**ViSo-Nice ERD**

Thank you for using ViSo-Nice ERD. I hope it serves you well.

**Purpose**

The purpose of this application is to provide a visual representation of a database, where the relations are not necessarily registered as constraints, and to manage your database and provide a code base scripting for general queries.

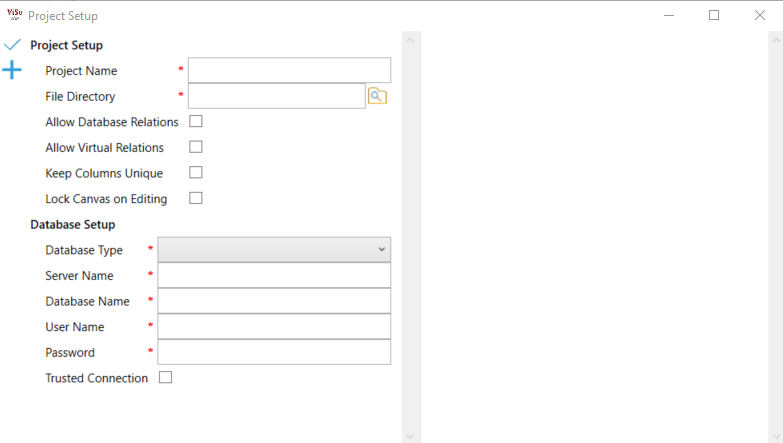
**This Document**

This document will guide you through the application in a step by step manner. Functionality will be discussed from a menu item perspective, irrespective of being a application menu or context menu.

**Getting Started**

Before you can work on a project you will need to create a new project. You will see that use was made of the old fashion menu items, and this is with the purpose of providing the user with as large as posable working space once the project is taking form.

To create a new project, from the menu, select File -> New Project. This will open the Project Setup screen where the project can be defined.



**Project Name**

First you will have to provide a project name. The project name will be used as a file prefix for all your project files. For this reason, use Alpha Numeric characters only (Spaces are allowed).

**File Directory**

Next you must provide a File Directory. This is the directory where all the project files will be stored. If you are working on a team, you may choose path on a file server. Note that if you would select the ‘Lock Canvas on Editing’ option the directory will be monitored by a ‘FileSystemWatcher’ and your own system securities will be applied, and when you are using some sort of source control there may be some issues surrounding securities.

**Allow Database Relations**

If this option is selected, the application will add a ‘Database Relation’ object to the left of the canvas screen, that can be dragged onto the working canvas. This type of relation will build a database relation on Forward Engineering, that forms part of your database, and database rules apply. The relation is displayed on the canvas as a light blue line.

**Allow Virtual Relations**

If this option is selected, the application will add a ’Virtual Relation’ object to the left of the canvas screen, that can be dragged onto the working canvas. This type of relation is a free relation that is only available in the model and allows for relations to be created without any impact on the database. The relation is displayed on the canvas as a light green line.

**Keep Columns Unique**

If this option is selected the application will validate columns by name (Case Sensitive) against their data types and will not update the database if there are discreteness where column data types differ. This is handy where columns carry a certain interpretation, and usage, especially where sub systems may use these columns and expect a certain datatype when ever the column is used irrespective of table, such as in generic reporting solutions build for custom integrations.

**Lock Canvas on Editing**

This option is selected where a team of developers will work on the same database at the same time, and one does not want to override the others work. The canvas will be locked for other developers the moment a user changes the canvas by adding a table or relation, or editing either of the two, or by removing either of the two. The lock will be released when the user that made the changes saves the project.

Other users using the system will then be able to access the canvas. Note that it will take several seconds before changes reflect in the other user’s application after the lock was removed.

**Database Setup**

For now, the application only supports MS SQL, and I hope to add support for other database types as soon as posable.

Is assume that the user will be familiar with the fields in this section.

This connection will be known as you ‘Default’. The Default will serve as your main development database.

****

**Add Alternative Connection ( )**

By selecting this option, you can add alternative connections to the model. Note that alternative connections do not require a username or password. This is handy when you are working in an environment where developers do not have access to a certain database, and the systems administrator would like to make use of an alternative database, such as a production database.

When the alternative database is selected the user will be prompt for the username and password, and the application will retain the information for the duration of the session but will not save the connections values.

****

**Accept ( )**

This option will close the screen and setup the application for use. Note that this will not save you project and you will have to select the ‘Save Project’ option from the menu under File -> Save Project.

**Getting the Tables**

From the File Menu select Database -> Get DB Tables. Note that if you have defined alternative connection the menu item will list all the available option. For the first run select the ‘Default’ if multiple connections were defined.

