Advanced Querying Features for Disease Surveillance Systems

Mohammad R. Hashemian[1]

[1]**The Johns Hopkins University Applied Physics Laboratory**

Abstract: ***Most automated disease surveillance systems notify users of increases in the prevalence of reports in syndrome categories and allow users to view patient level data related to those increases. Occasionally, a more dynamic level of control is required to properly detect an emerging disease in a community. Dynamic querying features are invaluable when using existing surveillance systems to investigate outbreaks of newly emergent diseases or to identify cases of reportable diseases within data being captured for surveillance. The objective of the Advance Querying Tool (AQT) is to build a more flexible query interface for most web-based disease surveillance systems. This interface allows users to define and build their query as if they were writing a logical expression for a mathematical computation. The AQT allows users to develop, investigate, save, and share complex case definitions. It provides a flexible interface that accommodates both advanced and novice users, checks the validity of the expression as it is built, and marks errors for users.***

Keywords— ***public health informatics, population surveillance, disease outbreaks, software tools***

Introduction

In its 2007 annual report, the World Health Organization warned of the increased rate at which diseases spread in a world where 2 billion people travel by air [1]. The early detection of known and emerging illnesses is becoming more important. Automated disease surveillance systems have been in existence for over 10 years [2-4]. Most of these systems analyze data by syndrome and search for disease outbreaks. A syndrome in this context is defined as a group of diseases related in some fashion, such as respiratory diseases. This level of investigation is often sufficient, but a more dynamic level of control may be required to understand an emerging illness in a community.

For example, during the 2002–2003 Severe Acute Respiratory Syndrome (SARS) disease epidemic [5], the respiratory syndrome definition used by most automated disease surveillance systems was too broad to track SARS [6]. In this case, the users needed to create queries that looked for specific keywords in the patient chief complaint or specific combinations of ICD-9 codes [7]. A chief complaint is text entered by a triage professional in an emergency room or a

clinic, based on a patient's description of their primary symptoms. Today's public health departments must deal with a multitude of data coming from a variety of sources. For example, Electronic Medical Record (EMR) data include sources such as radiology, laboratory, and pharmacy data. A more sophisticated querying tool is needed to assist investigators with creating inquiries across multiple data sources [8-10].

Currently, there are surveillance systems, such as the Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE) [11], which provide limited dynamic querying capability. However, we wanted to design a flexible and simple graphical user interface (GUI) for this and other types of surveillance systems. Our prototype system, the Advanced Querying Tool (AQT), allows the investigators to handle complex cases where one can incorporate any data elements available in a disease surveillance system, then mix and match these data elements in order to define valid queries. Hence, this system removes the need for database administrators and application developers to define pre-packaged database queries and user interfaces every time a new and innovative query is written.

As an example, investigating a potential influenza outbreak in an adult population may require respiratory syndrome queries only, while investigating a similar outbreak in children under 4 years old may involve queries in both gastrointestinal and respiratory syndromes (Figure 1).

