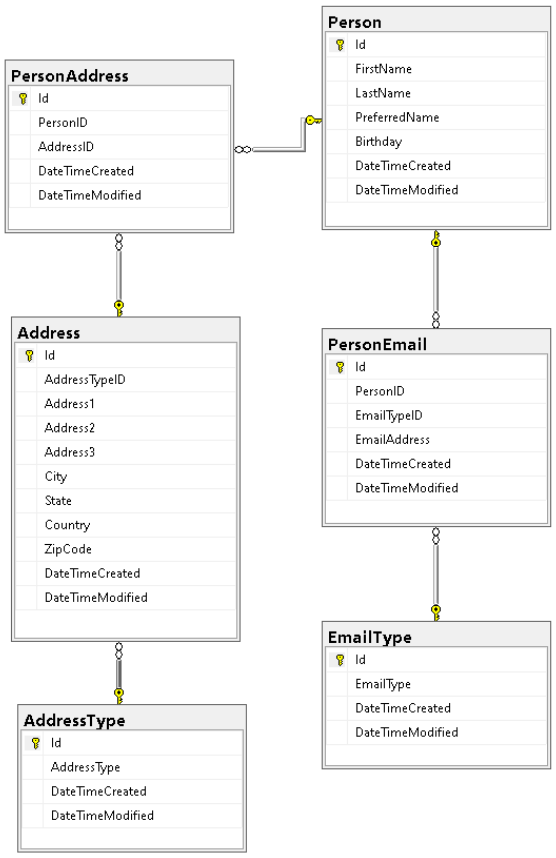
I am creating a basic CRUD Address Book application using SQL Database and C#. I am looking for design assistance to see if I am designing the database correctly and coding it in a “best practice”. I plan on using this application as the foundation of a larger application, in the future, thus I need this to be designed well and will scale for large numbers of records and users accessing the application.

Below is my database design diagram from SSMS. Does it make sense? Note, Foreign Key’s do not show up in the diagram but are defined where appropriate. E.G. PersonID, AddressID, etc.



Below are my C# model classes. Note, the Person class contains a list of PersonAddress and PersonEmail that are populated by my SQL Connection class when inserting or fetching records. All other model classes mimic the tables exactly. I use stored procedures to handle the CRUD operations.

```
namespace iAddressBookLibrary.Models
{
    public class Person
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string PreferredName { get; set; }
        public DateTime Birthday { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeModified { get; set; }

        public List<PersonAddress> PersonAddresses { get; set; } = new List<PersonAddress>();
        public List<PersonEmail> PersonEmail { get; set; } = new List<PersonEmail>();
        public string FullName
```

```
        {
            get => $"{FirstName} '{PreferredName}' {LastName}";
        }

        public override string ToString() => FullName;
    }
}

namespace iAddressBookLibrary.Models
{
    public class PersonAddress
    {
        public int Id { get; set; }
        public int PersonID { get; set; }
        public int AddressID { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeModified { get; set; }
    }
}

namespace iAddressBookLibrary.Models
{
    public class Address
    {
        public int Id { get; set; }
        public int AddressTypeID { get; set; }
        public string Address1 { get; set; }
        public string Address2 { get; set; }
        public string Address3 { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string Country { get; set; }
        public string ZipCode { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeModified { get; set; }
    }
}

namespace iAddressBookLibrary.Models
{
    public class AddressType
    {
        public int Id { get; set; }
        public string TypeName { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeModified { get; set; }

    }
}

namespace iAddressBookLibrary.Models
{
    public class PersonEmail
    {
        public int Id { get; set; }
        public int PersonID { get; set; }
        public int EmailTypeID { get; set; }
        public string EmailAddress { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeModified { get; set; }
    }
}

namespace iAddressBookLibrary.Models
{
    public class EmailType
    {
        public int Id { get; set; }
        public string TypeName { get; set; }
```

```
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeModified { get; set; }
    }
}
```

