Windows Azure SQL Database Development with the dtSearch Engine

Bruno Terkaly, 6 Nov 2013

In this post will focus on how you can index a database table so that you can perform lightning quick full-text searches

Editorial Note

This article is in the Product Showcase section for our sponsors at CodeProject. These articles are intended to provide you with information on products and services that we consider useful and of value to developers.

Overview

Windows Azure SQL Database does not support the **CONTAINS** keyword. **CONTAINS** is important because it supports searches for precise or fuzzy (less precise) matches to single words and phrases, words within a certain distance of one another, or weighted matches. **CONTAINS** is a predicate used in the **WHERE** clause http://technet.microsoft.com/en-us/library/ms188047.aspx of a Transact-SQL SELECT statement.

But there are ways to get this functionality with the help of third party software. I recently downloaded all the movie information for films that took place in San Francisco. The San Francisco government website provided this interesting information that I wanted to search through very quickly. The data wasn't very normalized so there are many columns that are difficult to search. I downloaded this information (https://data.sfgov.org/Arts-Culture-and-Recreation-/Film-Locations-in-San-Francisco/yitu-d5am) and imported it into a database. Specifically I used SQL database hosted in the Windows Azure platform.

My goal was to be able to find records in the database table, searching in all columns and in all rows quickly and efficiently. Essentially, I wanted to build one master index that would allow me to perform the search locally against the generated index and then be able to pull up the specific record in the database table once I found what I was looking for.

dtSearch is a product that allows you to accomplish this goal. I downloaded it and installed it. Of course, I use it for many of my other searching needs as well. dtSearch has a variety of products. This post will focus on how you can index a database table so that you can perform lightning quick full-text searches.

How To Get Full-Text Search Capabilities in a Windows Azure SQL Database

Objectives

In this hands-on lab, you will learn:

1. How dtSearch can be used to create a full index over an Azure-hosted SQL Database table.
2. How to store the index locally on the client for super fast querying with fuzzy logic
3. How to identify and build the appropriate connection string (OLEDB) to enable the Visual Studio Sample Project application provided by dtSearch
4. How to perform full text searches and find the matching record in SQL Azure

To develop this example further, it is possible to extend the sample and retrieve matching records from dtSearch.

Prerequisites

The following is required to complete this hands-on lab:

5. A Windows Azure subscription
6. An evaluation version of dtSearch (http://www.dtsearch.com)

Setup

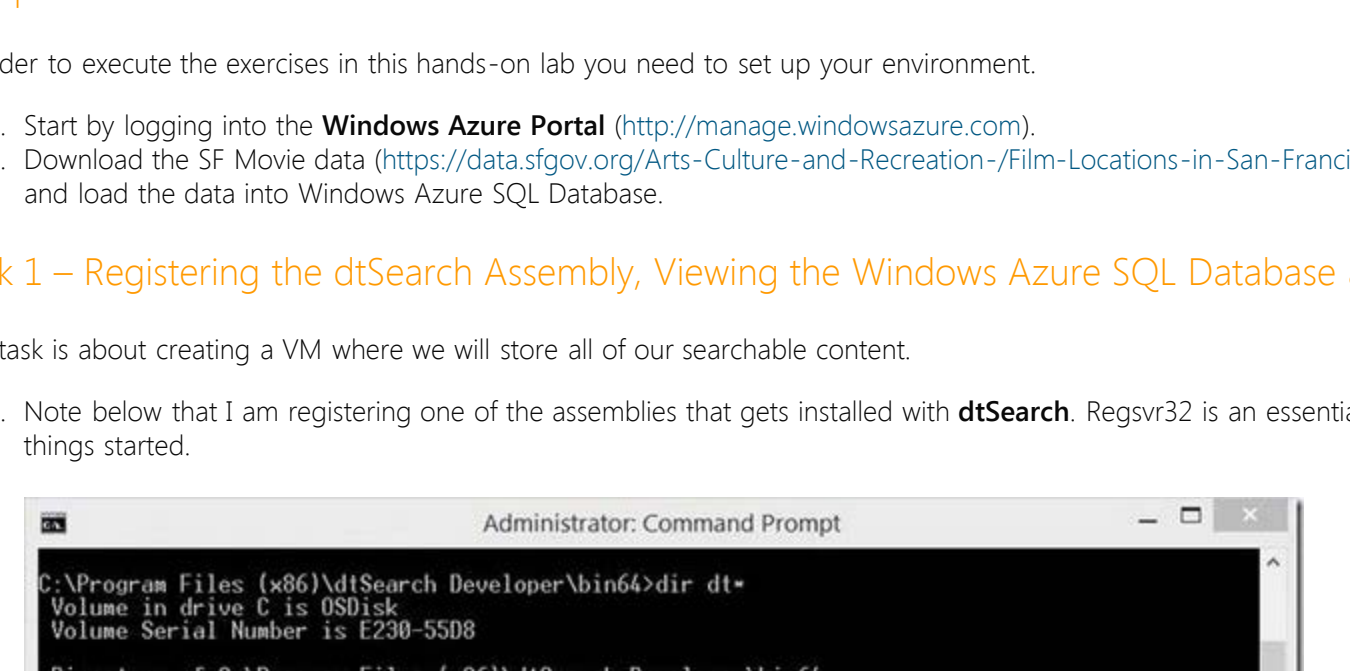
In order to execute the exercises in this hands-on lab you need to set up your environment.

7. Start by logging into the **Windows Azure Portal** (http://manage.windowsazure.com).
8. Download the SF Movie data (https://data.sfgov.org/Arts-Culture-and-Recreation-/Film-Locations-in-San-Francisco/yitu-d5am) and load the data into Windows Azure SQL Database.

Task 1 – Registering the dtSearch Assembly, Viewing the Windows Azure SQL Database at the Portal

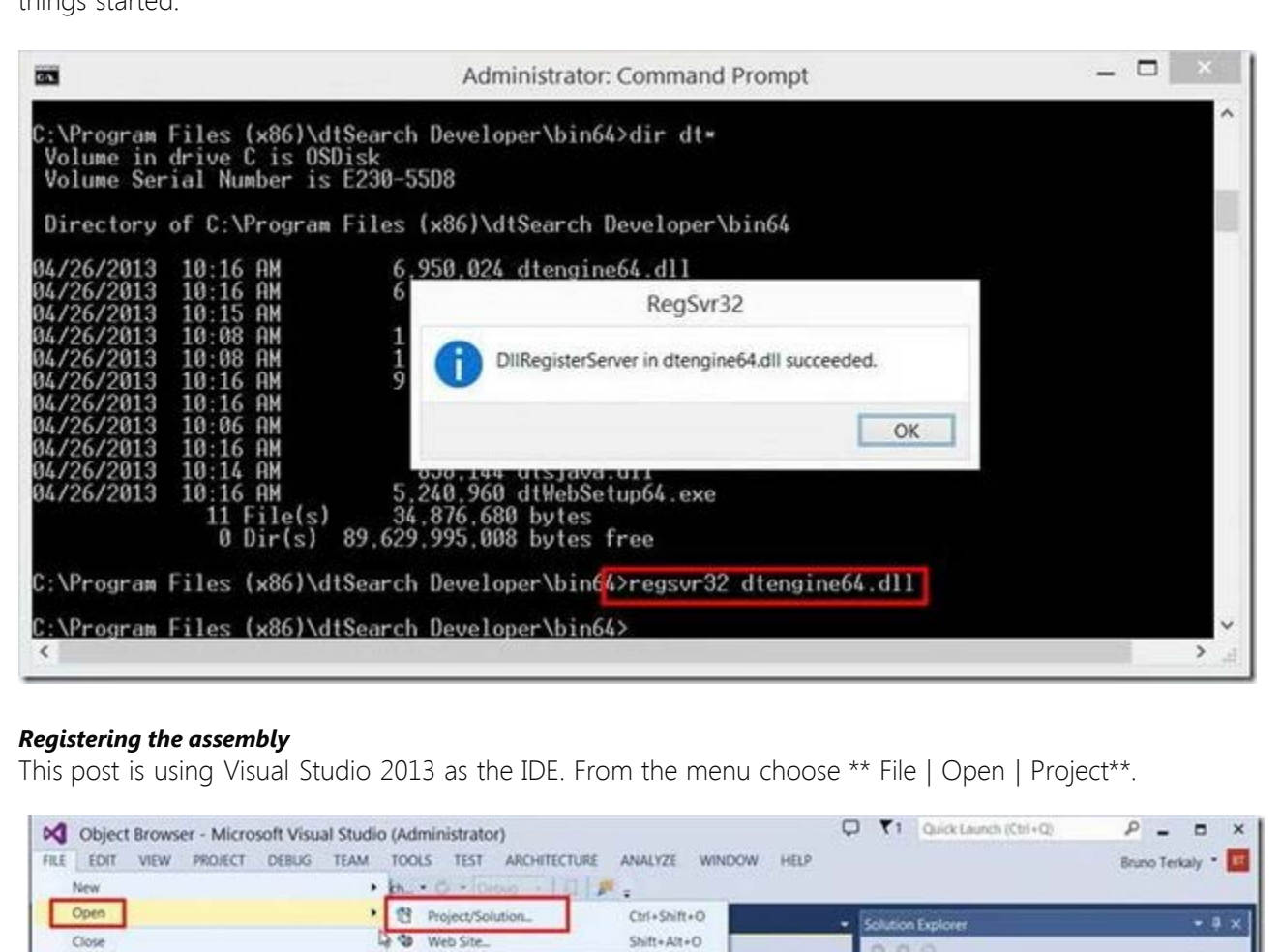
This task is about creating a VM where we will store all of our searchable content.

