RogueWave SOFTWARE

Accelerating Great Code

USING SOURCEPRO DB WITH MICROSOFT AZURE SQL DATABASE

SourcePro DB was created long before cloud computing. So it may be a surprise to some that it that SourcePro DB can simplify your use of the Microsoft Azure SQL Database.

SourcePro is a complete enterprise C++ development platform that provides robust, reliable cross-platform C++ tools in areas of databases, networking, analysis, and fundamental C++ components. SourcePro DB, a part of SourcePro product suite, is a library of C++ classes featuring a high-level object-oriented cross-database abstraction over the database vendor native C APIs. It provides a high performance, consistent API encapsulating database-specific differences in behavior and syntax. The DB Access Module for Microsoft SQL server provides access to Microsoft SQL server using ODBC drivers.

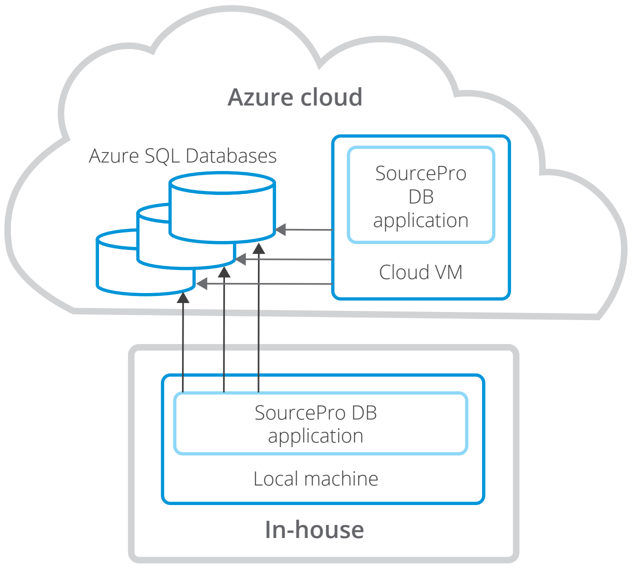
In this paper, we will describe using SourcePro DB while connecting to a Microsoft Azure SQL database. We will cover some minor limitations observed as compared to a locally installed SQL Server.

INTRODUCTION TO MICROSOFT AZURE SQL DATABASE

Microsoft Azure SQL Database is a relational database-as-a-service offering on Microsoft Azure cloud. Azure SQL Database is built on standardized hardware and software that is owned, hosted, and maintained by Microsoft. With Azure SQL Database, you can develop directly on the service using built-in features and functionality.[1]

With the ability to use ODBC drivers such as the SQL Server Native Client ODBC driver and the ODBC Driver for SQL Server, SourcePro DB applications can connect and work with Azure SQL Databases, just as they would with a locally installed SQL server instance. This enables existing applications written for in-house SQL server installations to be migrated to use Azure SQL Databases with few or no changes and provides all of the advantages that Azure SQL Databases offer, such as hardware and software maintenance and administration, high availability, disaster recovery, point-in-time restore, geo-restore, and geo-replication, to name a few.[1]

USING SOURCEPRO DB WITH AZURE SQL DATABASE



Setup

Azure SQL Database can be set up in Microsoft Azure Portal (https://portal.azure.com/) by creating a new SQL database under the category of Data + Storage. Besides providing database-specific settings, such as database name, database source, pricing tier, and collation, you will need to configure the Azure SQL Server in which the database will reside. An existing Azure SQL Server can be selected if it is version V12, or a new server can be created.

[1] "Understanding Azure SQL Database and SQL Server in Azure VMs", assessed June 2016.

To create a new server, you will need to provide the server name, admin credentials, and the physical location.[2]

In the question of whether to create a V12 server, select YES. The V12 version has significant advantages over the earlier V11 version, including increased compatibility with SQL Server 2014, higher performance, support for elastic database pools, and security enhancements, amongst others.[3] V11 version did not support tables without clustered indexes, which is a major limitation in functionality as compared to standalone SQL servers.

Configuration

Once the Azure SQL Database is set up, you need to ensure that your choice of SQL server ODBC driver on the client machine is able to connect to the Azure SQL Database.

- In order for the Azure SQL Server to accept a connection request coming from the IP address of the client machine, the client IP address needs to be added to its firewall settings. In the Microsoft Azure Portal, select the Azure SQL Server used to host the Azure SQL Database. Select the firewall settings and add a rule for accepting connections from the client IP address. Alternatively, you can select a range of IP addresses to accept connections from.[2]
- ODBC DSN setup on the client machine is same as it would be for a locally installed SQL server. The name of the host is the name of the Azure SQL Server and the name of the database is the name of the Azure SQL Database. At the end of the ODBC DSN setup, there is an option to test the database connection.

