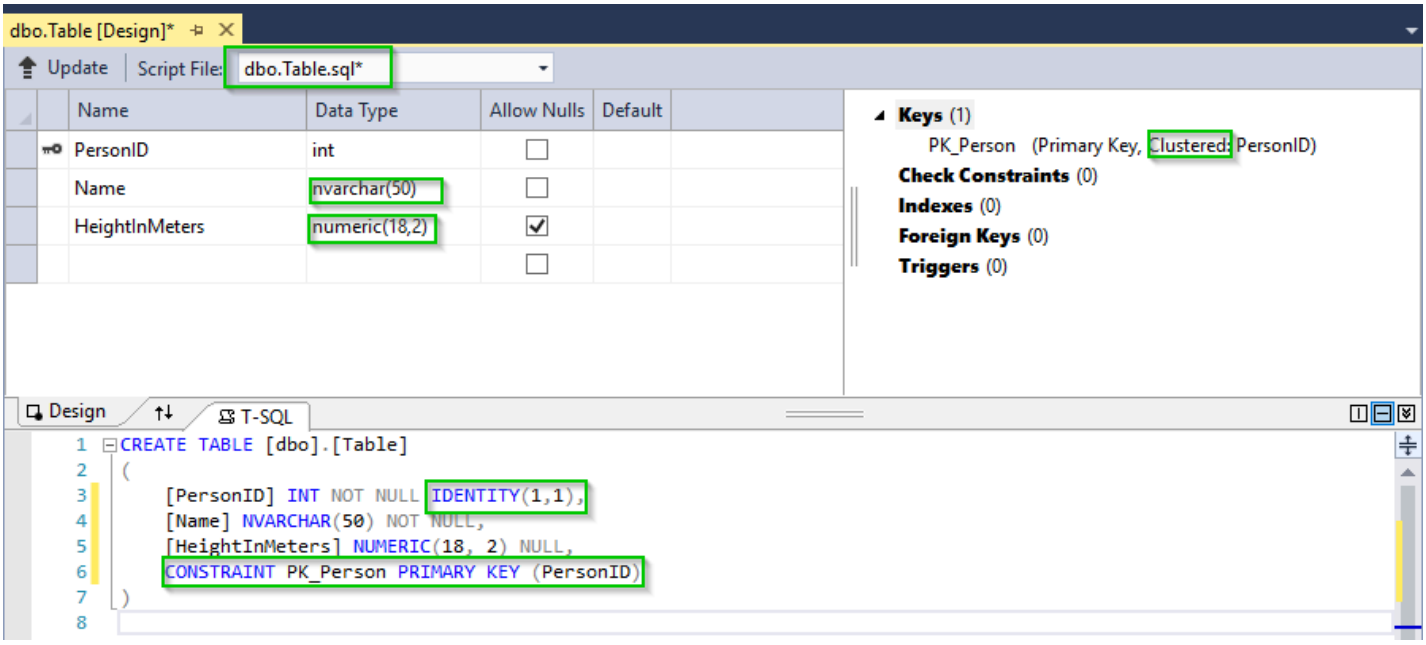
It takes a while, but eventually you end up at a designer, which I'm told looks very similar to MySQL:



This gives you two ways to work with the table design -- either in the grid in the upper left, or by writing a DML (Data Modelling Language) statement in the bottom. Changing either one changes the other, which is pretty cool.

I'm going to fill in some stuff -- and if you haven't seen this before, it can be confusing. Here's some of the decisions / guidelines I'm using -- It's not super-important to follow these your first time around, but when you get on a team with a team project, it becomes important.