9. Note below that I am registering one of the assemblies that gets installed with **dtSearch**. Regsvr32 is an essential command to get things started.



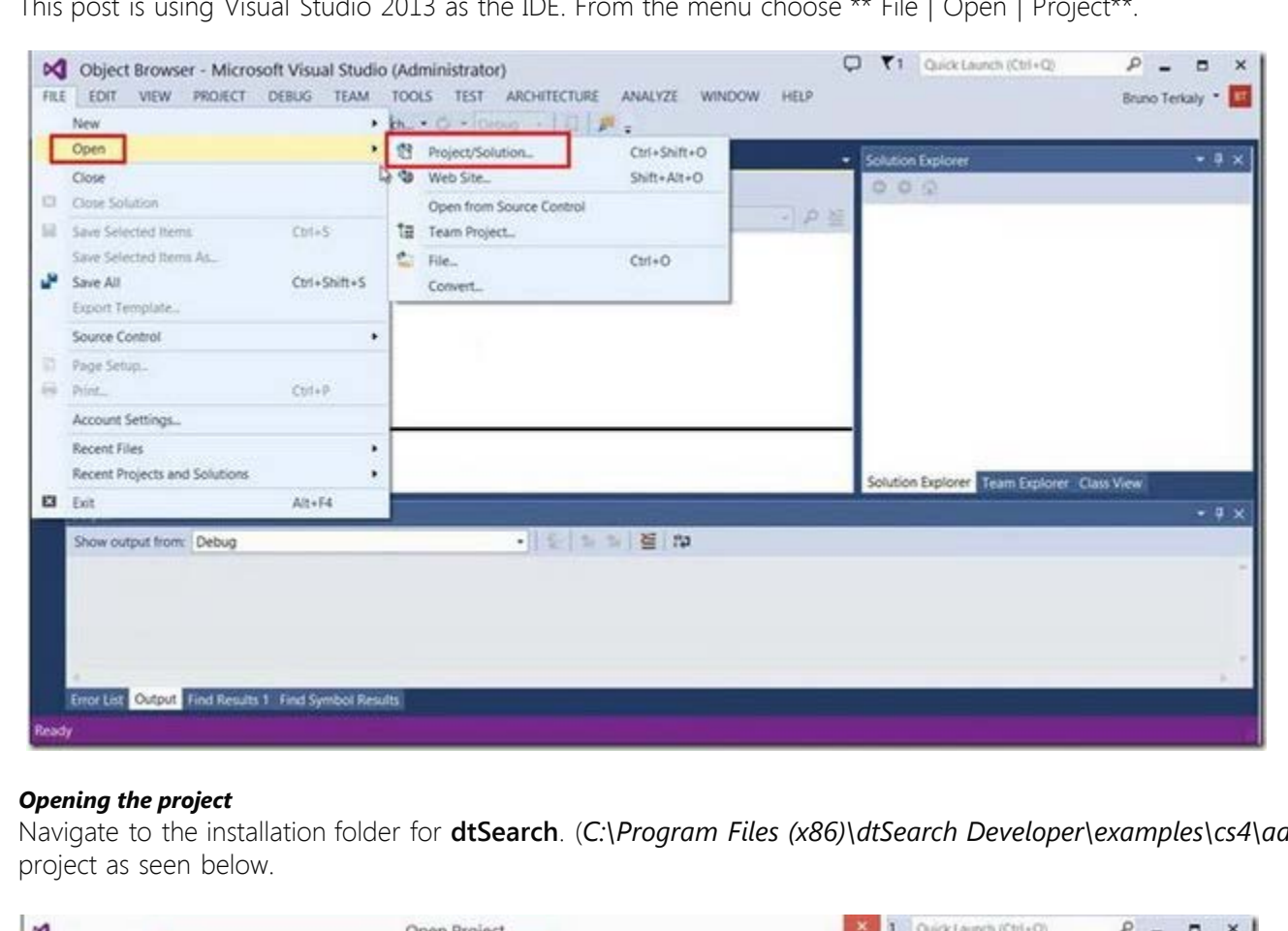
Registering the assembly

10. This post is using Visual Studio 2013 as the IDE. From the menu choose ** File | Open | Project**.



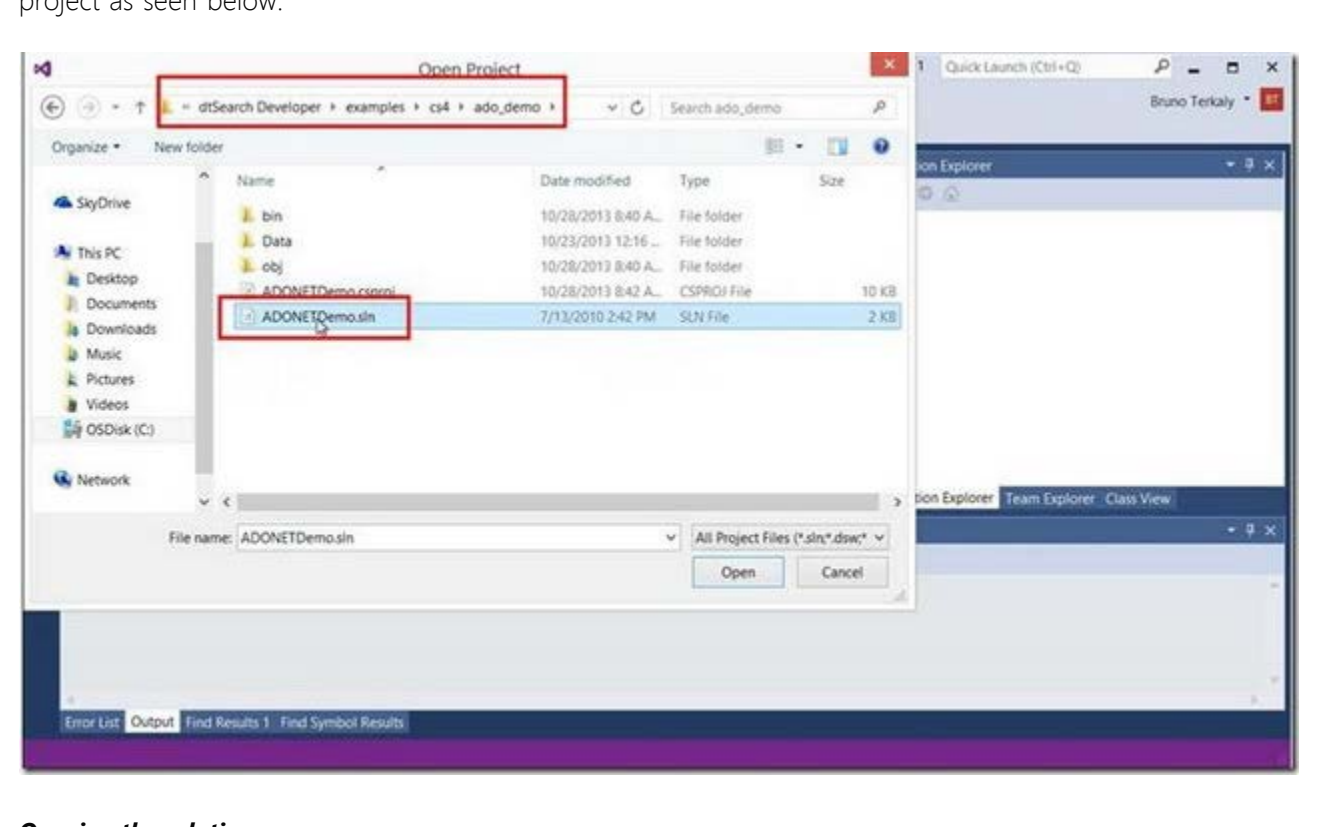
Opening the project

11. Navigate to the installation folder for **dtSearch**. (*C:\Program Files (x86)\dtSearch Developer\examples\cs4\ado_demo*). Open the project as seen below.



Opening the solution

12. Notice in the screen below that the database already exists. The database is called **SFmovies**. As mentioned previously, the data represents the movie filmed in San Francisco over the last several decades.

