## Database

The initial schema/hydration scripts were executed vs. a local version of Sql Server, Database name = **Projects**. [*Workstation/Server Name for development was* “**lemur2**.”]

The SQL Scripts will be bundled (for the purpose of this exercise) in the **/\_Deployment/DatabaseScripts** Folder. All “USE [Projects]” t-sql statements should be updated to reflect the deployment destination Database, as well as all fully qualified table/sproc names in the sql scripts.

*\* Of Note*, for development and pre-production efforts, I have not had to create new SQL Logins frequently, and when I have it’s almost always been an initial manual process vs. delivering a scripted SQL User creation. Also, in the last ~ 10 years, I’ve had the luxury of actual DBAs to review my scripts before they were queued for Staging/Production, so apologies in advance if my SSMS Right-Click *DB\_User* script is not bullet-proof. Ensure the Projects DB User “appcore\_user” has **db\_owner** role membership to execute the stored procedures.

DB Scripts:

1. **DB\_Schema\_Script.sql** (original schema/hydration script, includes INSERT statements for 3 addresses)
2. **DB\_User.sql**  Creates SQL Login for consuming application to access DB.
   1. **User**: appcore\_user
   2. **Password**: appcore1234
3. dbo.spGetAddress.sql *stored procedure*
4. dbo.spSaveAddress.sql *stored procedure*

## Visual Studio

All libraries are included in the CoreApp solution. General structure:

* \_Deployment folder for DB Scripts and deployment/design notes
* Interfaces folder for the ResourceAccess and Manager interfaces
* Tests folder for unit tests and a quick Proof Of Concept MVC validator that addresses can be retrieved by ID, and that new/existing addresses can be saved to the DB
* The AddressManager library, to coordinate data retrieval and handle any business logic
* DataAccess library - leveraging SqlClient to pull from a specific data store. IN thoery, this could be refactored/updated to pull data from Oracle, Mongo, XML files on disc,etc
* ResourceAccess library. Points at specific DataAccess methods for data retrieval, packages requested data in a ResourceResponse object container to provide additional information to calling Manager (**IsSuccess()** indicates whether an error was encountered requesting data)
* Models Library - Business Object(s) - in this exercise, Address and State. Also contains the ResourceResponse class/interface, as both ResourceAccess and Manager libraries will contain a reference to Models to know what objects to pass back to the requesting application.

## Verifying DB Connectivity:

Given Local/Test/Stage/Production Builds and Deployments, the DB Connectivity will need to be accessed via config files for each environment. My “coding exercise” approach to make this configurable vs. hard-coded was to identify the Server Name (“DataSource”) and target Database Name (“databaseName”) in the appSettings node of the config file.

The “**SmokeTest/ValidationClient**” MVC project contains these appSettings values so that the app will be able to target the appropriate DB. These values are read in to form the DB Connection String in **DataAccess.Address.GetDbConnString(**). The SQL User Login username and password is actually hard-coded in this method.