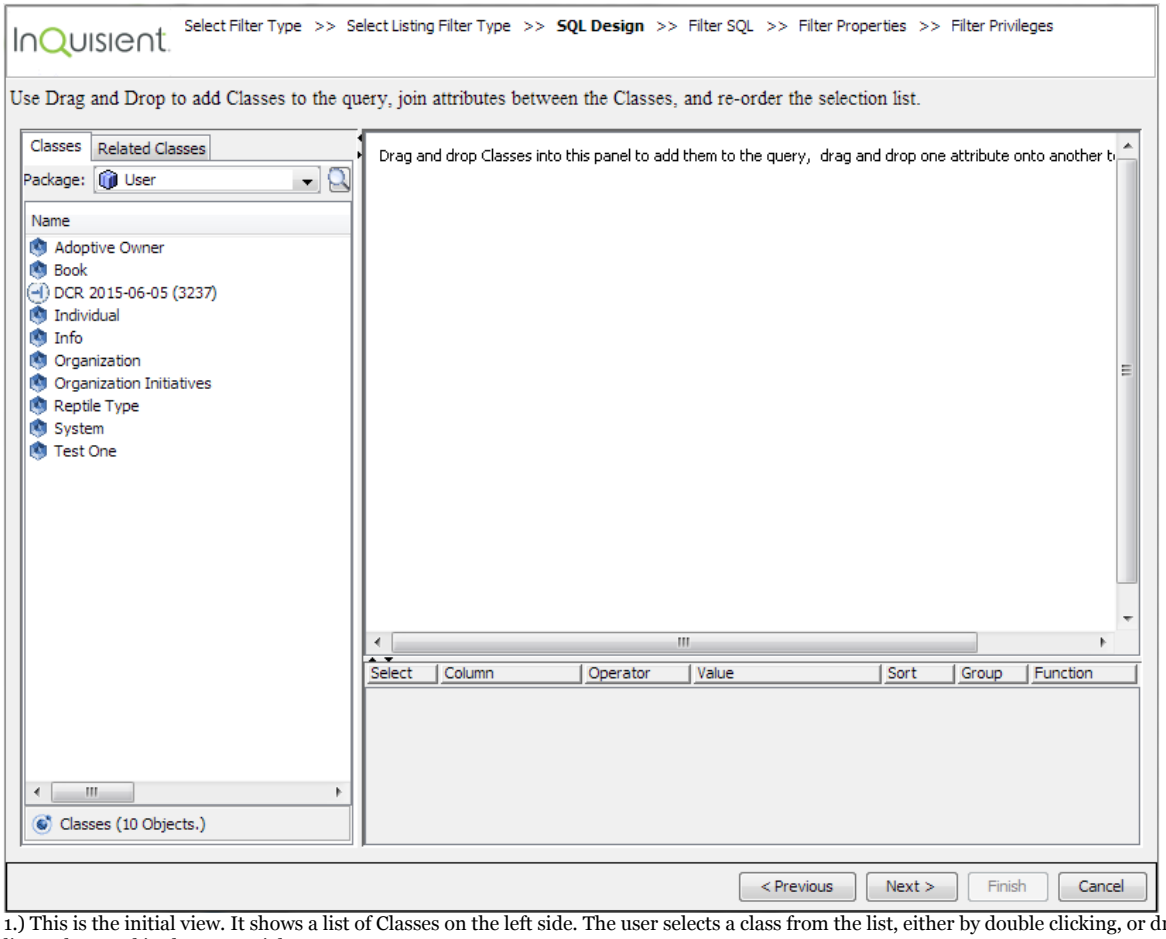
HTML SQL Query Wizard

Summary:

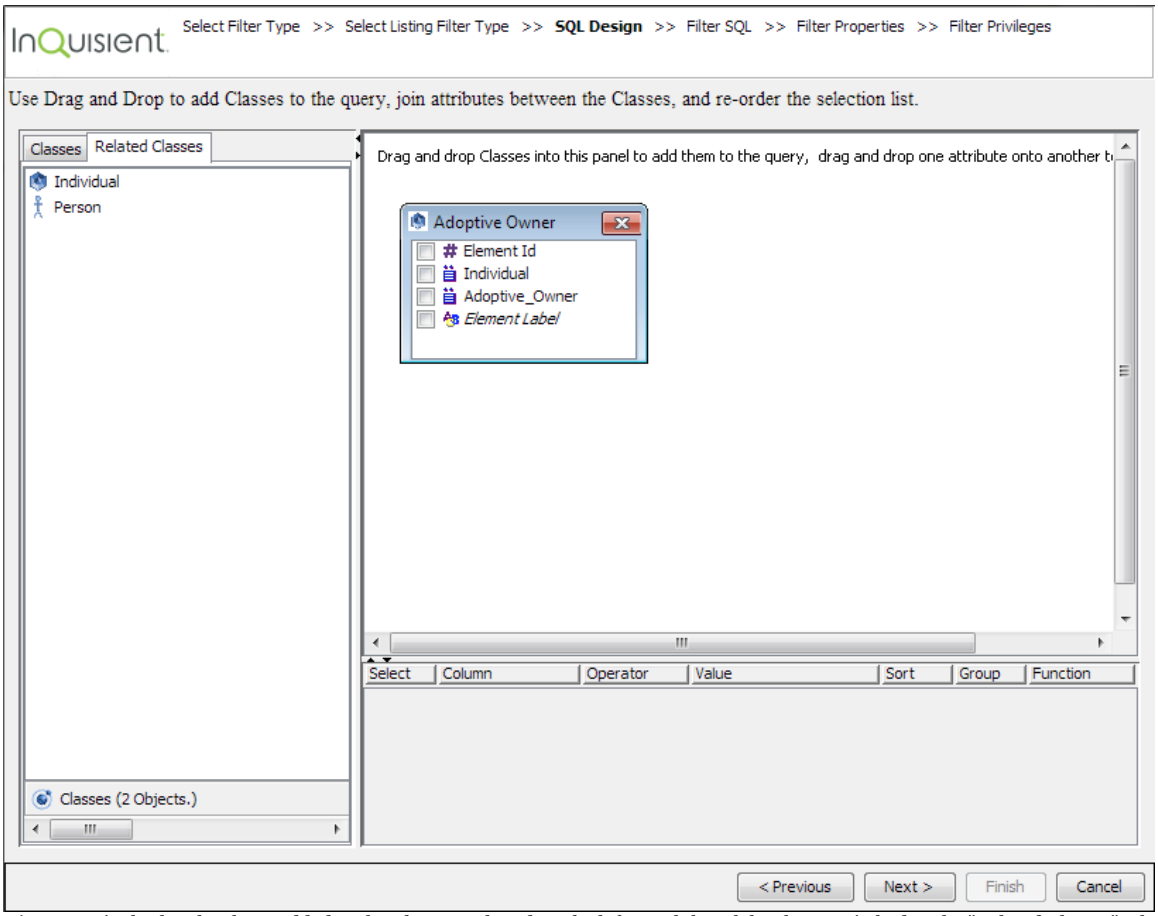
Our product currently has a SQL query creation wizard, implemented in Java Swing. In addition we have an HTML “Query By Example” (QBE) feature. The goal of this project is to produce an accessible, highly usable, SQL query creation wizard implemented in HTML5. We expect this will be a multi-step project.

At a high level, the design will present the user with a list of database tables (hereafter referred to as "classes"). The user will select the class, or classes, they want to query, and enter "filtering", or SQL WHERE clause restrictions. The user must be able to indicate which columns they want to include, in the SELECT clause, from each of the tables. They should also be able to add SQL functions such as (but not limited to) min, max, avg to the SQL select clause. The interface should “automagically” take care of the required SQL joins, by using the server supplied meta data which will include the database’s key relationships (relations). The interface must also allow the user to declare column joins between tables even if there is no existing meta data to indicate a valid join.