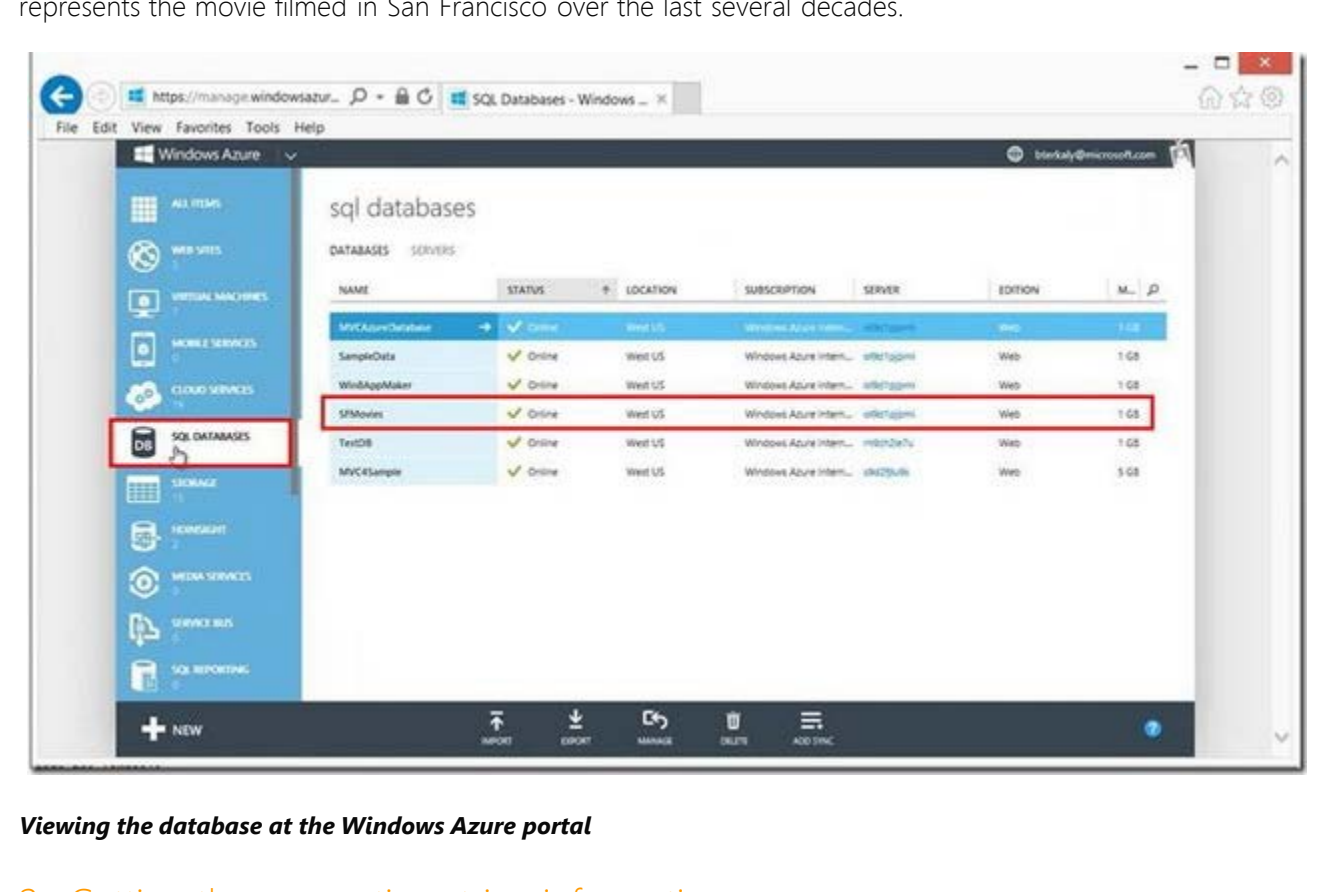


Viewing the database at the Windows Azure portal

Task 2– Getting the connection string information

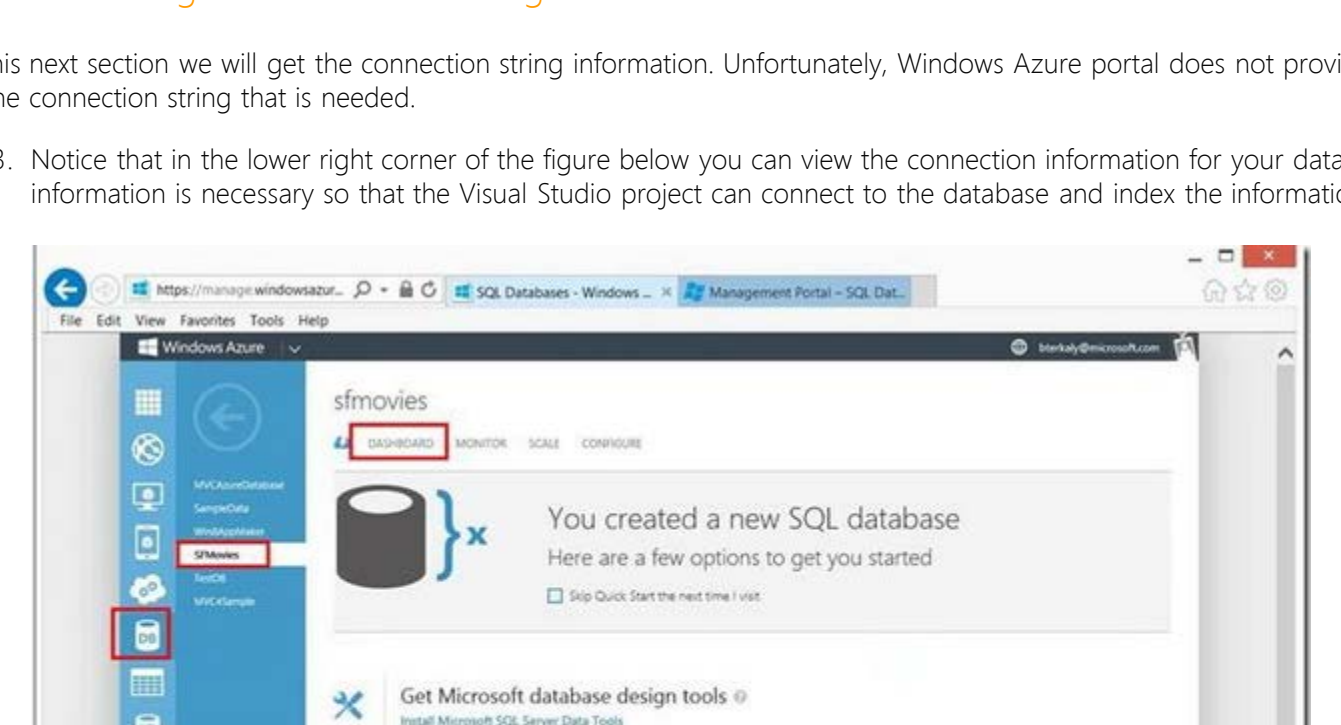
In this next section we will get the connection string information. Unfortunately, Windows Azure portal does not provide the exact format of the connection string that is needed.

13. Notice that in the lower right corner of the figure below you can view the connection information for your database. This information is necessary so that the Visual Studio project can connect to the database and index the information inside of it.



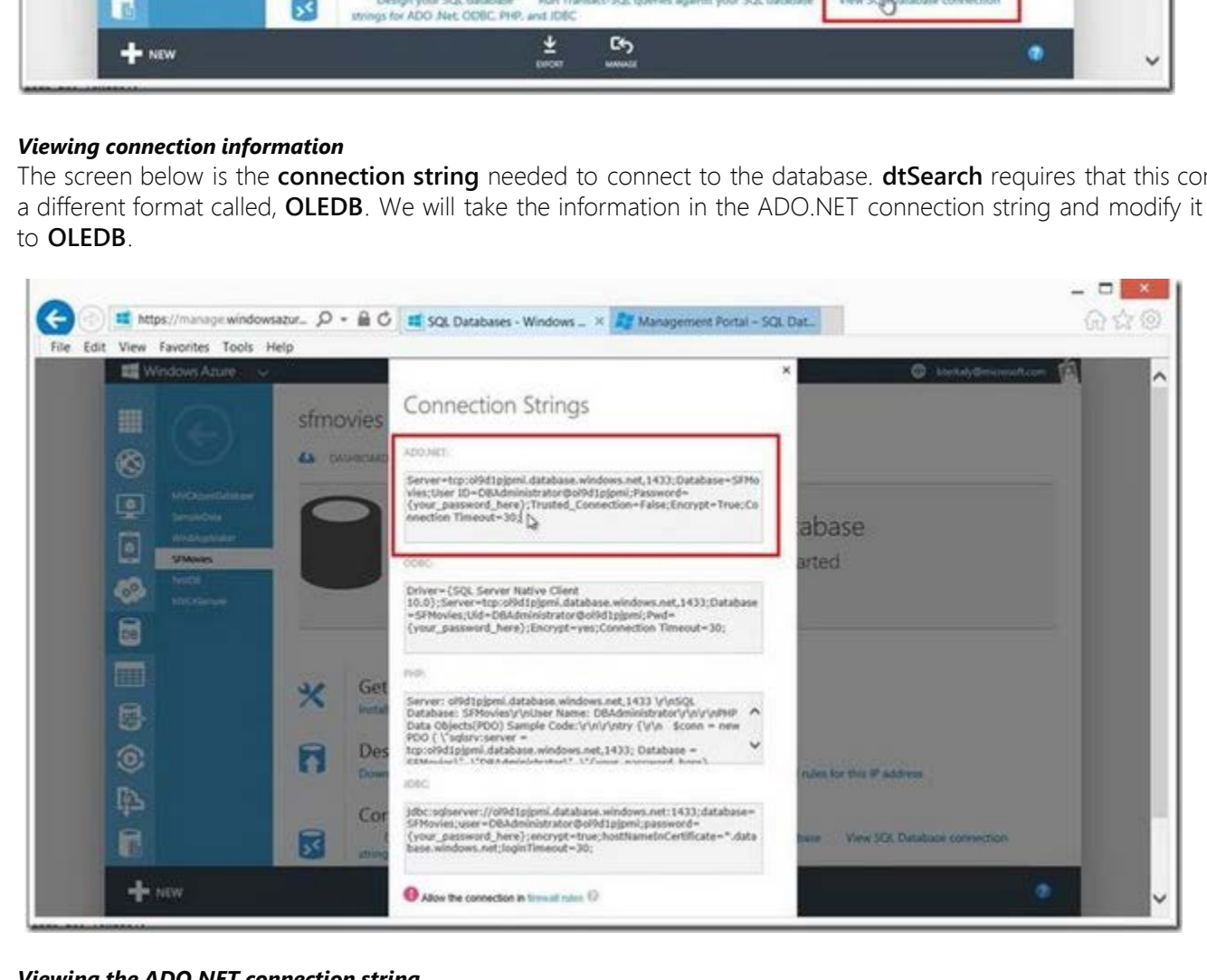
Viewing connection information

14. The screen below is the **connection string** needed to connect to the database. **dtSearch** requires that this connection string uses a different format called, **OLEDB**. We will take the information in the ADO.NET connection string and modify it to be conformant to **OLEDB**.