LIBRARY SQL CLASS

```
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;

using Dapper;

using iAddressBookLibrary.Models;

namespace iAddressBookLibrary.DataAccess
{
#pragma warning disable IDE0063 // Use simple 'using' statement
    public class SQLConnector : IDataConnection
    {
        private const string db = "iAddressBookDB";

        //  ToDo:  Generate database create script
        //  ToDo:  Generate script to add emailType and AddressType Values


        public void CreatePerson( Person personModel, List<Address> addressModels,
List<PersonEmail> personEmailModels )
        {
            //  ToDo:  Create Transaction here?

            //  Create the Person record first and get the returned ID from SQL
            InsertPerson( personModel );

            foreach( PersonEmail aPersonEmail in personEmailModels )
            {
                //  Create the PersonEmail record
                aPersonEmail.PersonID = personModel.Id;
                InsertPersonEmail( aPersonEmail );
            }

            //  Create the address record
            foreach( Address anAddress in addressModels )
            {
                InsertAddress( anAddress );
                //  Create the linkage record of address to person...
                PersonAddress aPersonAddress = new PersonAddress
                {
                    PersonID = personModel.Id,
                    AddressID = anAddress.Id
                };
                InsertPersonAddress( aPersonAddress );
            };
        }
        public void InsertPerson( Person model )
        {
            using( IDbConnection connection = new SqlConnection( GlobalConfig.ConectionString( db
) ) )
            {
                DynamicParameters p = new DynamicParameters();
                p.Add( "@FirstName", model.FirstName );
                p.Add( "@LastName", model.LastName );
                p.Add( "@PreferredName", model.PreferredName );
                p.Add( "@Birthday", model.Birthday );
                p.Add( "@Id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output );
                _ = connection.Execute( "dbo.spPerson_Insert", p, commandType:
CommandType.StoredProcedure );
                model.Id = p.Get<int>( "@Id" );
            }
        }
```

```
        public void InsertPersonEmail( PersonEmail model )
        {
            using( IDbConnection connection = new SqlConnection( GlobalConfig.ConectionString( db
) ) )
            {
                DynamicParameters p = new DynamicParameters();
                p.Add( "@PersonID", model.PersonID );
                p.Add( "@EmailTypeID", model.EmailTypeID );
                p.Add( "@EmailAddress", model.EmailAddress );
                p.Add( "@Id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output );
                _ = connection.Execute( "dbo.spPersonEmail_Insert", p, commandType:
CommandType.StoredProcedure );
                model.Id = p.Get<int>( "@Id" );
            }
        }

        public void InsertAddress( Address model )
        {
            using( IDbConnection connection = new SqlConnection( GlobalConfig.ConectionString( db
) ) )
            {
                DynamicParameters p = new DynamicParameters();
                p.Add( "@AddressTypeID", model.AddressTypeID );
                p.Add( "@Address1", model.Address1 );
                p.Add( "@Address2", model.Address2 );
                p.Add( "@Address3", model.Address3 );
                p.Add( "@City", model.City );
                p.Add( "@State", model.State );
                p.Add( "@ZipCode", model.ZipCode );
                p.Add( "@Country", model.Country );
                p.Add( "@Id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output );
                _ = connection.Execute( "dbo.spAddress_Insert", p, commandType:
CommandType.StoredProcedure );
                model.Id = p.Get<int>( "@Id" );
            }
        }

        public void InsertPersonAddress( PersonAddress model )
        {
            using( IDbConnection connection = new SqlConnection( GlobalConfig.ConectionString( db
) ) )
            {
                DynamicParameters p = new DynamicParameters();
                p.Add( "@PersonID", model.PersonID );
                p.Add( "@AddressID", model.AddressID );
                p.Add( "@Id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output );
                _ = connection.Execute( "dbo.spPersonAddress_Insert", p, commandType:
CommandType.StoredProcedure );
                model.Id = p.Get<int>( "@Id" );
            }
        }

        public List<Person> GetPerson_All()
        {
            List<Person> output;

            using( IDbConnection connection = new SqlConnection( GlobalConfig.ConectionString( db
) ) )
            {
                output = connection.Query<Person>( "dbo.spPerson_GetAll", commandType:
CommandType.StoredProcedure).ToList();
                foreach( Person person in output )
                {
                    DynamicParameters param = new DynamicParameters();
                    param.Add( "@PersonID", person.Id );
                    person.PersonAddresses = connection.Query<PersonAddress>(
"dbo.spPersonAddress_GetByPersonID", param, commandType: CommandType.StoredProcedure ).ToList();
                    person.PersonEmail = connection.Query<PersonEmail>(
"dbo.spPersonEmail_GetByPersonID", param, commandType: CommandType.StoredProcedure ).ToList();
                }
            }
```

```
            return output;
        }
    }
}
```

TEST PROGRAM

```
using System.Collections.Generic;
using iAddressBookLibrary.DataAccess;
using iAddressBookLibrary.Models;

namespace iAddressBookTestConsole
{
    class Program
    {
        static void Main( string[] args )
        {
            Person aPerson = new Person
            {
                FirstName = "Steve",
                LastName = "Rieger",
                PreferredName = "Steven",
            };

            Address anAddress = new Address
            {
                AddressTypeID = 1,  //  1 = Home
                Address1 = "1144 Devonshire Rd",
                City = "Buffalo Grove",
                State = "IL",
                ZipCode = "60089",
                Country = "USA"
            };

            PersonEmail aPersonEmail = new PersonEmail
            {
                EmailTypeID = 1,  //  1 = Personal
                EmailAddress = "srieger@riegergroup",
            };

            List<Address> listOfAddresses = new List<Address>();
            List<PersonEmail> listOfPersonEmails = new List<PersonEmail>();
            listOfAddresses.Add( anAddress );
            listOfPersonEmails.Add( aPersonEmail );
            SQLConnector connection = new SQLConnector();
            connection.CreatePerson( aPerson, listOfAddresses, listOfPersonEmails );
        }
    }
}
```

I only have some basic insert and get methods created so far but did not want to go too far down the rabbit hole if I am not following best practices. The application, when finished, will allow basic CRUD operations on Person's. Each person could have 0 to many addresses and 0 to many email addresses. Does this make sense the way I designed the database? Does the code make sense? Do you think this will scale for large numbers of users? Any thoughts on using Transactions when creating Records? Will that slow things down?