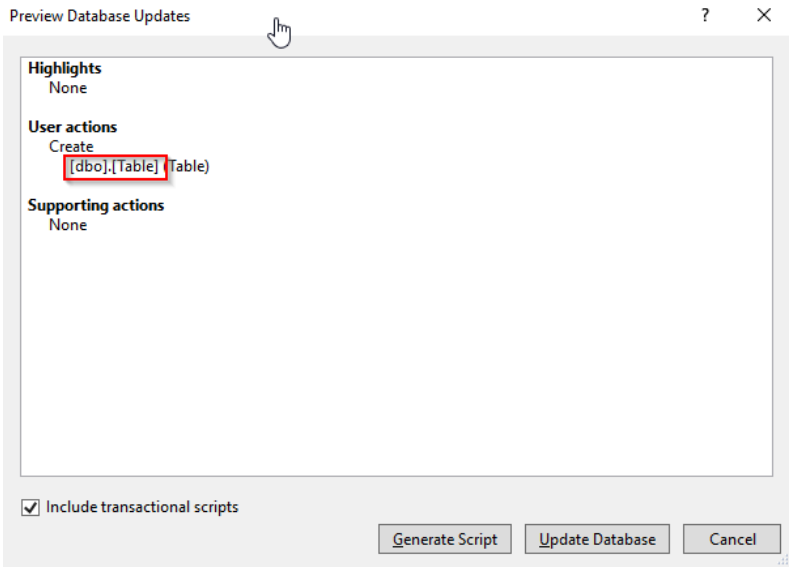
- I personally favor having the ID Column be prefixed with the name of the Table. This way, when I join between tables, the name of the column is the same on both sides of the join, rather than one side having just the word "ID".
- I'm also going to use Sql Server naming conventions, which is "LikeThis" instead of "not_this" or "notthis".
- After great debate at my workplace, we settled on "ID" instead of "Id" because it stands out more. (That was a 2-3 hour discussion followed by a vote).
- Using Singular rather than Plural for the table name. This is so that when the object gets created, it's the same name as the table that it came from, but in memory as an object, there's only one, so … singular.



Whoa! Even more stuff. Lets see:

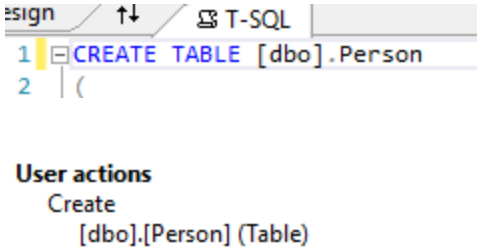
- If you don't know how long something is, don't use MAX. :) I've found that 50 is a nice number.
- Nvarchar instead of varchar, most of the time. "N" means it can hold international characters. Takes up more disk space. Usually not a problem.
- If you have something that doesn't need decimal places, you want **int** most of the time. (Sql Server int's are pretty big). But if you need decimal places, then usually (18,X) is the way to go; usually (18,6) is a good default. The 18 maps to "the most decimals you can get from a N-byte floating point number without going up to the next larger size of floating point number", and it seems to be a theme repeated between Oracle and Sql Server. (And I could be wrong about this).
- I'm specifying that PersonID is an IDENTITY column, as in, SQL Server will provide the numbers and do the automatic counting for you. It starts at 1, and increments by 1, hence (1,1).
- The Table doesn't have a name yet. That's the "*" next to the name in the tab. ~~When you go to save the table, it asks for a name at that point. You could provide a name in the "Create Table" statement, then it won't ask during save.~~

I click Update, and:

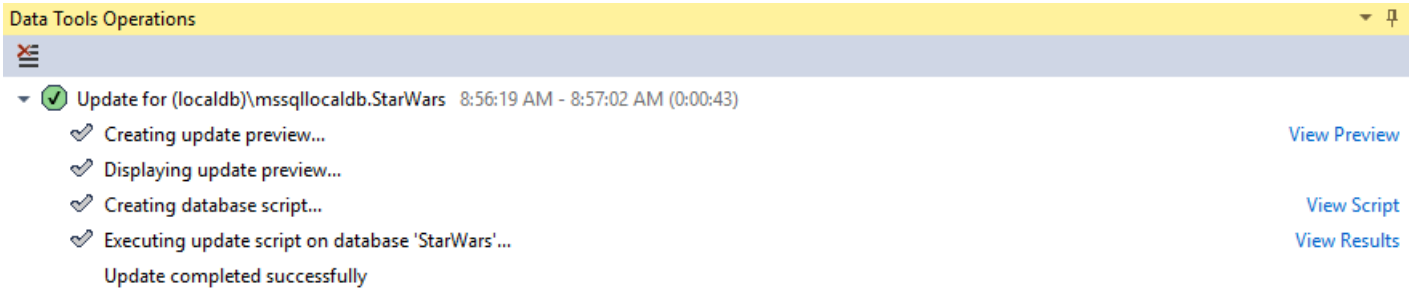


Wha?? No table name. Okay, this is different from the other tool that many folks use, which is SSMS (Sql Server Management Studio)