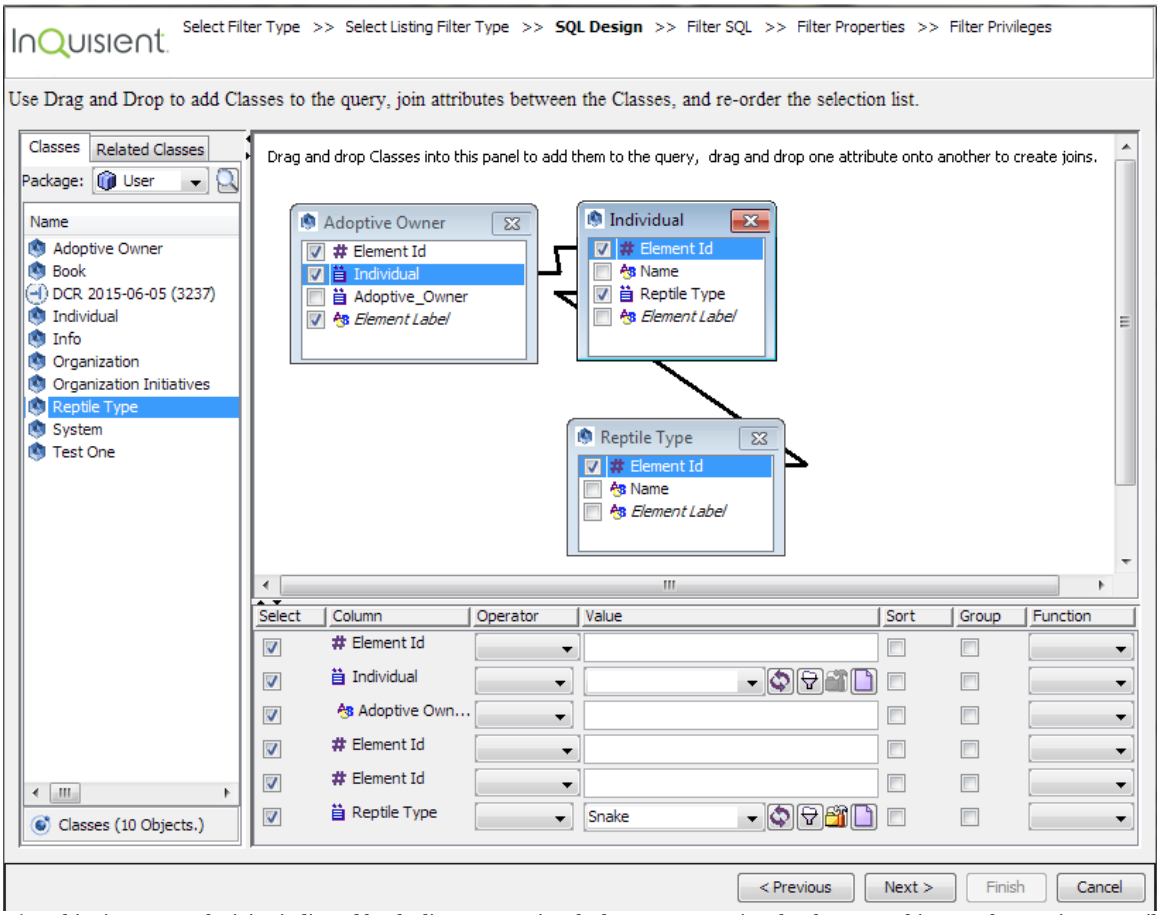
Following are some screen captures of the existing Java Swing interface:



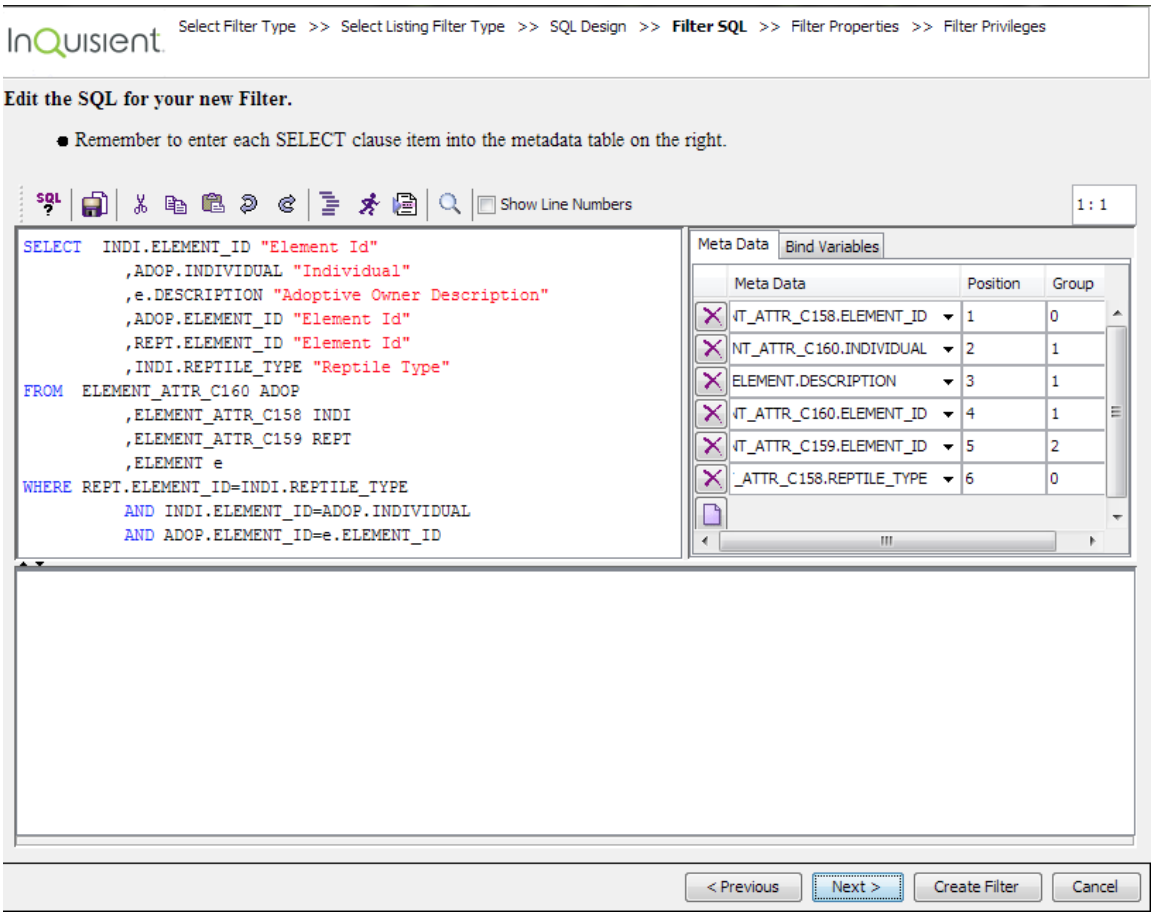
1.) This is the initial view. It shows a list of Classes on the left side. The user selects a class from the list, either by double clicking, or drag and drop from the list to the panel in the upper right.



2.) Here a single class has been added to the Class Panel, and on the left panel the tab has been switched to the "Related Classes" tab. This tab shows the other classes which have declared relationships to the selected class. In this case the idea of a "relationship" corresponds to the SQL concept of Primary Key/Foreign Key relationship. Our product stores such meta data, although the keys are not declared in SQL DDL. From this list a user should be able to select one of the related classes and add it to the Selected Class Panel.