Once a successful connection has been established, the new ODBC DSN can be supplied as the server name parameter to the SourcePro DB API `RWDBManager::database()`, while using the DB access module for Microsoft SQL Server. Here is an example:

```
RWDBDatabase db = RWDBManager::database("MS_SQL", "<ODBC DSN name>",
    "<user name>", "<password>", "<Azure SQL Database Name>");
```

The `RWDBDatabase` object can then be used in the same way as you would a local SQL server instance.

Limitations

An Azure SQL Database is not able to access other databases on the same Azure SQL Server. For this reason, any attempt to access objects on another database is not supported, and object references such as *<database name>.<schema name>.<object name>* are not supported. For the same reason, global temporary objects are not supported, as they are created in a temporary database. These operations result in following errors:

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Reference to database and/or server
name in '<database name>.<schema name>.<object name>' is not supported in this version
of SQL Server.

[Microsoft][ODBC SQL Server Driver][SQL Server]Global temp objects are not supported
in this version of SQL Server.
```

[2] "SQL Database tutorial: Create a SQL database in minutes using the Azure portal", assessed June 2016.

[3] "What's new in SQL Database V12", assessed June 2016.

Summary

Using DB Access Module for Microsoft SQL Server, new SourcePro DB applications can be written connecting to the Azure SQL Databases hosted in Azure cloud with the same easy-to-use, object-oriented, high-performance API. Existing applications connecting to local SQL Server instances can easily be migrated to connect to Azure SQL Databases with little or no change to the code, while taking advantage of all the features that cloud hosted databases offer.

USING THE ELASTIC DATABASE FEATURE

After configuring the Azure SQL database, we now look at the basic setup for elastic database feature. There are no changes required to SourcePro DB application in order to connect to an SQL Azure database with elastic feature enabled.

Setup elastic database server

Create a SQL database on Azure portal by following the instructions here:
https://azure.microsoft.com/en-us/documentation/articles/sql-database-get-started/

Note: Please ensure that the IP address of your machine is in the list of allowed IP addresses.

Setup shards

Setup shards by following instructions here:
https://azure.microsoft.com/en-us/documentation/articles/sql-database-elastic-scale-get-started/#Getting-started-with-elastic-database-tools

The sample application in the article sets up shards and also loads it with sample tables. We will use this sample in for our next step. So, execute step 3, a few times so that data is inserted in the shards.

Setup tables on the Azure database server

Open SQL server management studio (or command line via sqlcmd) and connect to your elastic database server. Run the attached script to create a database, configure an external data source and setup external tables. The tables on this server provide the interface to execute queries on the shard.

Test the shards

Query the 'Customers' table using SourcePro DB

```
RWDBTable customerTable = myDbase.table("Customers");
RWDBConnection connection = myDbase.connection();
RWDBSelector select = myDbase.selector();
select << customerTable["CustomerId"];
RWDBReader rdr = select.reader(connection);
int customerId;
while(rdr()) {
    rdr >> customerId;
}
```

Example for external tables

This sections includes script to setup external tables.

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'pwd';

CREATE DATABASE SCOPED CREDENTIAL SMMCred
WITH IDENTITY = 'user',
SECRET = 'pwd';

CREATE EXTERNAL DATA SOURCE ElasticDataSource WITH
                 (TYPE = SHARD_MAP_MANAGER,
                 LOCATION = 'server name',
                 DATABASE_NAME = 'ElasticScaleStarterKit_ShardMapManagerDb',
                 CREDENTIAL = SMMCred,
                 SHARD_MAP_NAME = 'CustomerIDShardMap');

CREATE EXTERNAL TABLE [Regions] (
        [RegionId] [int] NOT NULL,
        [Name] [nvarchar](256) NOT NULL
    )
WITH
(
    DATA_SOURCE = ElasticDataSource,
    DISTRIBUTION = REPLICATED
);

CREATE EXTERNAL TABLE [Products] (
        [ProductId] [int] NOT NULL,
        [Name] [nvarchar](256) NOT NULL
    )
WITH
(
    DATA_SOURCE = ElasticDataSource,
    DISTRIBUTION = REPLICATED
);

CREATE EXTERNAL TABLE [Customers] (
        [CustomerId] [int] NOT NULL,
        [Name] [nvarchar](256) NOT NULL,
        [RegionId] [int] NOT NULL
    )
WITH
(
    DATA_SOURCE = ElasticDataSource,
    DISTRIBUTION = SHARDED(CustomerId)
);
```

```
CREATE EXTERNAL TABLE [Orders](
        [CustomerId] [int] NOT NULL,
        [OrderId] [int] NOT NULL,
        [OrderDate] [datetime] NOT NULL,
        [ProductId] [int] NOT NULL
    )
WITH
(
    DATA_SOURCE = ElasticDataSource,
    DISTRIBUTION = SHARDED(CustomerId)
);
```

Accelerating Great Code