This action will list all the tables in the database to the left of the application. Note that only the tables are retrieved and not the columns. The application will only apply changes for tables placed on a canvas.

**Creating you Canvas**

Before you can add a table from the database, create a new table or add a relation whether it be a virtual or database relation you will need to add a canvas to the project. You can add as many canvases to the project as you require. Each canvas can hold its own tables, but a table can be added to one canvas.

From the File Menu select Project -> Add Project Canvas



**Tab Name**

This field holds the name of the tab. Note that this name will be used to save the canvas as a separate file in the directory defined in the project setup. Alpha Numeric and spaces are allowed only.

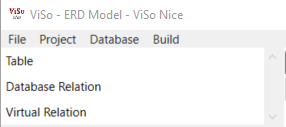
**Table Prefix**

If you would like to prefix all the tables in your database with a certain character set to group them together you can define it here. The option is not ideal for existing databases where no prefixing was applied, and the option should be left empty for these instances.

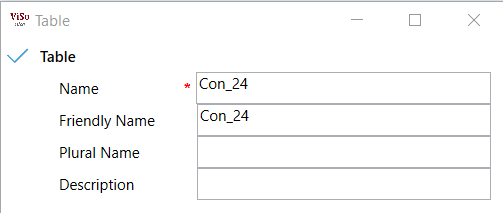
E.g. I created a table structure for a asset management application, and would like to prefix all my tables specifically used for assets with the prefix ‘Ast\_’, then I will add the prefix ‘Ast\_’ here. All the tables added to the canvas will enforce the ‘Ast\_’ prefix.

**Adding Tables**

If you would like to add a new table, from the left of the application, to the top, you will see a ‘Table’ option. Ct and drag the ‘Table’ object to your canvas.



This will open the Table Dialog



**Name**

The name of your new table. Database rules apply.

In the image above you will note that the prefix ‘Con\_24’ is added. This is because I am working on my Contacts canvas with a prefix of ‘Con\_’. The 24 is jut an arbitrary counter for background purposes and can be removed. If I would remove the whole wording and type in Contacts the application will re-add the ‘Con\_’ part and my table name would be ‘Con\_Contacts’

**Friendly Name**

A more user-friendly name for the table. Note that the table prefix is not enforced here.

**Plural Name**

This field provides for a plural name in instances where your tables are defined as singular, and you would like to have a plural name to use in the script builder program

**Description**

A short description of the table, if you desire so.

**Adding an existing table**

From the left side of the application select a table from the tables list and drag it to the canvas. If you drop the table on the canvas the application will retrieve the columns for that table and load the table structure on the canvas. Relations defined in the database will be drawn on the canvas.

Note Prefixing is enforced, so take care.

**Database File Menu Options**

In this section we will discuss the Database File Menu option not handled yet. Note that where applicable, if alternative connections were setup the file menu option will list the connections.

Note: When a database connection is selected the application will switch over to that database. For the most part this will not bother, except for Forward Engineering where the user will be forced to commit changes to the ‘Default’ database before any other database can be Forward Engineered.

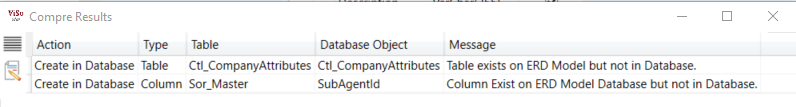
**Database**

* Refresh canvas from DB
  + This option will read the selected database and bring the canvases in sync with the database. Note that for the most part this will only applies to tables that are on a canvas, as tables not placed on a canvas are regarded as not in use.
* Compare to DB
  + This action will compare the canvas to the selected database. If the tables on canvases are the same as the database, the application will provide a message indicating so. If not, a screen will appear displaying the differences. Lines can be edited and the action set against it. As it is in my opinion rather bad practice to manage alternative databases in this manner, I have only provided an option to script the changes, and do not allow the application to do so for you.

It is rather handy though where you would like to drop tables and relations in a production database where development changes dropped tables and columns and the instruction to do so did not reach the build.

It is dangerous where development renamed a table or column, as the application will not be able to pick this up.

Make sure that you are sure that you would like to cation the changes before executing the script.



* Forward Engineer
  + This action will push changes made on the tables and columns the database. Id the ‘Default’ option is selected columns will be populated with the Default database ids to allow for the correct sorting when creating relations. Alternative databases will not update ids.
* Script Changes
  + This option will script all the Tables, Column and Database Relations with the changes to a SQL file. Virtual Relations are not included in this file.

