# Assignment 4 Relational Databases and Proper Data Types

In this assignment, you will take what you did in the last three assignments and enhance the functionality. You will be turning your single table in a database into a full blown relational database model and be adding some new functionality to your form and admin pages.

Before starting on this application read about the following topics

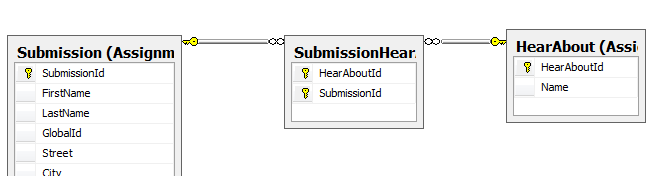
* T-SQL Identities - <http://www.simple-talk.com/sql/t-sql-programming/identity-columns/>  
  <http://en.wikipedia.org/wiki/Identity_column>
* T-SQL Data Types - <http://msdn.microsoft.com/en-us/library/ms187752.aspx>
* Differences between char data types - <http://stackoverflow.com/questions/176514/what-is-the-difference-between-char-nchar-varchar-and-nvarchar-in-mssql>
* Link (Junction) tables - <http://en.wikipedia.org/wiki/Link_Table>
* Relational Databases - <http://en.wikipedia.org/wiki/Relational_database>
* Relationship Types - <http://sqlrelationship.com/>
* SQL Update Statement - <http://www.w3schools.com/sql/sql_update.asp>
* SQL Delete Statement - <http://www.w3schools.com/sql/sql_delete.asp>
* Repeater Control - <http://msdn.microsoft.com/en-us/library/cc295084.aspx>
* T-SQL Stored Procedures - <http://msdn.microsoft.com/en-us/library/aa174792%28v=sql.80%29.aspx>
* SCOPE\_IDENTITY() - <http://msdn.microsoft.com/en-us/library/ms190315.aspx>
* General SQL Information - <http://sqlzoo.net/>

Additionally, there is a ZIP file containing a project in the Assignments folder of the team site entitled “Assignment4Supplemental”. This will give you an example of all the advanced functionality you will be doing in this assignment.

### Correct Data types

Now that you have read about data types, you probably have realized that your submission table has some incorrect data types. You should rebuild your submission table, so that it has correct data types. None of your fields should have fixed lengths and they **do not** need to accept Unicode characters. Anything that is storing a Boolean value, should now be using the **bit** data type. Also, you will need to add a datetime field for SubmissionDateTime, which will need to be populated with the date/time of the submission.

### New Tables

Up until now the “hear about” section of your form has been entirely static. Now we want to be able to add new options to the section. We want these sections to come out of the database, so you will need to add a table to store these values. You will also need to create a link table that allows your submission to have a one to many relationship with your hear about table. In the end your database structure should look something like this.

### Stored Procedures

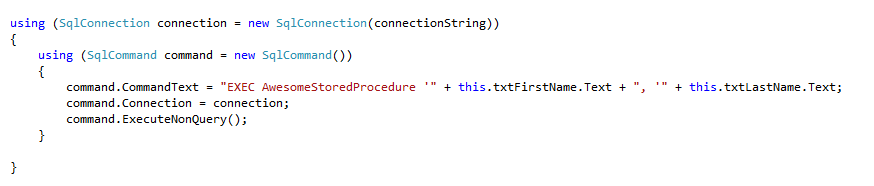
Instead of using straight INSERT, UPDATE, and DELETE SQL commands inside the C# code-behind of your application you will need to use stored procedures. You should be able to create a stored procedure for each operation you need to do against the database. For the insert stored procedure you will need to return the SCOPE\_IDENTITY for the new row in order to supply that value to your stored procedure for inserting into the link table between submissions and your hear about table.