Viewing the ADO.NET connection string

15. Notice in the figure below the connection string has been altered to conform to the format required by **dtSearch**. This is the format required by **OLEDB**.



The reformatted connection string

16. The connection string needs to be one continuous line, as seen below. Your connection string will obviously differ, based on the database you are hosting in the Windows Azure cloud.

```
Provider=SQLNCLI11; Password=P@ssw0rd; User ID=DBAdministrator@ol9d1pjpmi; Initi
al Catalog=SFMovies; Data Source=tcp:ol9d1pjpmi.database.windows.net,1433;
```

The connection string

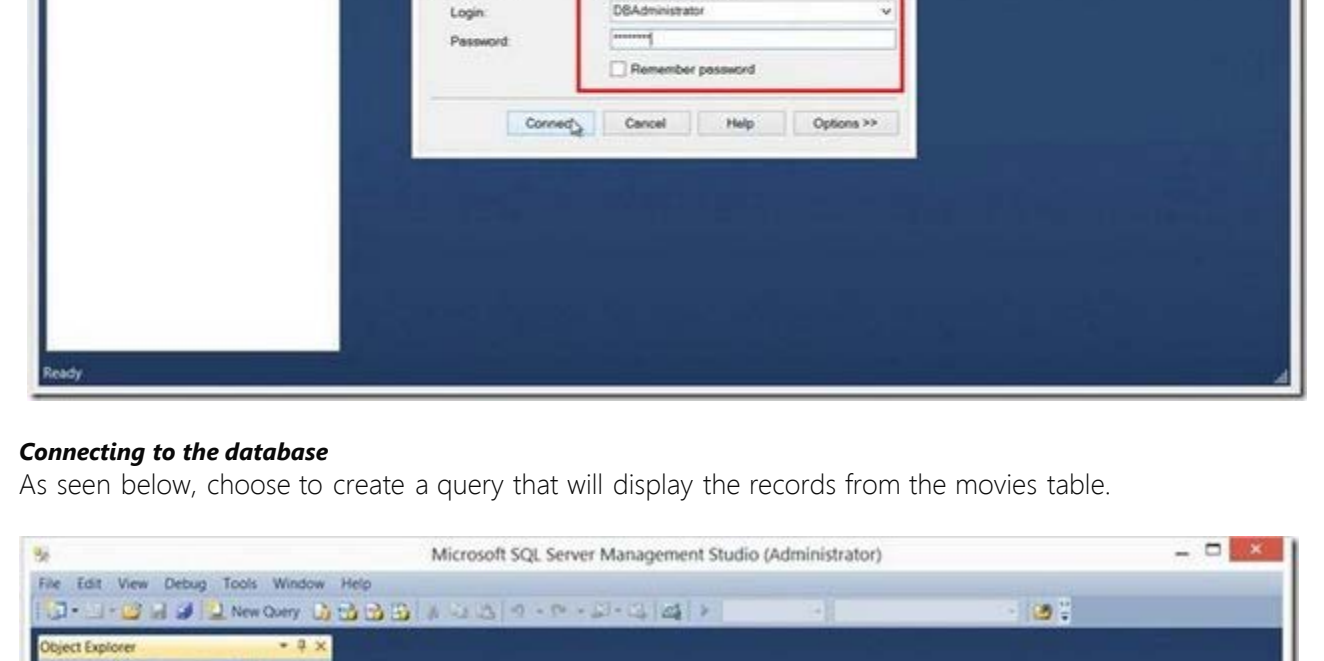
Task 3 – SQL Server Management Studio

We will use SQL Server Management Studio to view the database that we wish to index.

17. In order to test connectivity and view the actual data structures, we will start SQL Server Management Studio. From the Object Explorer window, click database engine.