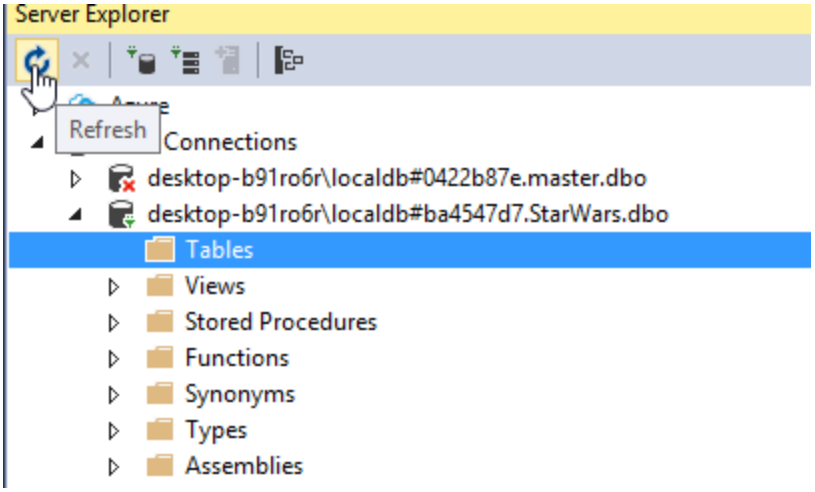
Cancel, go back and put in a table name, and try that again:



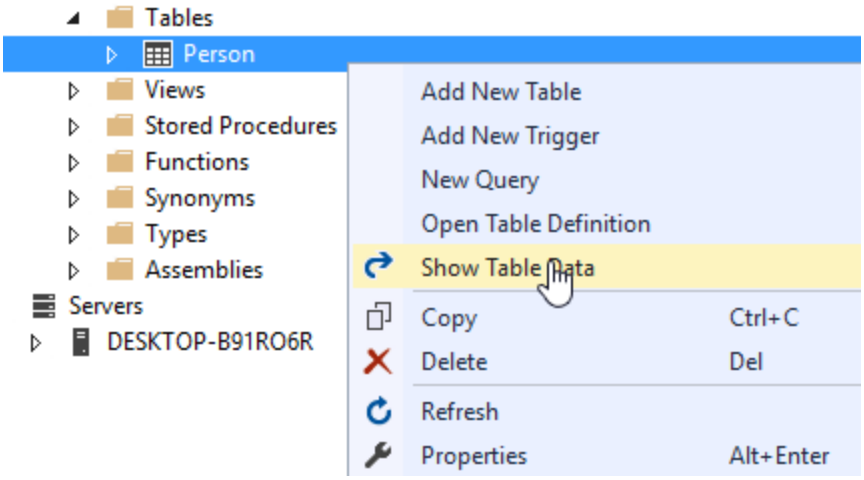
That's better. Click "Update Database". A "Data Tools Operations" window pops up with a summary of what's done, and hopefully a green checkmark saying the update was successful:



If we go back to server explorer and click the refresh icon while the database is highlighted

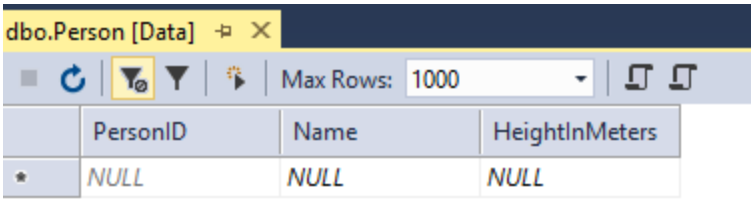


It looks for the new table, and now we get a bunch more options.

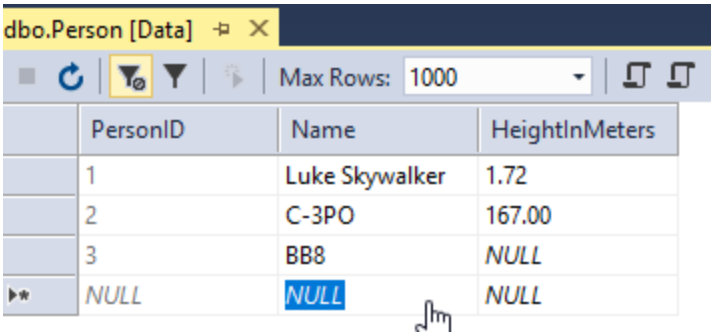


CRUDding Data

Choosing that handy dandy "Show Data" option there, gives us this screen:



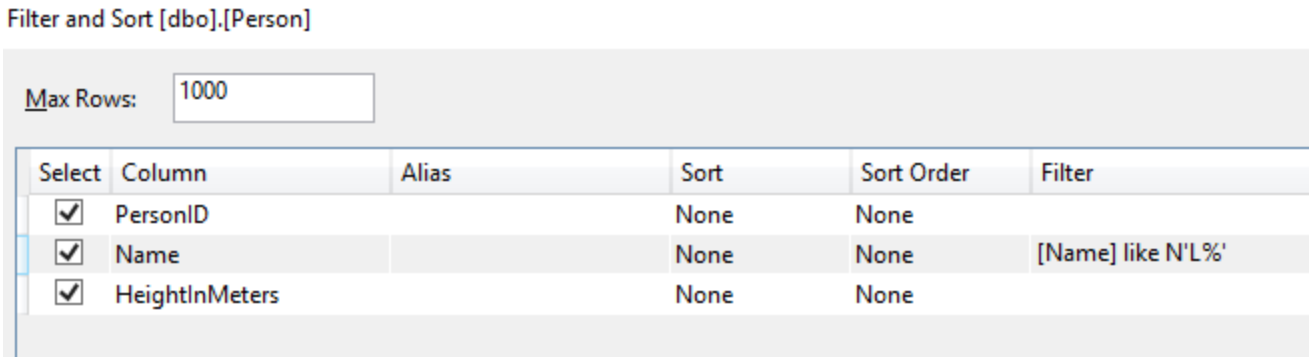
We can use this screen to Insert, Delete, Update, and View data. Hint - don't put anything in PesonID, the database takes care of that. I'm going to put in a few records (stolen from SWAPI) -



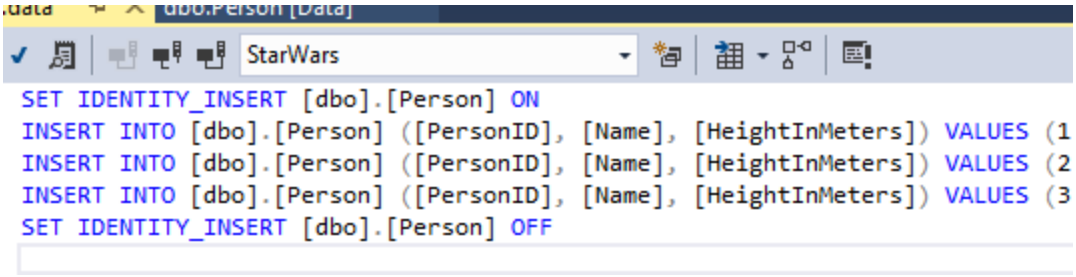
(Turns out BB-8's height is unknown in SWAPI)

Right click a row to bring up the delete option. Also note that unlike the Designer for create table, the rows are immediately saved when you leave them.

There is also a "filter" screen where you can limit the data shown:

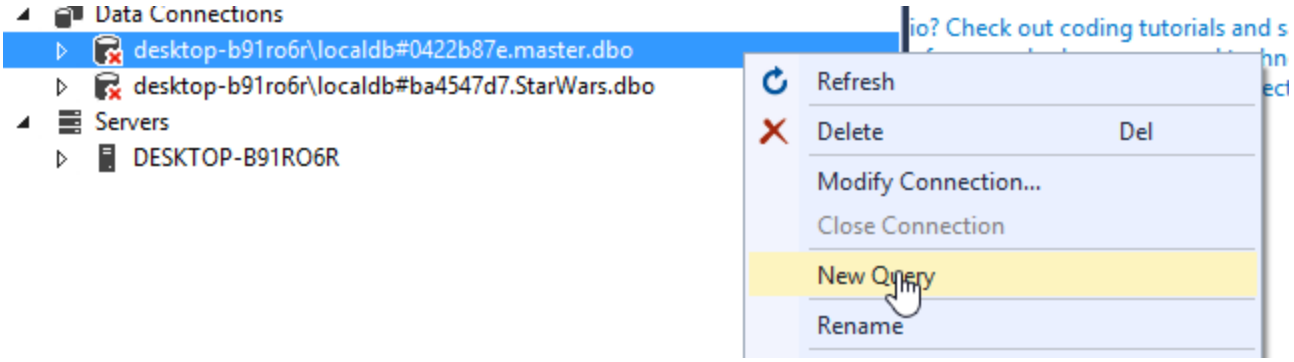


And the buttons on the right of the grid give you options for generating SQL for inserts:

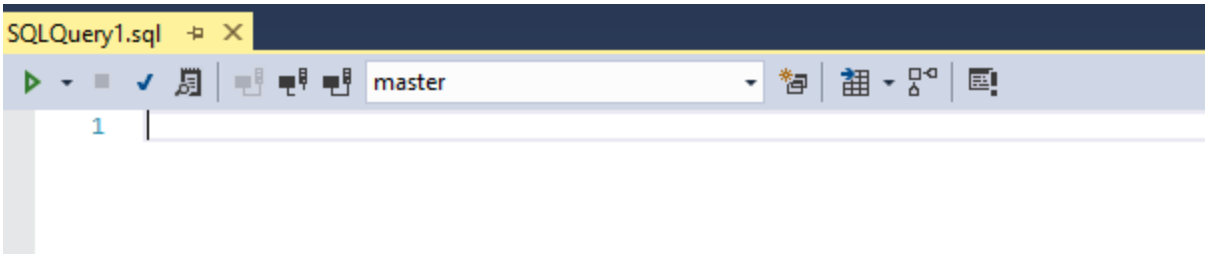


CRUD using Visual Studio + SQL

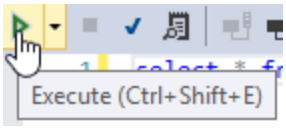
Going to mention this briefly, because it's a skill you ought to know exists. I'm going to start of with a Query window pointing at the "Master" database --



We get the query window, which has a LOT of things going on in it:



From left to right:

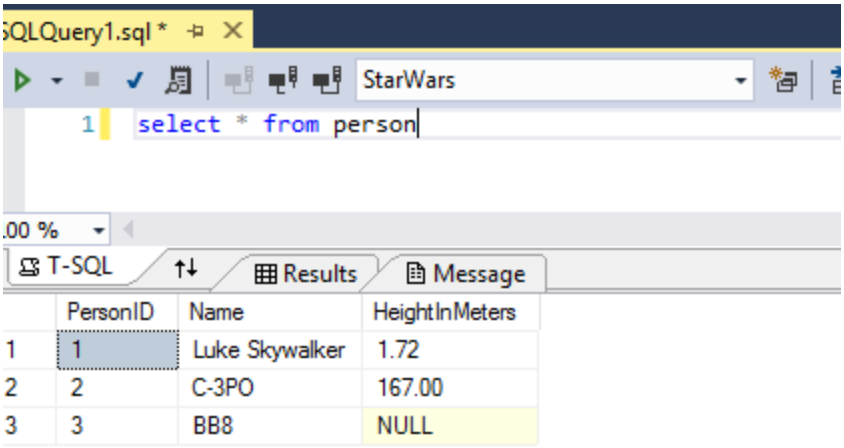
- Execute query + Execute with Debugger
  - 

Many things have hotkeys!
- Stop Execution
- Check syntax
- Display Estimated Execution Plan
- Connect, Disconnect, Change Connection
- Choose which database (master vs starwars for example) that you're working with
- Create new window
- Setting: Results as Grid, Text, File
- Setting: Display actual execution plan
- Setting: "SQLCMD" mode.

From here, i can type in lots of SQL commands to do all the stuff I did above. For example:

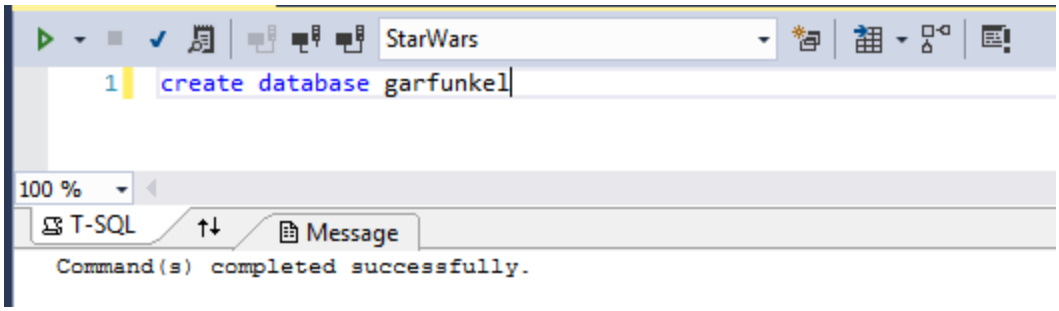
- use starwars
  -  -- execute. The database selected switches: 