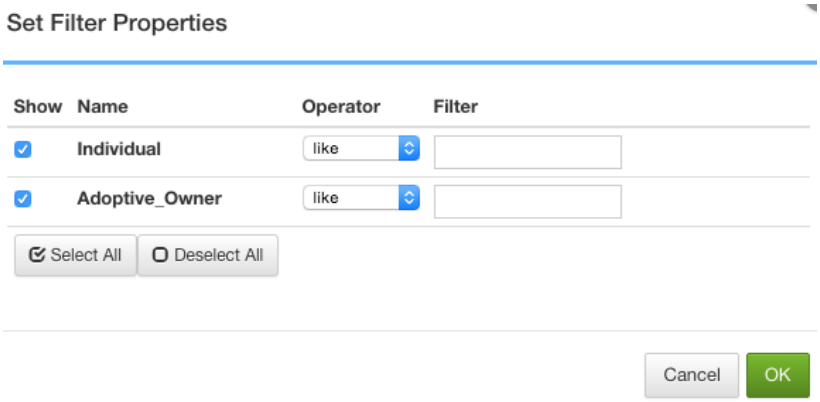


3.) In this view we see the joins indicated by the lines connecting the boxes representing the classes. In this case, the user is responsible for indicating the columns for the joins. NOTE: With the "Reptile Type" class selected, we've added a WHERE clause restriction of "Snake" for the reptile type in the form area below.



4.) In this picture the user has clicked the Next> button from the previous screen and the wizard is displaying the SQL text to the user. The toolbar above the SQL text allows the user to execute/test the query, and perform other functions.

The following screen shot shows the existing HTML QBE interface. This interface operates with a single table (class), and does not support the multi-table joins. However, when setting the properties for individual columns within a single class, our first effort can follow this model. Multiple classes in a single query could use a collapsible stack of tables, when one of the tables is expanded, it would look like the following (with some additional controls, and fields).



Architecture:

The existing wizard does a good job for users who are familiar with SQL and the underlying concepts. The goal for this project is to produce an interface which will be more usable for ordinary business users, and those who are unable to use Java Swing when launched from a browser.

We'll employ a split design, the HTML interface will communicate with the application server via HTTP/JSON to retrieve meta data about the database tables and relations. It will load, or create, a JSON object describing the query in terms of the class meta data — this represents the wizard's UI state. The application server will use this JSON object to actually create the SQL. The JSON description of the meta data driven query will be stored in the database so it can be reloaded into the HTML interface for editing, or to be cloned to create new queries. The scope of this project does not include the server side, SQL generating, code.

Order of operations:
1. The event handler for a link/button will create a call to the wizard component's constructor passing in the configuration object. The configuration object will either contain the "class meta data" that is "in scope" for the host page, OR it will provide a URL, that the object can use to determine the initial context set of classes. The wizard will display these in a tree. The reason for the tree is to allow the user to traverse the relationships that involve the specific class. The header of the list will also contain a search field so that the user can search the database for additional class information, and add those to the list.

2. The user will select a class, which will then display on another panel. Since the wizard must remain accessible, we will use HTML markup to define the list of classes that make up the query (see QBE description above). We may add a tab, or other UI device, to also create a graphical, or diagrammatic (ERD like) view in the future, but the initial requirement is for an HTML DOM representation.

2a. The user can select which columns from this class will appear in the queries SQL select clause, and they should be able to enter a label, or "alias" for that column as it will appear in the result set. They should be able to enter SQL WHERE clause restrictions, combining an "operator" from a list (specific to column type), and an input for the predicate value.

3. Repeat step 2 for as many classes as are needed. The wizard will need to perform bookkeeping on the joins (relations in the meta data) as the user works. It must also support a user defined join in the case where there is no meta data defined relation. But all tables must be joined (number of joins is one less than the number of classes in use), we do not want the user to be able to create queries which produce Cartesian products of multiple tables.