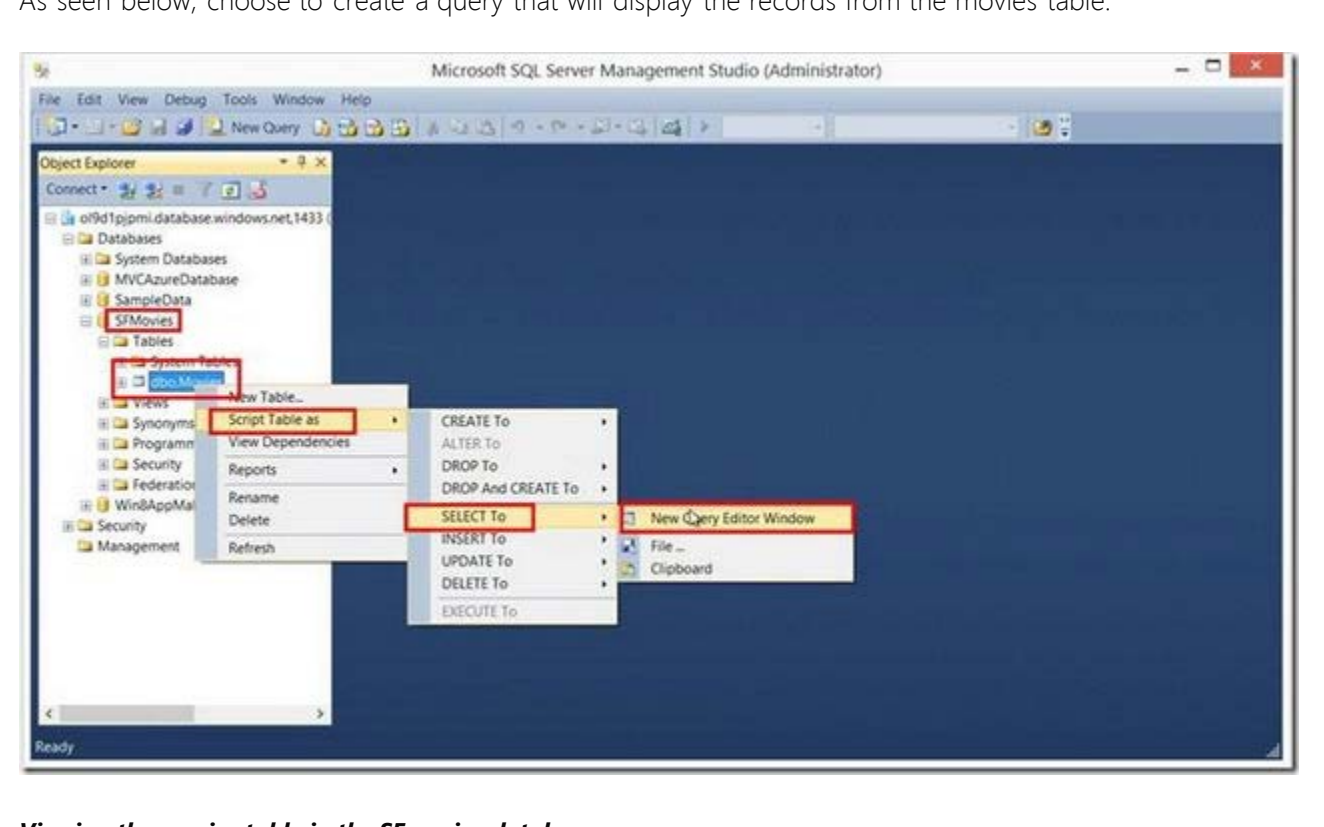
SQL Server Management Studio

18. The same information that was used for the connection string will work here within SQL Server Management Studio.



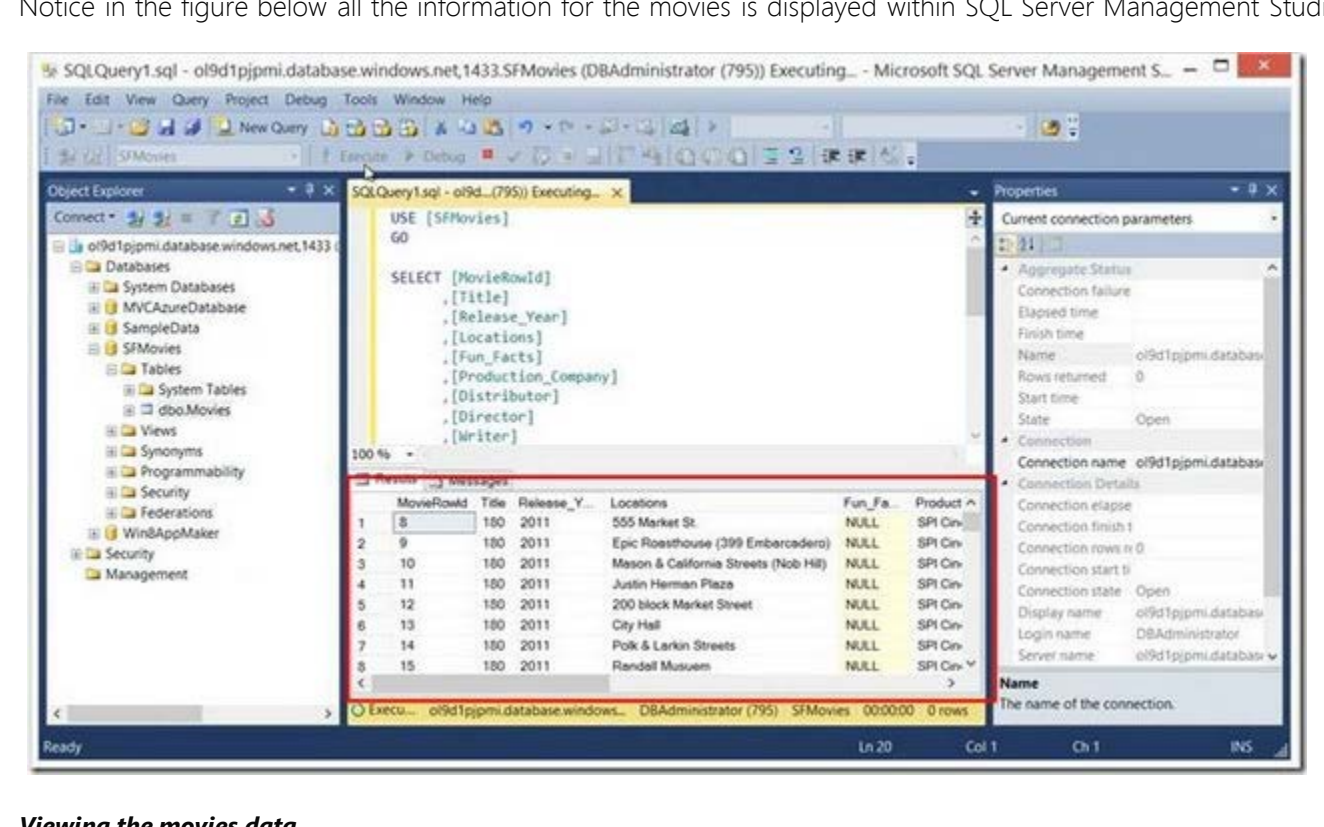
Connecting to the database

19. As seen below, choose to create a query that will display the records from the movies table.



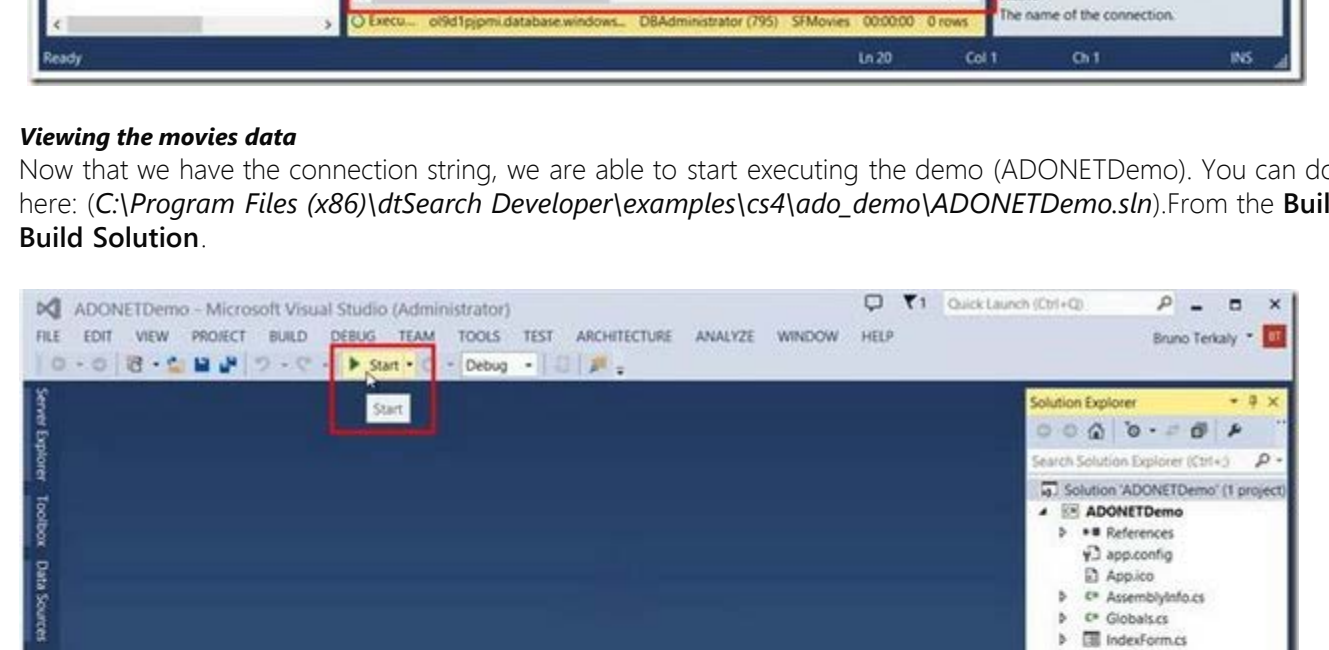
Viewing the movies table in the SFmovies database

20. Notice in the figure below all the information for the movies is displayed within SQL Server Management Studio.



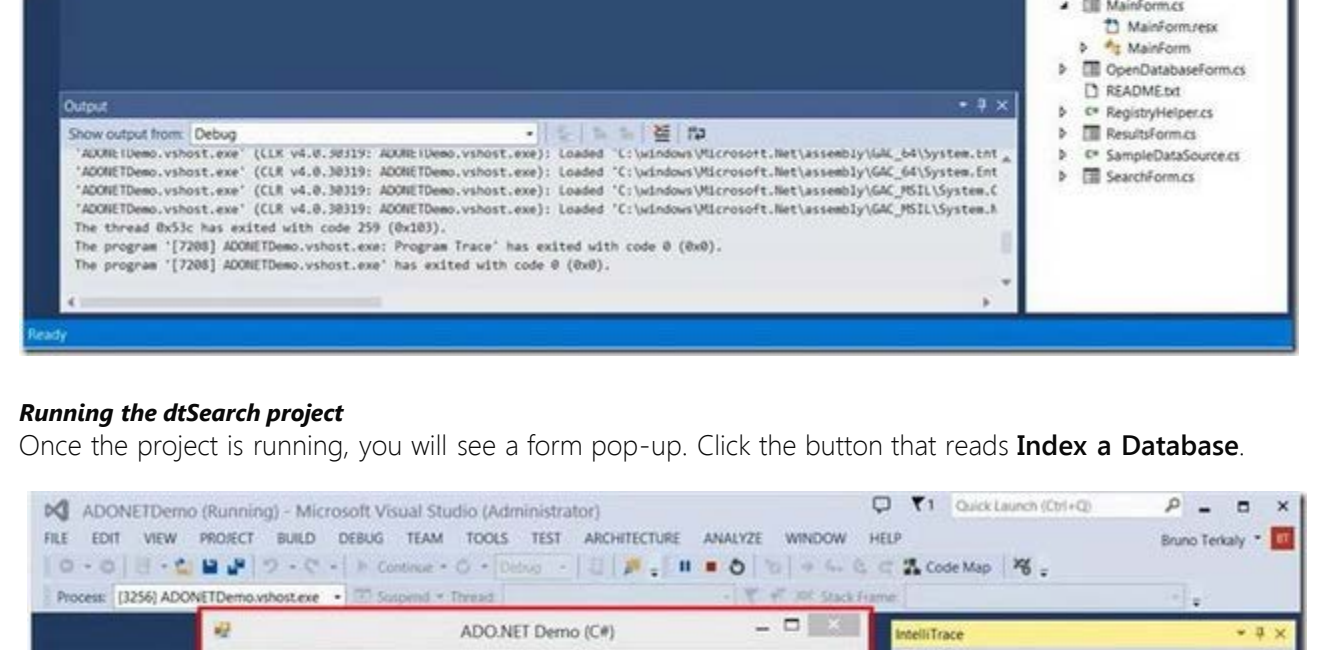
Viewing the movies data

21. Now that we have the connection string, we are able to start executing the demo (ADONETDemo). You can download this project here: (*C:\Program Files (x86)\dtSearch Developer\examples\cs4\ado_demo\ADONETDemo.sln*).From the **Build** menu you can **Build Solution**.