**The Canvas**

The canvas is the working area where tables can be placed and viewed with their respective relations.

To the top left of each canvas you will see a select button.

By Selecting this button, you will open the Table Selector, or able to Include. This is a bit confusing if you do not understand the intend.

The intend of this is not to provide you with a list of what is obvious, but rather to provide you with an option to add a table to the canvas without adding it.

When the Build options are discussed you will notice that the problem is that you may want to add a table to a built that does not belong the canvas. Let’s say you are using Entity Framework with a Repository model, and in your repository, you would like to add your lookup tables.

The lookup tables are on your ‘Lookups’ canvas, and to make it even worse you have decided to use a table prefix.

This option the allows you to select the Lookups table, and it will only be used during the Build options.

Next to the Search button, and if you have added a prefix the canvas prefix will display, and if Locks are selected, the lock if applicable

**Navigating your Canvas**

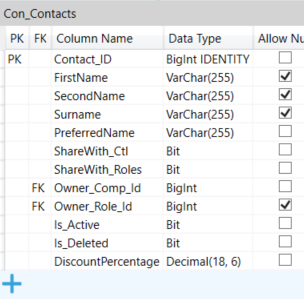
Canvases can be navigated by Left Mouse Button down and dragging.

Mouse wheel events will zoom in or out.

Table can be moved by selecting the table header, Left Mouse Button down and dragging. This is not an exact yet, so you may experience some frustration, until such time that I can fix it. Sorry about that.

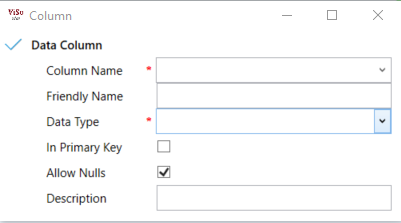
**Table Objects**

The table object is where you can interact with you table, database, and columns from the canvas.



**Adding Columns.**

By selecting the Add Option (  ) from the bottom of the table you will open the Column Add Dialog.



**Column Name**

The name of the Column. Database Rules apply.

**Friendly Name**

A friendly name that can be used in the script build, mostly for part of descriptions, or column mapping in EF type Builds

**Data Type**

The SQL Datatype to use for this column

**In Primary Key**

Select this option to indicate if the column forms part of the tables primary key structure.

**Allow Nulls**

Select this option if the database should allow ‘NULL’ values.

**Description**

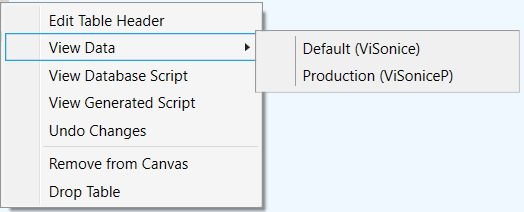
Provide a short description of the column’s usage and purpose. This can be used as part of the comments line in Build.

**Table Object Context Menus**

The table object consists of two contexts menus. Th first can be found on the table header, and the second on the column rows.

The Table Header Context Menu holds option valid to the table, and the Column Context Menu holds options mostly relevant to the selected column.

**Table Header Context Menu**



Note: If no alternative connections are available all menu options that are able to show these will only display the menu option without any child options.

**View Data**

This will open the table data view with a ‘SELECT TOP 100’ as default. There is still a minor bug on the rendering that I am working on that causes the scrollbar of the text man not to show in initial load. You can just select the Grid Split on top of the data grid and move it a little bit to display the scrollbar.

This view displays the data in the table by default performing a select top 100 from the database.

Note this option is dangerous as I don no validations and a delete from will execute.

F5 will refresh the data.

Note: All Data grids have an Export CSV, with a Pipe delimiter.

**View Database Script**

This option opens n plain text file with the SQL script for this table. It does not execute the script.

**View Generated Script**

This opens the files that will be generated if the Build options are setup and completed. The number of files will depend on your setup.

**Undo Changes**

Undo changes made to the table in this session. Not an exact science yet, and I am still working on it.

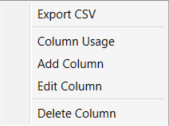
**Remove from Canvas**

Removes the table from the canvas and make it available for dragging on to another canvas.

**Drop Table**

Performs the same action as the ‘Remove from Canvas’. If the table was not placed on another canvas when the ‘Forward Engineering’ option is selected the table will be dropped from the database and removed from the tables list.

**Column Context Menu**



**Column Usage**

This option displays a list of all the tables where the Column Name is found and some information regarding its usage in the table.