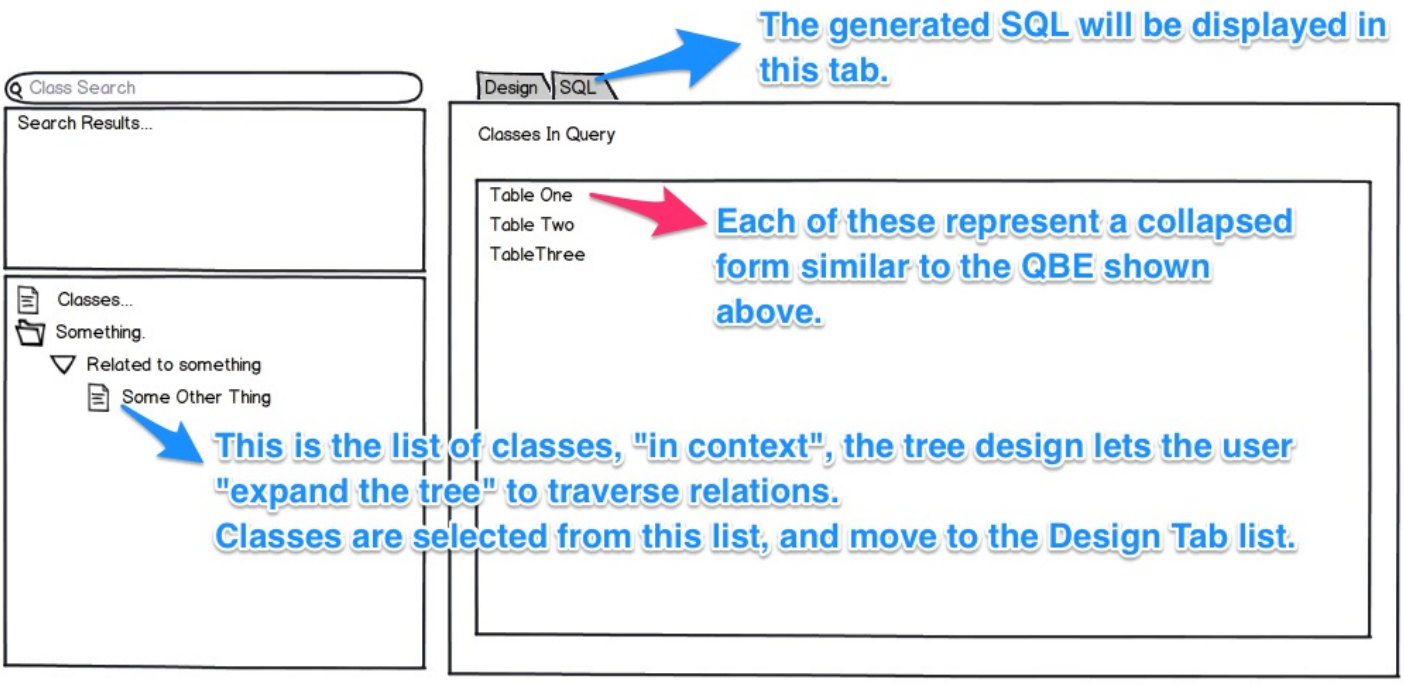
Initial Project Plan:

This document is not intended to be a comprehensive specification, it is intended to be an introduction to the project, we will flesh out many more detailed requirements as we go. The first step is for our Product Management group to agree on a prototype design.

As a first deliverable I would like to have an HTML mockup of the user interface available to discuss at InQuisient's product management meeting on 24 July, 2015, to accomplish that, I would like to have the mockup completed prior to July 23. This mockup should be implemented in HTML and use Bootstrap CSS styling, and does not have be fully functional. It would be nice to be able to show a class being taken from the tree, and dragged to the Design tab, where it would show a form similar to the QBE example above. For that to work, I should supply you with some sample meta data, which I will try to do, but lets work on the HTML to start with.

My intent is to put it on screen for a discussion of the design. At that point we'll make additional decisions on how to proceed.

Here is a mockup, as an idea for the first deliverable. Note that this mockup is not the required design, it is merely one idea. We ask that you include your own ideas too. They are very much welcomed!


The generated SQL will be displayed in this tab.
Class Search
Design
SQL
Search Results...
Classes In Query
Table One
Table Two
TableThree
Each of these represent a collapsed form similar to the QBE shown above.
Classes...
Something.
Related to something
Some Other Thing
This is the list of classes, "in context", the tree design lets the user "expand the tree" to traverse relations.
Classes are selected from this list, and move to the Design Tab list.