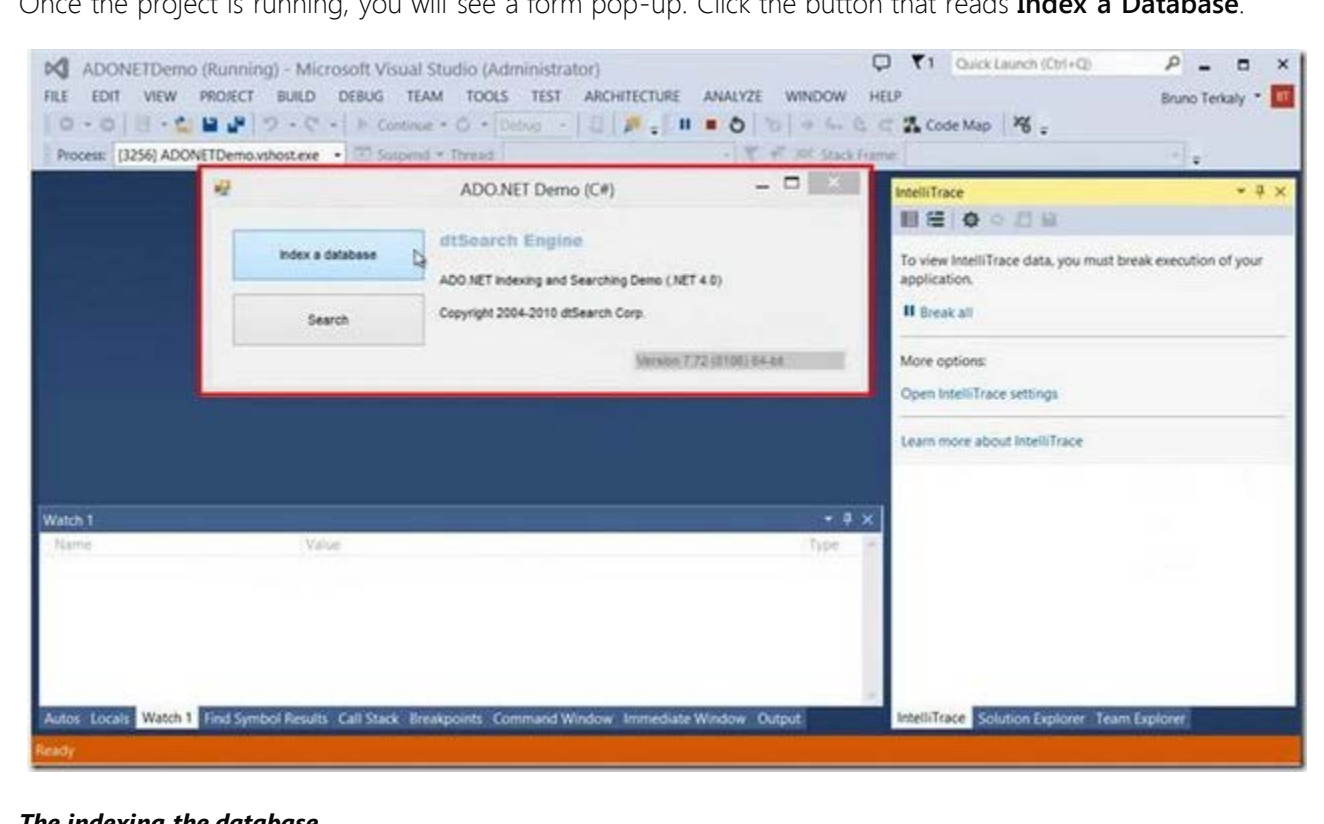
Running the dtSearch project

22. Once the project is running, you will see a form pop-up. Click the button that reads **Index a Database**.



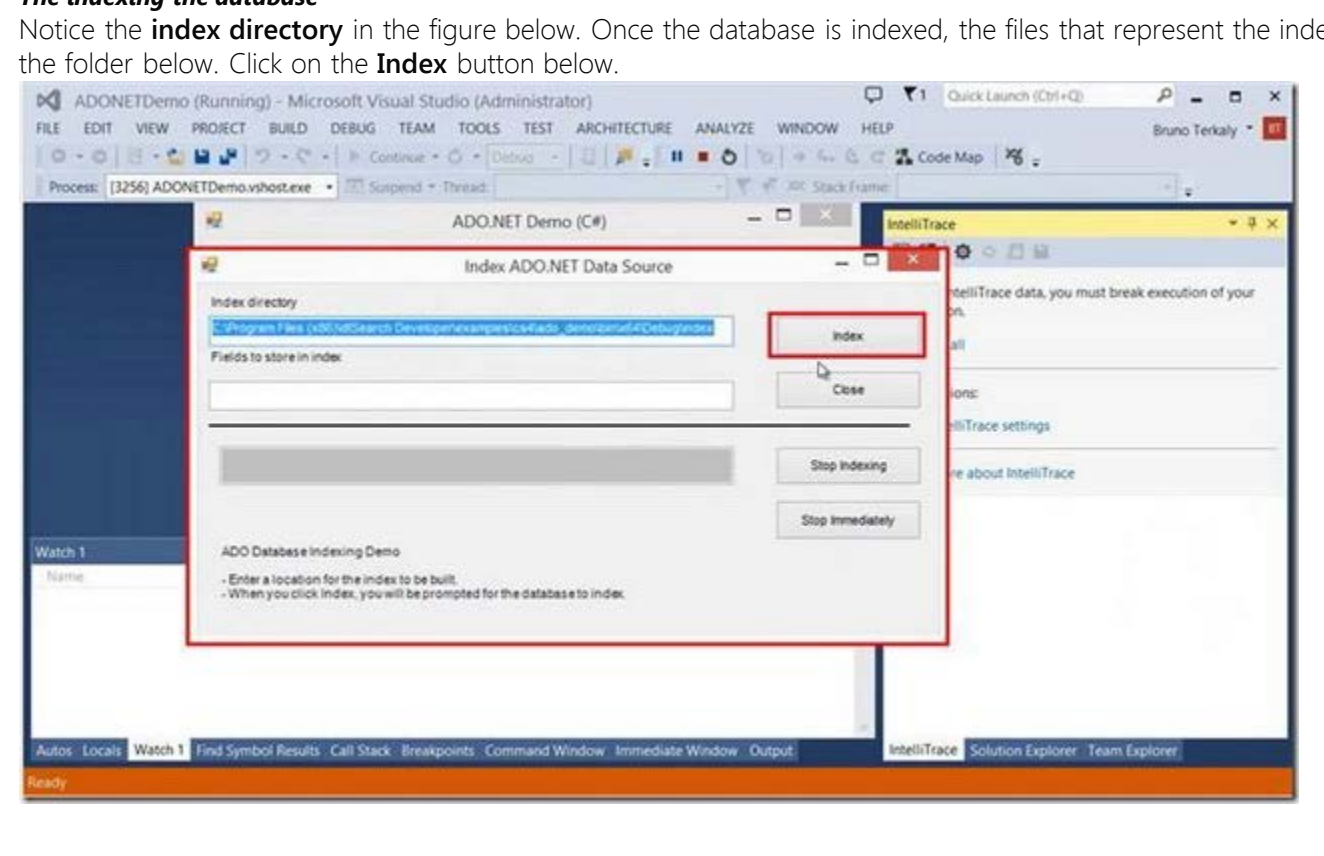
The indexing the database

23. Notice the **index directory** in the figure below. Once the database is indexed, the files that represent the index will be stored in the folder below. Click on the **Index** button below.



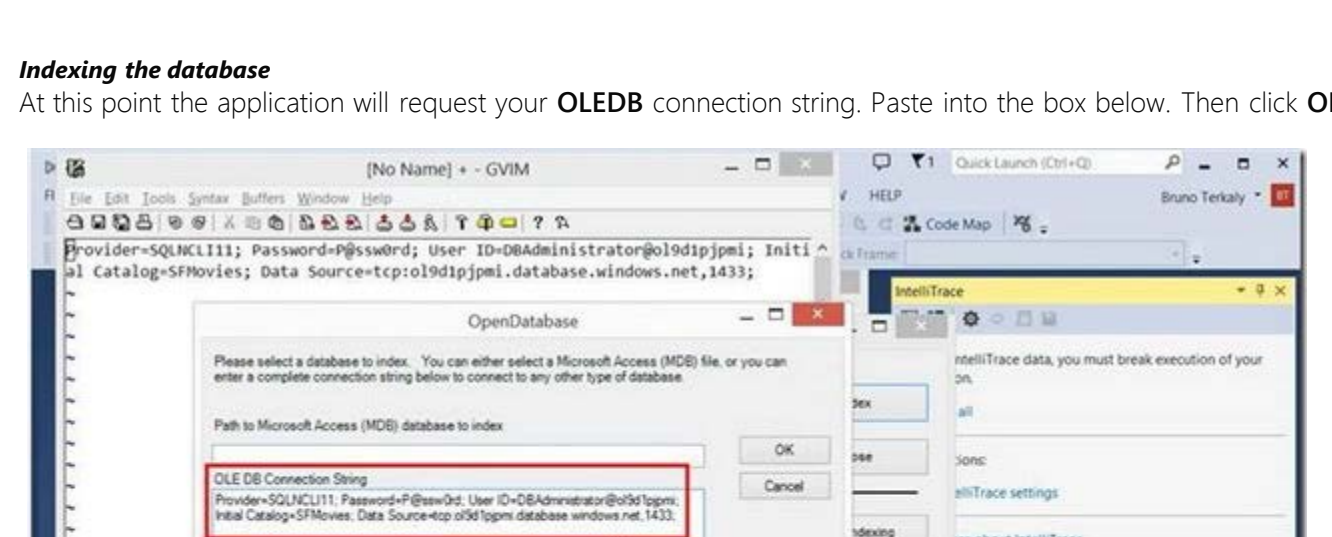
Indexing the database

24. At this point the application will request your **OLEDB** connection string. Paste into the box below. Then click **OK**