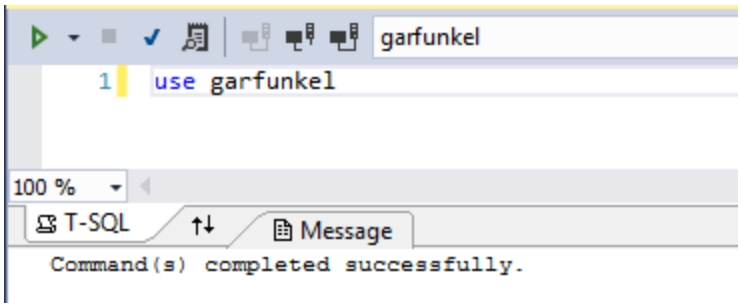
- Select * from Person
  -  -- results are displayed:

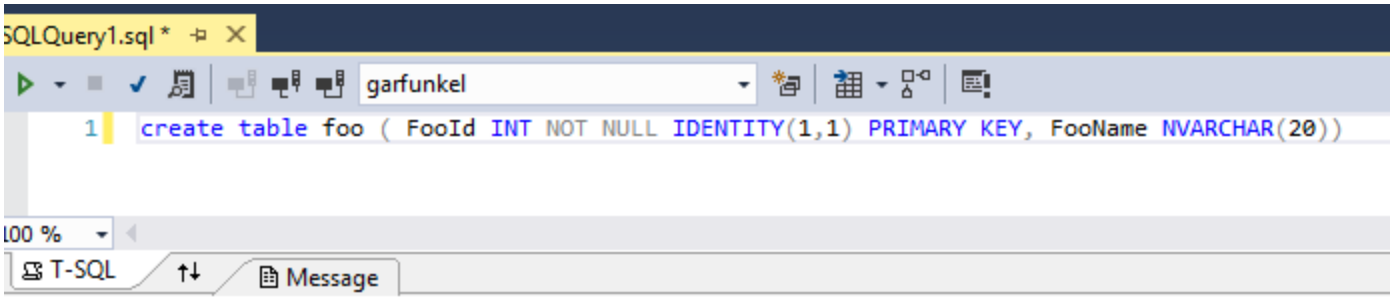


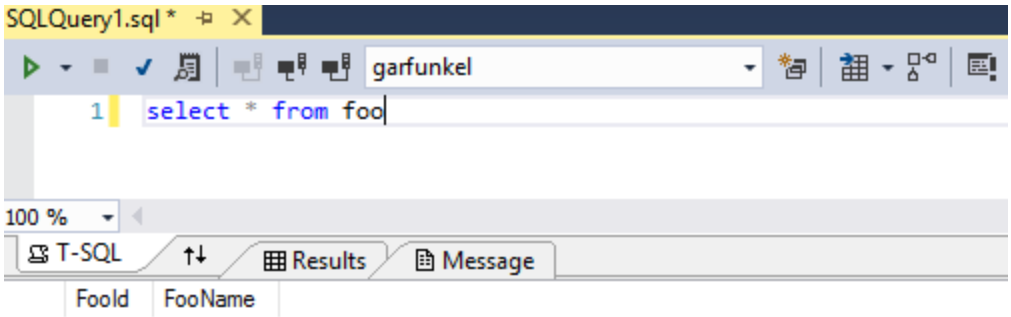
	PersonID	Name	HeightInMeters
1	1	Luke Skywalker	1.72
2	2	C-3PO	167.00
3	3	BB8	NULL

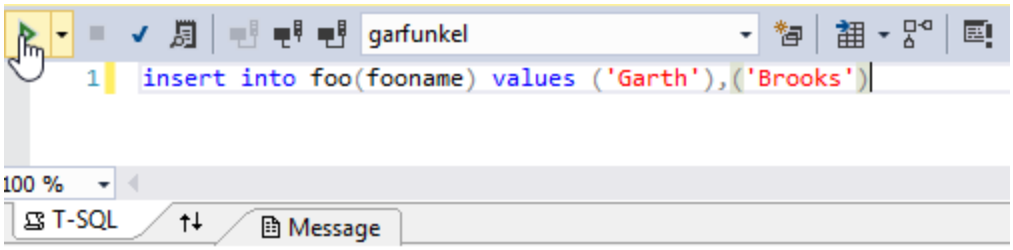
Gah, this is taking too long. I'm going to take pictures after i've run execute to speed things up a bit. Look in the query window for what it is I was doing, and you should see the results in the bottom part of the screenshot.

