The purpose of this project is to connect multiple databases across a grid in such a way that they appear as a single data repository while optimizing data flow on the back end. More specifically, the goal in this case deals with simplifying the process of acquiring data for health researchers. For example, when a researcher needs to find data on cancer statistics or HIV drug resistance, that researcher needs to spend time connecting individually to each hospital's database and then manually compile the data. This project will provide a web application for researchers to search a collection of databases from participating institutions and present the data as though it were coming from a specific source. Additionally, the researcher has the ability to choose a display template to only focus on the specific data that he or she needs from the collection of databases.

The problem with linking databases is that hospitals will most likely store their data in completely different fashions. Though some of the basic data supplied will be the same, table names and field names within each database will almost all be different. Getting cooperation from each hospital to restructure their database to a master format would be impractical. To solve this problem, this web app provides the administrator with the ability to set up a master template of the fields that they want to be available in search results. When each database is added to the system, the administrator links the appropriate fields to the master template. It is from this master template that the administrator creates the multiple display templates. For example, across several databases, a field with data about the researcher might be known as source, site, source_data, etc. Additionally, these fields are probably in tables with all different names, such as hiv_data, gene_info, etc. This makes it very difficult for a grid system to link the data unless it has the information on how the fields should be linked. The administrator can simply choose to call this field in the master template “Source,” and then regardless of all the different table and field names, the information will be correctly linked and displayed to the researcher.

This project seeks to address the issue of data optimization so that researchers have the fastest access to data that they seek. Image files by the MRI or X-RAY are each several megabytes in size, and thus this could present a particular problem if researchers in America are consistently trying to access this data from Asia. The process would be much slower than if the data were stored on a more local server with good bandwidth. In order to solve this problem, the web app will track statistics on usage patterns and decide where to move the data so that it is best optimized for each researcher. To make the system more realistic for actual use, it will implement a method of caching in order to avoid legal issues of actually deleting and moving the data between systems as well as to avoid putting the original owner of the data at a disadvantage.

Clearly the perfect situation would be complete caching, where every institution has a complete cache of every other system. Since this would be impractical, the best realistic solution would be to monitor usage patterns of particular data sets and cache only the most highly requested data. There are several constraints on how to decide where to cache the data, including whether or not that system allows caching, whether the system belongs to an institution that should receive special preference for research data, and constraints regarding the space allocated for files, the database, and the amount of bandwidth. All of these settings are maintained by an administrator and guide the system in deciding where to cache data.

When the caching process is complete, each region should have as many local copies as possible of frequently used databases from other regions. Searches and downloads will be much faster because the majority of the accessed data will now be local, and the system will even skip over querying remote servers when it knows it already has an updated cached copy of that server.