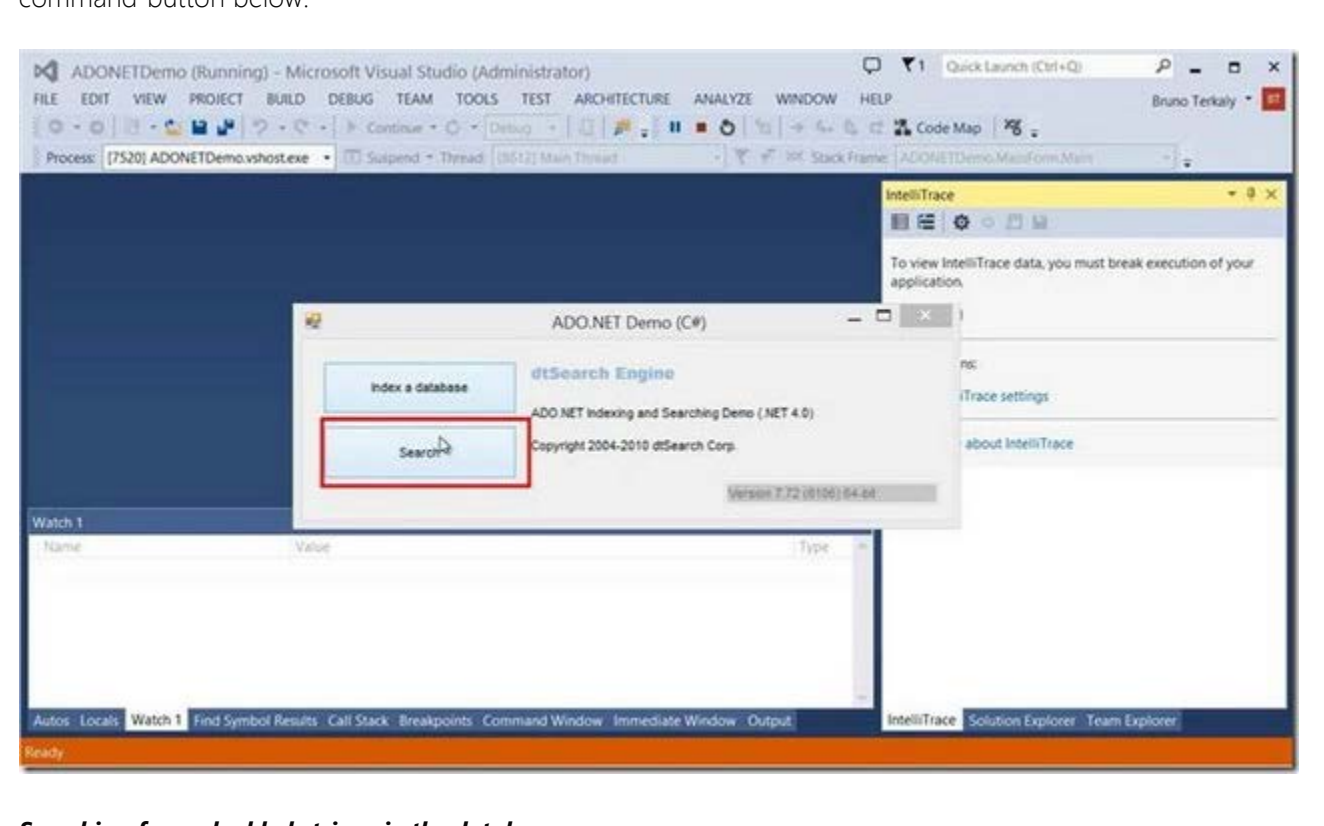
Entering the connection string

25. At this point, it might take a few moments for the indexing to complete. The lower red box represents the progress being made during the indexing process.



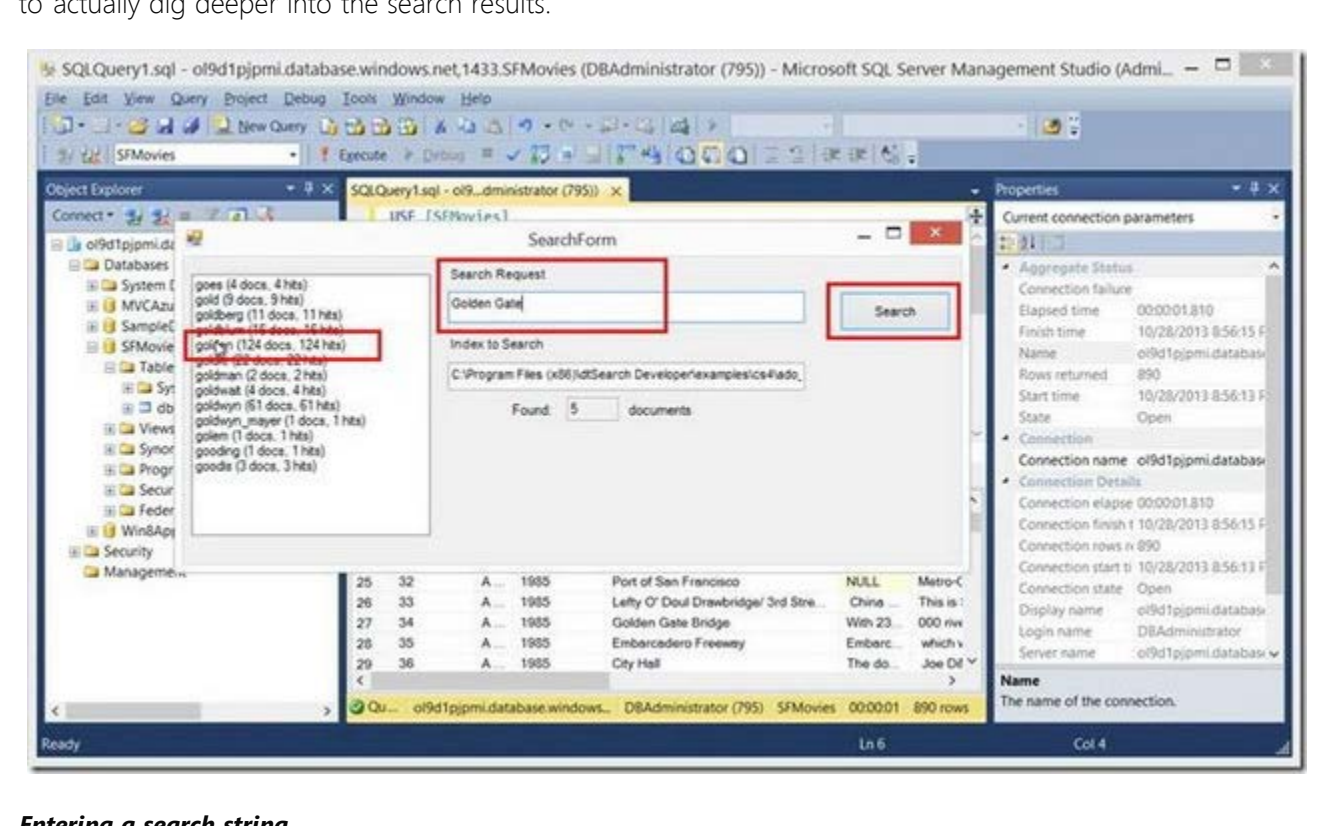
Indexing the database

26. In the folder indicated previously, you can see a list of index files that were generated during this process.



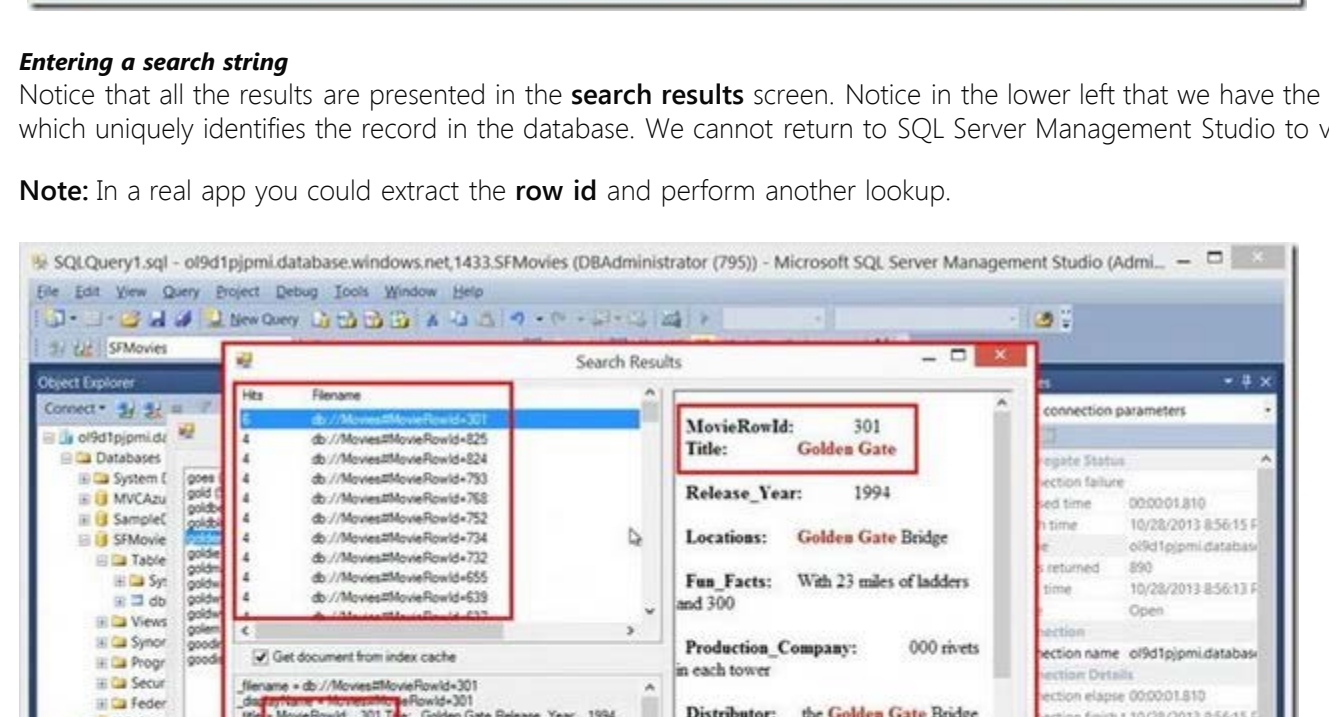
Viewing the generated indexes

27. Now that the indexing process is complete, we are ready to run the application and perform a search. Click on the search command button below.