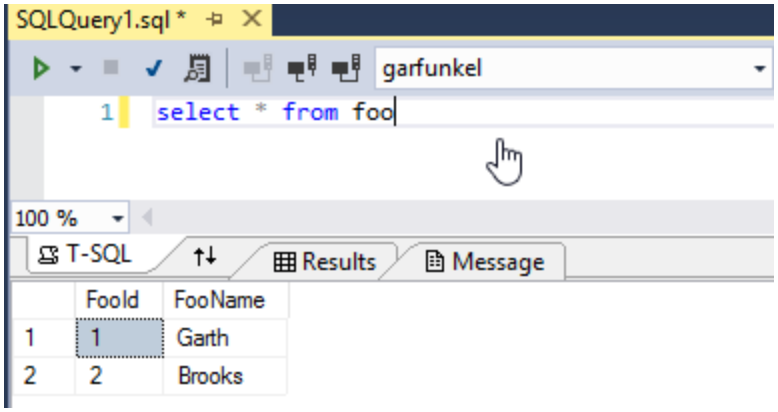












What I'm trying to show here is, I didn't have to use a designer to do stuff. I could use only the SQL Query window and do all the operations i did before -- creating the database, creating the table, inserting and selecting data, and (not shown) update and delete statements as well.

Bonus Section: CRUD using SQLCMD

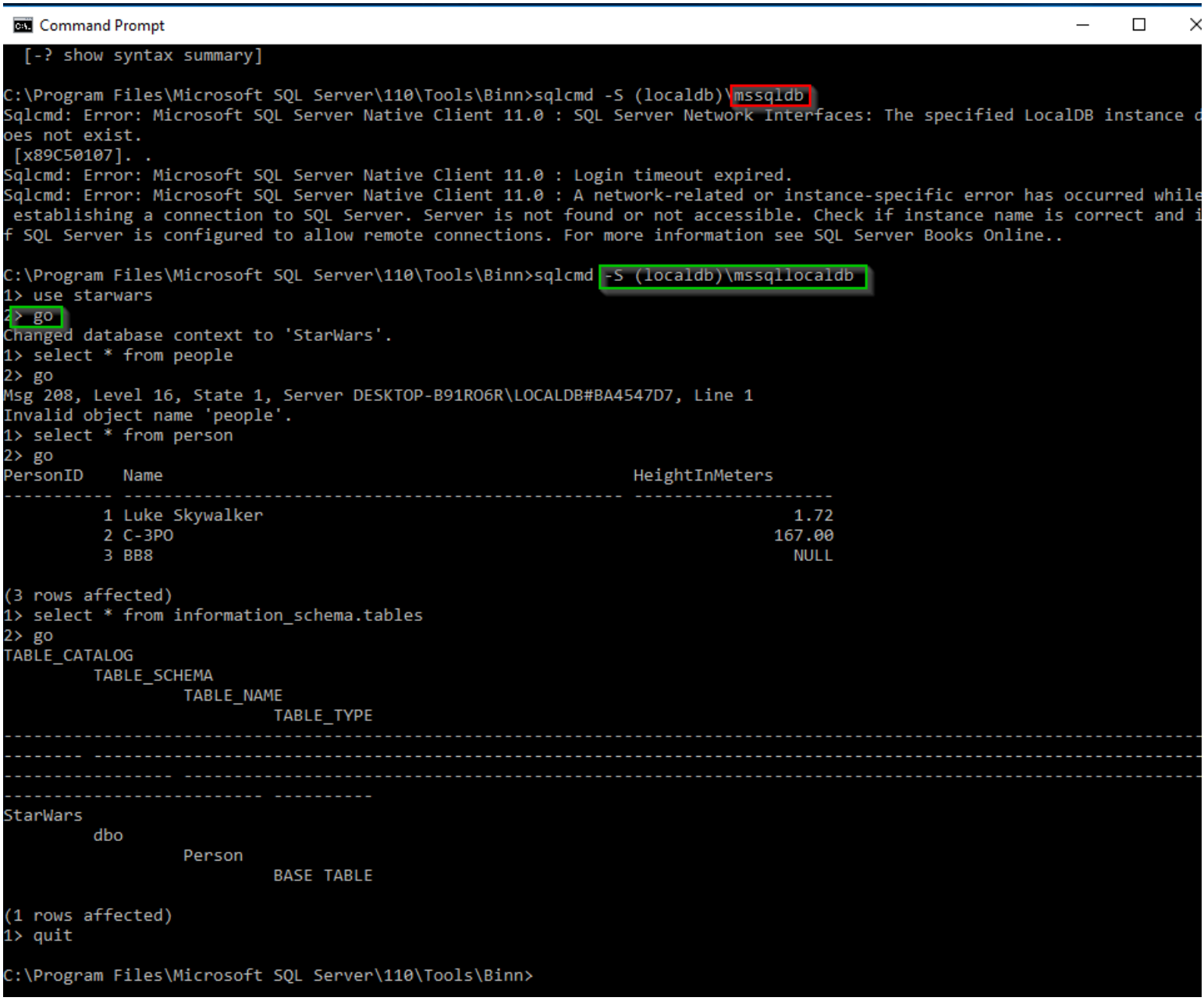
There's another level that you should be aware of. **Most developers never touch this in .Net land,** but it is the equivalent of what folks have to do in other database systems, so we should cover it. Also, if you ever have to automate doing something to a database -- from a scheduled task or from a CI system -- you'll want to know this.

