# Query System Development

## Overview

The basic premise of the dynamic query system is that it functions on querying against a domain and range. Since the purpose of the entire application is to evaluate how students are doing in the program, the high level domain is always going to be the student when running queries. This corresponds directly to the student table in the database. Now, the domain can be expanded upon into other dimensions (subdomains), while it also simultaneously can be narrowed down by selecting on attributes within the domain space. In the end, the baseline of the domain is always the Student table. Subdomains can be selected to add dimensionality to the domain space, but the subdomain must have a relationship to the student (ie. a table with a relationship to the Student table). The domain space can be filtered down to a specific subset by using conditions on attributes to filter. Filtered attributes must exist within the domain (student) and subdomain tables.

While the domain for the query system is pre-determined, due to the nature of the application, it is up to the user (or the developer) to specify the range space in which they wish to query. Like the domain, the range must correspond directly to a table within the database. Also, the range space cannot be disjoint from the domain space (table in the range space must have a relationship with a table in the domain sub-space). You cannot query about a relationship that does not exist! Since the application is focused on students, all query results return a count of students that exist within intersection of the domain space and the range space. The range space must be narrowed down to a specific attribute to query against (eg. grades in the section range), as well as all of the conditions to check for that attribute (eg. A, B, C, D, F for grade attribute). A count of students will be returned for each condition on the range attribute.

Lastly, comparison groups can be made, to be able to query multiple groups of students against the same range at once. This essentially allows the user (or developer) to specify multiple domain sub-spaces to query against the range in sequence.

## GET vs. POST Functionality

Currently, the dynamic query system supports two different types of RESTful requests: GET and POST. The query controller has two different functions to support these requests: results and search. The results function is designed to handle dynamic querying by GET requests. Parameters here are expected to be sent in via the URL. Note that this function is not as robust as the search function, due to the nature of URL parameters. The results function cannot query with comparison groups, nor does it display any visualizations!

The second function, search, is the recommended function to use for dynamic querying. This function expects a POST request, from, say a form, that sets the params dictionary. Since POST requests allow for more robust data handling and representation, comparison groups can be queried against here. Querying using the search function will also create a visualization for the results as well.

## Description of Parameters

URL params:

* group: An array of comparison groups, each with a subdomain, a number of dattributes, and a number of corresponding dconditions for those dattributes.
* subdomain: The domain table to query against in conjunction with the student table.
* dattribute: An array of attributes to condition on within the domain.
* dcondition: An array of conditions to apply to the dattributes. Each dattribute should have a corresponding dcondition.
* range: The range in which the results will be searched upon. **REQUIRED**
* rattribute: The specific attribute in which the results will be focused on. **REQUIRED**
* rcondition: An array of condiitions to apply to the rattributes. Each rattribute should have a corresponding rcondition. **REQUIRED**
* literal: A boolean value. If this is set to true, the dattributes with their dconditions will be linked by SQL conjunctions/disjunctions. Therefore you must include these in the conditions if you want to use literal!

Visualization Params:

* v\_type: The type of visualization to display. See type options below.
* v\_data: The representation of the data. Can either be "count" (default) or "percent". This parameter only applies if the v\_type is stacked.

Visualization Types:

* histogram: This is the default option. A visualization with data in separate columns. Can only display counts.
* stacked: A stacked histogram visualization where data is stacked. Data is stacked into columns by comparison groups. Can display counts or percents.
* polar: A polar coordinates visualization of the data. Can only display counts.

## Security

Due to the nature of the dynamic queries, the dynamic query system is not secure! This system is at risk for SQL injection, and possibly other malicious attacks. Since the entire application is designed to be used by internal users, security risks are quite small. If this system goes beyond that, security should be a top priority and seriously considered and researched.

## Conclusion

In conclusion, the dynamic query system is robust and allows for any sort of count queries to be run, as long as the domain and range space of the query are directly related to the Student. Unfortunately, any tables in the database that are not directly associated with the Student cannot be queried against. Unfortunately, the current design of the database and poor integrity of testing data proves a major roadblock in developing further robustness to the dynamic query system. With a redesign of the database system to represent entities in a more pure manner, along with a fuller testing set for verification and testing purposes, it may be possible to do. In theory, allowing for multiple subdomains, or even possibly subranges, may do the trick. A lot of effort can go into the security aspect of the system as well. With a bit of an overhaul to the entire database schema, along with a bit more time and effort, the dynamic query can definitely be improved and become even more robust if desired.