Searching for embedded strings in the database

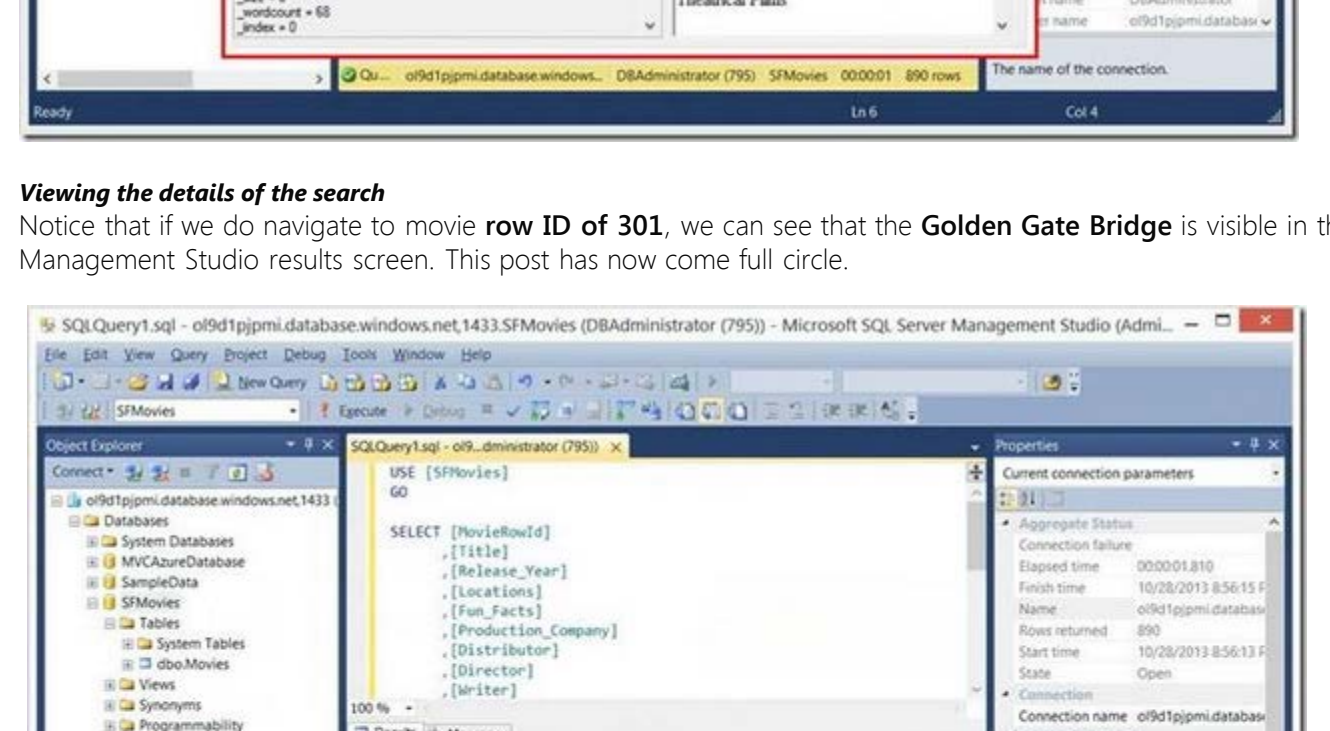
28. As you type in the **search string**, you will see the successfully found strings on the list box on the left. Click on the **search button** to actually dig deeper into the search results.



Entering a search string

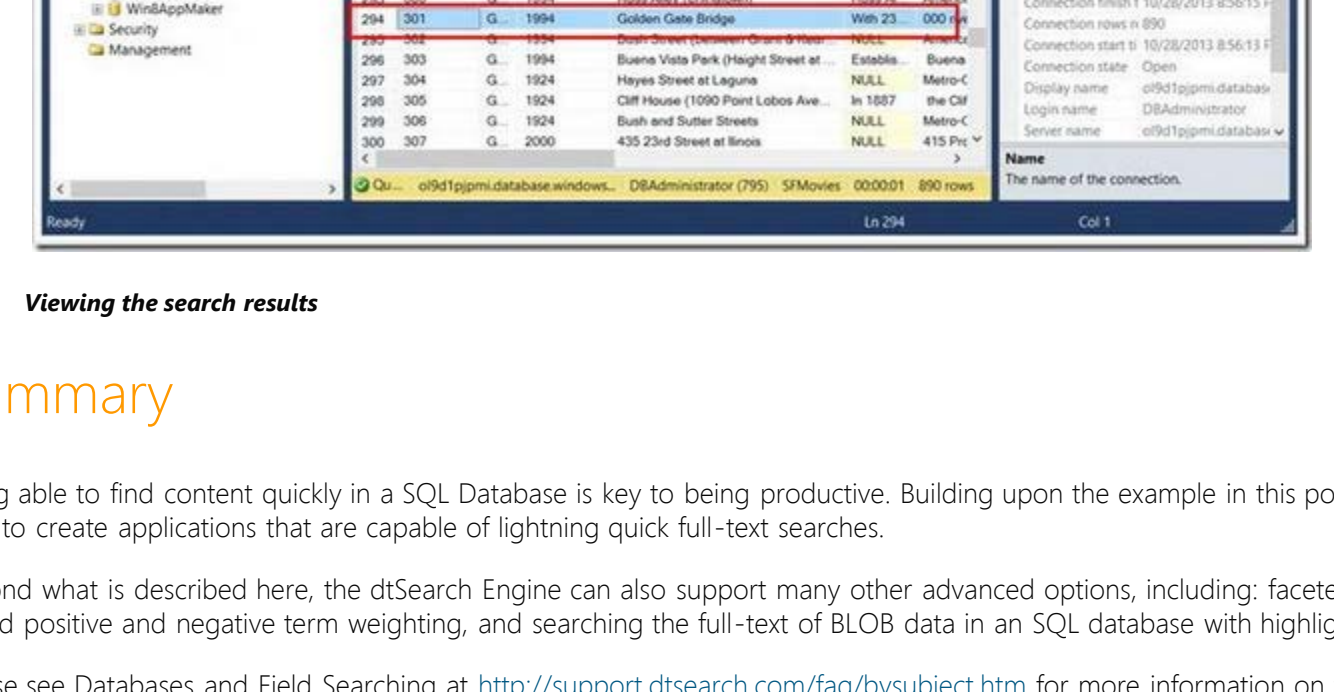
29. Notice that all the results are presented in the **search results** screen. Notice in the lower left that we have the movie **row ID**, which uniquely identifies the record in the database. We cannot return to SQL Server Management Studio to view this record.

Note: In a real app you could extract the **row id** and perform another lookup.



Viewing the details of the search

30. Notice that if we do navigate to movie **row ID of 301**, we can see that the **Golden Gate Bridge** is visible in the SQL Server Management Studio results screen. This post has now come full circle.



Viewing the search results

Summary

Being able to find content quickly in a SQL Database is key to being productive. Building upon the example in this post, you should be able to create applications that are capable of lightning quick full-text searches.

Beyond what is described here, the dtSearch Engine can also support many other advanced options, including: faceted search, fields-based positive and negative term weighting, and searching the full-text of BLOB data in an SQL database with highlighted hits.

Please see Databases and Field Searching at http://support.dtsearch.com/faq/dts0111.htm for more information on these advanced search options. Please also see http://www.dtsearch.com/PLF_Features_2.html for a general overview of dtSearch full-text search options. For an overview of what you can do generally programming with the dtSearch Engine, including supported data types for BLOB data, etc., please see http://www.dtsearch.com/PLF_engine_2.html and http://www.dtsearch.com/PLF_DocFilters.html

License

Share

About the Author

Bruno Terkaly

Canada

Starting with Turbo C and 8086 Assembler in the late 80s, Bruno has kept busy teaching and writing code in a multitude of platforms, languages, frameworks, SDKs, libraries, and APIs.

Bruno's depth of knowledge comes from years of experience in the field, where he can bring real-world knowledge and combine it with forward thinking that is required for his current role as a Developer Evangelist at Microsoft. Prior to evangelism, Bruno was a Microsoft Premier Field Engineer, helping customers in remote locations on a moment's notice to help with extreme troubleshooting scenarios, including problem isolation and correction, live and post-mortem debugging, on-the-fly application design and code reviews, performance tuning (IIS, SQL Server, .NET), application stability, porting / migration assistance, configuration management, pre-rollout testing and general development consulting.

As an evangelist, Bruno spends time writing code and giving live presentations on building cloud based applications, specifically using the Windows Azure Platform. He also takes a strong interest in Mobile Computing and is convinced that both mobile and cloud platforms, separately and together, are poised for huge growth over the next 10 years.

Bruno is very optimistic about the potential for new interactions with software using Kinect. The software industry has been languishing in aging approaches to human interactions and software. Kinect opens up an brave new world of possibilities to next generation software engineering.

Comments and Discussions

1 message has been posted for this article Visit **https://www.codeproject.com/Articles/679102/Windows-Azure-SQL-Database-Development-with-the-dt** to post and view comments on this article, or click **here** to get a print view with messages.