There's a program, "SQLCMD", that provides a command line interface to connect to SQL server databases. Our default installation did not install the version that goes with database version 130 (which is in the path), but somebody did install it for version 110:

```
C:\>dir sql*.exe /s/b
C:\Program Files\Microsoft SQL Server\110\Tools\Binn\SQLCMD.EXE
C:\Program Files\Microsoft SQL Server\130\LocalDB\Binn\sqlservr.exe
C:\Program Files\Microsoft SQL Server\130\Shared\SqlDumper.exe
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\SqlLocalDB.exe
C:\Program Files\Microsoft SQL Server\90\Shared\sqlwriter.exe
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6 Tools\SqlMetal.exe
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools\SqlMetal.exe
C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\SqlMetal.exe
C:\Program Files (x86)\Microsoft SDKs\Windows\v8.1A\bin\NETFX 4.5.1 Tools\SqlMetal.exe
C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE\Extensions\Microsoft\SQLDB\DAC\120\sqlpackage.exe
C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE\Extensions\Microsoft\SQLDB\DAC\130\sqlpackage.exe
C:\Program Files (x86)\Microsoft Visual Studio 14.0\SDK\Bootstrapper\Packages\SqlExpress2008R2\SqlExpressChk.exe
C:\Program Files (x86)\Microsoft Visual Studio 14.0\SDK\Bootstrapper\Packages\SqlExpress2012\SqlExpressChk.exe
```

If I change directory there, and run it with "-?", get a lot of options. There are many documentation pages that go into these options, which have been about the same for a decade:
https://msdn.microsoft.com/en-us/library/ms162773.aspx

Here's a sample of using it, with some highlights:



- -S (localdb)\mssqllocaldb is how you specify which database to connect to. You don't need the -E (trusted connection) or -u -r -p parameters (because localdb doesn't deal in usernames and passwords)
- I did not specify the -d parameter, but that's how you could choose between StarWars and Garfunkel.
- Once in, "go" is the keyword for "execute"
- If you don't know the name of what you want, there's something called "information_schema". Its named like that because its a standard across several database engines -- i think it started with oracle. There are other tables (systables, syscolumns, etc) that are specific to sql server, but i've never memorize them -- instead, i know information_schema.tables, .columns, and that pretty much gets me everything i need.

Interlude 1.

Woot. We have come a LONG way. This would be a good time to stop, get a coffee, stretch some legs, let the dogs out to pee, etc. Try some of this stuff out, see if you can create a database.

Future -- the OTHER tool that most .Net folks use, which is Sql Server Management Studio. Probably a separate document.

Next Document: Connecting from C# to LocalDB