## C:\Users\Allen\Documents\Visual Studio 10\Projects\XQuery\XQuery.GUI\Images\Excel.pngUser Interface

Upon double clicking the executable file that comes with the application, the user will be presented with an interface that will be the place where querying the underlying relational database will be accomplished. The user interface is quite simple and intuitive. The main textbox covering the breadth of the application is the location where any number of queries can be entered. One nicety of the textbox is that any recognized keywords will be turned to bold, so that the user can recognize the keyword is supported. Withal, the application is smart enough to parse the queries, partition each query group into a different querying process, and return a table of results as needed. Should the user desire to only query a subset of “query groups”, this can be done with ease. The user simply needs to highlight with the mouse the desired queries and select the *Query* button. If the user fails to highlight the queries, no worries should arise. The application will instead query everything.

The *Query* button has already been introduced, but we now turn our eyes on the other little buttons that appear towards the bottom of the application. There is a *Clear* button that the user can press that will clear any queries present in the textbox and position the cursor inside the textbox so that the user can begin to type more queries as he or she sees fit. The little buttons on the right from left to right are explained as follows: The first button, when pressed, will bring up another window that consists of the current schema or state of the database. The schema is stored as a tree view. The first level of the tree consists of the tables present in the database. If this level is expanded, the items underneath contain the columns that exist in the table. For the user’s sake, next to each table the number of rows present in the database can be found. It is noteworthy to mention that this schema window can be kept open and after any queries are run, the window will automatically update itself without the user’s explicit intervention. Should the *Schema* button be opened multiple times, the application will replace the current window, so that only one window may be opened at a time. In addition, if the user right clicks on any item in the tree, whether it is a table or a column, the name is automatically added to the query textbox to facilitate users in the query design process.

The second button that appears to the right of the *Schema* button is entitled the *Log* button. When pressed a log will appear that will contain sets of timestamps along with their corresponding log message. These log messages can contain messages ranging from “*table: Inserted x rows”*, or “*table: Deleted x rows”,* “*table: Created”,* or even “*table: Dropped*”. These messages are invaluable for delete, drop, create, or import queries, as opposed to select queries, where a result window is not returned and only the state of the underlying database is modified. Similar to the way the *Schema* window automatically updated its information when queries were run on the main window, the *Log* window will also automatically update itself even when open. This way up-to-date feedback on query results can be obtained.

The last window, though not so apparent at first, is entitled the *Results* window. When a user runs any number of select-type queries, the database will query the desired tables and return a *Result* window for each select query found in the set of queries run. Each *Result* window will contain a data grid with columns and rows, which present the results of the query where one might possibly find joined, sorted, or grouped data depending on the query entered.