### Parameterized Queries

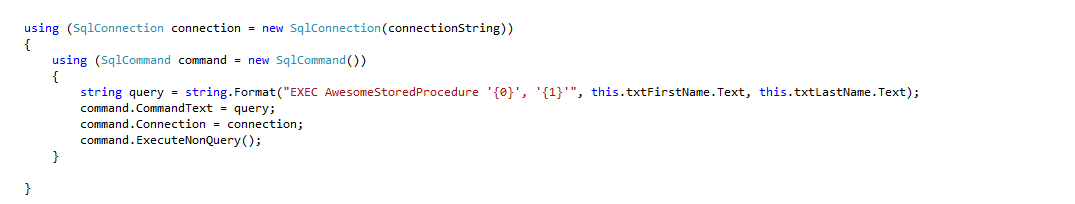
Now that you have done some level of SQL work you will need to start using SQL queries in a more “secure” fashion. One of the number one vulnerabilities in web sites in SQL injection, which is a very interesting topic that we highly recommend reading about. The basic nature of a SQL injection vulnerability is that a user can submit data to your query that allows them to execute SQL commands bypassing your security. The way we fight this on the ASP.NET world is by using parameterized queries.

Below are some examples of injectable non-parameterized code and examples of how to use parameterized queries to prevent injection. It is important to note, that it is still possible to write a stored procedure that is injectable even with parameterized queries, but assuming that your stored procedure is written securely it should not be a problem.

Injectable non parameterized query from the C# code behind



Another way that is still injectable and not parameterized



Finally a proper parameterized query



All SQL calls you make should not only be stored procedures, but for any that you need to pass parameters in they MUST use the SQLCommand Parameters collection demonstrated above.

In the previous assignment we had you using SQL Data Source in order to populate your GridView. This is no longer the case in Assignment 4. All SQL must be done in the code behind using SqlConnection, SqlCommand, and SqlDataReader. Please take some time reading about the following methods located on SqlCommand and have an understanding of what each do ExecuteReader(), ExecuteNonQuery(), and ExecuteScaler(). If everything is done correctly you should use one of each of these inside of this assignment.

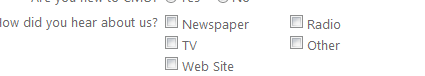
### Default.aspx

Your form will now need to use a repeater control to display the checkboxes now. In your code behind you will need to pull that data back and submit an entry into your link table for each checkbox selected.

For example my SQL table looks like this



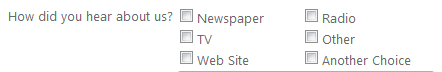
My site uses a repeater control that then generates this



I add the option “Another Choice”



The result of the added row would appear like this



In your submission button, you will need to account for the link table you created before. Your table should be setup in such a manner that the two columns that exist in the table are both foreign keys. One key pointing to the identity in your Submission table and on key pointing to the identity in your HeardAbout table. In order to get these two identities you will need to build your submission and insert it into your table using your stored procedure first. When you execute the SqlCommand that contains your stored procedure for inserting the submission, you should be returning the SCOPE\_IDENTITY(). That will be the identity for the submission. The second identity will come from your repeater control, you will need to loop through your repeater and find the identity from your HeardFrom table for all checkboxes that were checked. When you find a checkbox that is checked, you will need to execute a SqlCommand that runs a stored procedure that will insert into your link table.

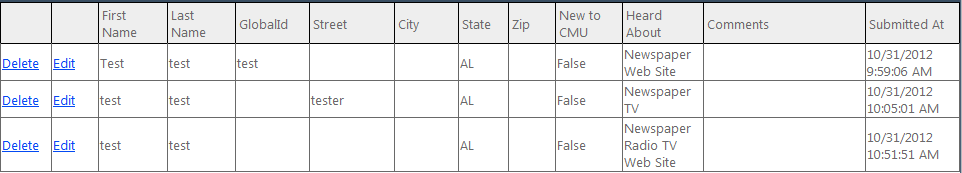
The order of operation MUST be

1. Insert submission return identity
2. Find a single checked box in the repeater and insert a record into the link table

### Admin.aspx

You will need to add the ability for users to delete and update entries. To do this, you will need to learn how to use a button or link button inside your gridview. You will also need some way of passing the identity for your submission to an update form and a delete stored procedure. This is going to be a difficult task that you may struggle with. This is to be expected as this is a fairly complex operation compared to what you have done so far. You should attempt deleting rows before updating rows since that is the easier of the two to accomplish.

Here is a screenshot of my final grid.



You will notice that the Heard About column has all options the user selected. It also has an edit and delete button. The delete button uses the ItemCommand event and the Edit button uses the EditCommand. For a <asp:ButtonColumn there is a command name property that will allow you to specify a specific command for the button.