**Add Column**

Opens the Add Column Dialog discussed earlier.

**Edit Column**

Opens the column in the Edit Column Dialog.

**Delete Column**

Removes the column from the table and drops it from the database during the next forward engineering session.

**Build File Menu Options**

**Build -> Script Builder Setup**

Note: Your project will require at least one Canvas with One Table and Columns for this option to be used.

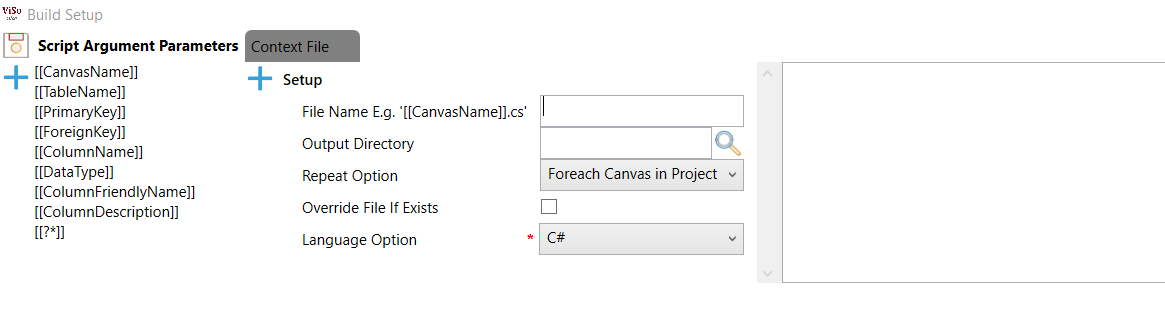
In the Script Builder Setup, you can define some files that would be generated if the ‘Run Script Builder’ is executed. This is nice to assist in generating routine code, such as default select queries.

On the First time the option is selected, or if no setup was done before, the application will prompt you for a name



This name will be the first Tab to represent your first file to be generated. If you are not happy with the file for some or other reason you can close the tab, and when saving the Build, the file will not be included and are gone forever.

Your screen may now look something like this, depending on your name provided.



**Save **

Saves the script options but does not close the window.

**Add Build Option **

Adds a new Tab to the window

**Scrip Argument Parameters**

List the parameters that can be used in the file. Note that some parameters can be used as part of the naming options.

**File Tabs**

Each tab has its own build setup options and will be applicable to its file only.

**Add Build Type**

Adds a new Build option to the tab. Build options are executed from top to bottom. I prefer to name my first option ‘Body’ as this holds the body structure of the code. We will discuss this a bit later in this document.

**File Name**

This is the file name to be generated. Note that depending on the option selected in the ‘Repeat Option’ will depend on the parameter options available and the filed caption will changed providing the options available.

E.g.



**Output Directory**

Provides the directory where to write the file to.

Note that the [[CanvasName]] parameter can be used in the directory

E.g. 

**Repeat Options**

This defines how this file should be generated.

Options are

* Foreach Canvas in Project
  + This file will be generated for each canvas in the project, containing all the tables for that canvas including the ones selected in the ‘Search’ option of the canvas
* Foreach Table in Project
  + This file will be generated for each table in the project, as long as the table are on some canvas, irrespective of the canvas it is on.
* Single File for Project
  + This file will be generated independent of tables and canvases.

**Override File if Exists**

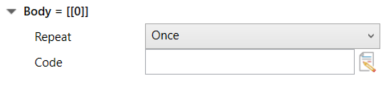
If this option is selected the file will be overridden if it exists, and only be created if it does not. This option is used for abstraction where the base holds the values that may change on each build.

For the file that inherits the abstracted file the option is deselected, and user defined code stays intact.

**Language Option**

This holds the mapping from the SQL data type to the selected language. Currently supports only C# and hope to add some more soon.

**Build Types**



For my first build type I added a type ‘Body’. This will hold the code for this build type. Note the parameter [[0]] next to the name. this parameter shows the build order index, and subsequent types may be referenced in parent types irrespective of order.

E.g. I may have a build type ‘Body = [[0]]’, and next a build type ‘Columns = [[1]]’ and one called ‘Properties = [[2]].

‘Body’ may reference [[1]] and [[2]] but ‘Properties’ may not reference any while ‘Columns’ may only reference [[2]].

**Repeat**

The repeat option here indicates how the build type should be executed.

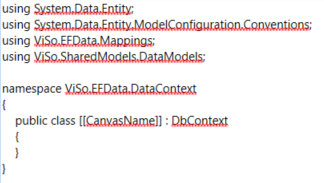
Options are:

* Once
  + The build Type will execute only once.
* Foreach Table in Canvas
  + The Build Type will execute for each Canvas in the project, subsequent build options will only have those tables available.
* Foreach Column in Table
  + Foreach Column in the table being executed.
* Foreach Primary Key in Table
  + Foreach Primary key in the table being executed.
* Foreach Foreign Key in Table
  + Foreach Foreign key int the table being executed.

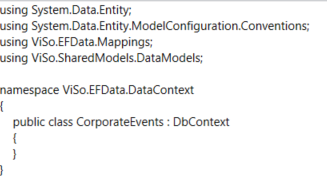
**Code**

Note that the right-hand side of the window displays your code output as it would look after generating, based on the First Canvas, and the First Table on that canvas the application can find.

For our main part ‘Body = [[0]]’ we would set this option to ‘Once’ and in the Code we will add the following.

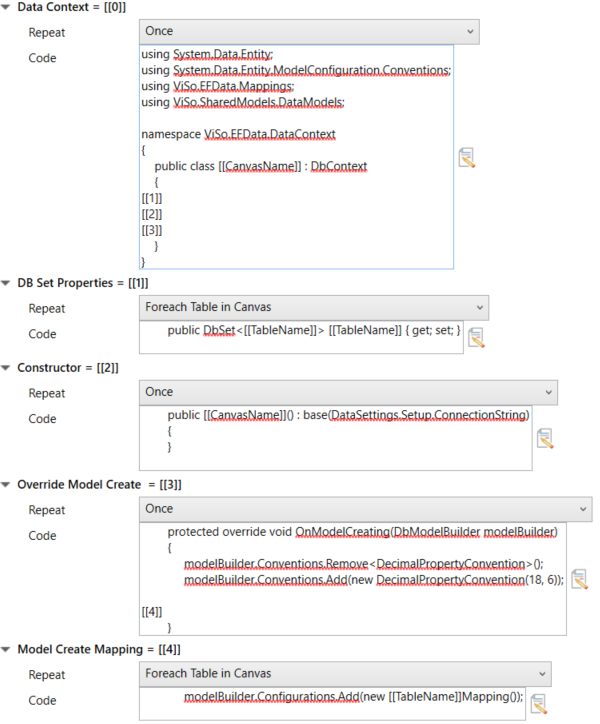


The output side will now look like this

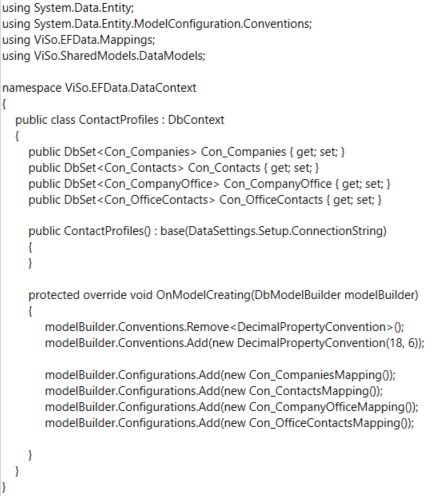


Note that the parameter was replaced with the table name.

Subsequent build types may be added and a setup that looks like this:

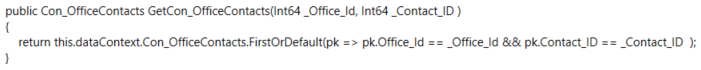


Will produce this code:



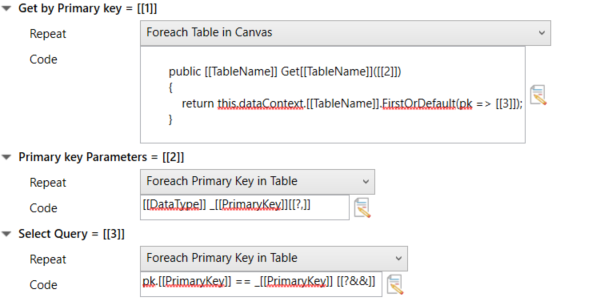
Lastly an explanation on the Script Argument Parameters. For most part they are exactly what they say. The only one with some explanation required are the [[?\*]] parameter.

This parameter can be used for repetitive actions where I would like to repeat some character after each iteration but not the last iteration. E.g. I would like to add my foreign keys with a coma delimiter to produce the following output.



I would then user the [[?\*]] parameter and change the \* to ‘,’ like this [[?,]] for the method signature, and like this [[?&&]] for the return results.

My setup would look like this



**Build -> Run Scrip Builder**

Runs the Script Builder based on your Script Setup.