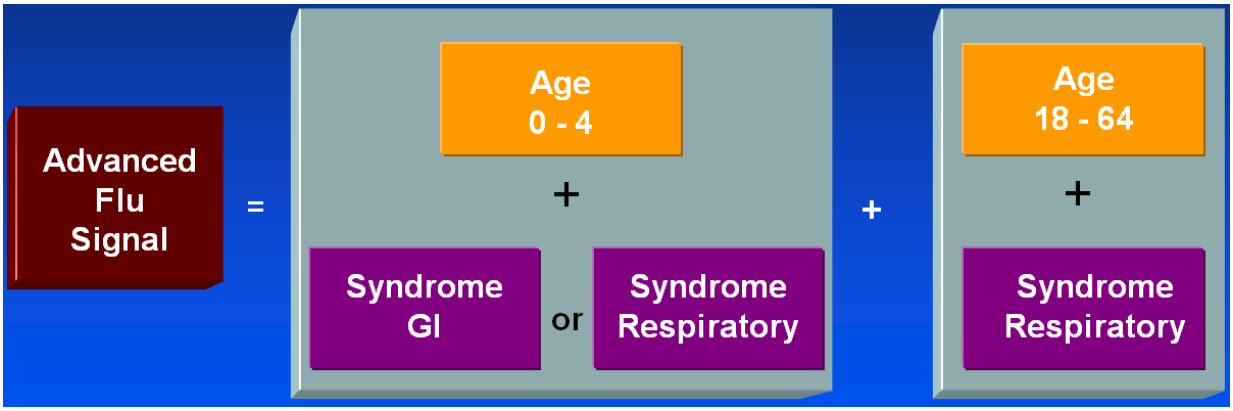

Figure 1. Running multiple inquires as one query

Table 1 provides examples of how a dynamic query tool exploits combinations of data elements available to disease surveillance systems. Most automated disease surveillance systems have a fixed number of predefined syndromes. These applications severely limit the surveillance system value for diseases that fall outside of its broad syndrome categories. The background noise level rises when all the chief complaints that potentially fall into a syndrome category are included, which in turn requires many more positive cases to identify an abnormal condition. Merely adding sub-syndrome categories, that are more granular than syndromes and cover a broader range of conditions than typical syndromic surveillance like injures and chronic disease [12], provides the users with a more comprehensive means to filter the analysis window. If a disease surveillance system has 400 sub-syndromes, then taken singly the user has 400 additional choices; by combining two or three sub-syndromes, the analysis options are magnified to over ten million choices. Of course not all of these options are sensible, so the actual number of

options is somewhat less. Even greater analytic flexibility is provided through the use of data elements contained within electronic medical records. The capability to select a combination of a microbiology laboratory result, radiology result, and ICD-9 code provides for a powerful tool that enables the public health community to rapidly identify specific high risk patients.

Table 1. Potential analysis combinations using multiple data sources in combinations

DATA TYPE	SINGLE ITEMS	DOUBLE COMBINATIONS	TRIPLE COMBINATIONS	TOTAL ANALYSIS CHOICES
SYNDROMES	**10**	**NA**	**NA**	**10**
SUB-SYNDROMES	**400**	**79,800**	**10,586,800**	**10,667,000**
ADVANCED QUERY TOOL:				
ICD-9 CODES	**1,000**	**499,500**	**166,167,000**	**166,667,500**
LABORATORY TESTS	**300**	**44,850**	**4,455,100**	**4,500,250**
RADIOLOGY	**100**	**4,950**	**161,700**	**166,750**
PRESCRIPTION DRUGS	**10,000**	**49,995,000**	**1.67E+11**	**1.66667E+11**
ICD-9 +LAB+RAD+PRESCR	**11,400**	**50544300**	**1.66787E+11**	**1.66838E+11**

Objectives

The following objectives summarize the design features of the AQT:

The tool's interface will help generate queries that can process any kind of data regardless of its source (e.g., emergency room visit, office visit, pharmacy, and laboratory). Unlike fixed-form query interfaces, AQT will not restrict users in what they can query. Instead, the user will be able to formulate ad-hoc queries across assorted data sources without the need to understand the underlying data models and the query languages associated with different systems. In addition, using this tool should save investigators' valuable time in obtaining the query results. Currently, if the surveillance system cannot generate the desired queries, the application developers and/or database administrators may have to create new interfaces or functionalities. The AQT, however, empowers the users to move forward with their research without waiting for developer or administrator modifications to the surveillance systems.

The interface will accommodate users with different levels of experience in creating complex and valid queries. The process will be natural and follow the same patterns that one uses to express a mathematical equation. At the same time, it will give the more experienced users, who are familiar with the data elements, the freedom to define complex queries by

sidestepping the guiding tools. The advanced users will have the ability to type in their queries and the tool will validate them and provide feedback on possible syntax errors.

The interface will allow users to save and share queries with other public health professionals, even in different jurisdictions. After defining a complex query the user has the ability to store the query for future investigations. One should be able to execute the stored query repeatedly in the future, include it as a segment of a bigger query, or customize and execute it. These saved queries can then be shared as part of collaborative efforts among users in different departments and jurisdictions. AQT will provide an interface for disease surveillance systems to store, retrieve, and share queries. These capabilities are especially valuable for users employing a case definition for following a disease outbreak. A case definition is a set of symptoms, signs, etc., by which public health professionals define those patients considered to be directly a part of the disease outbreak.

Finally, the tool should be self-contained and generic. This allows most web-based disease surveillance systems to incorporate the AQT into their systems.

Methods

Interface

The entire functionality of the tool is placed within a single web page (Figure 2).

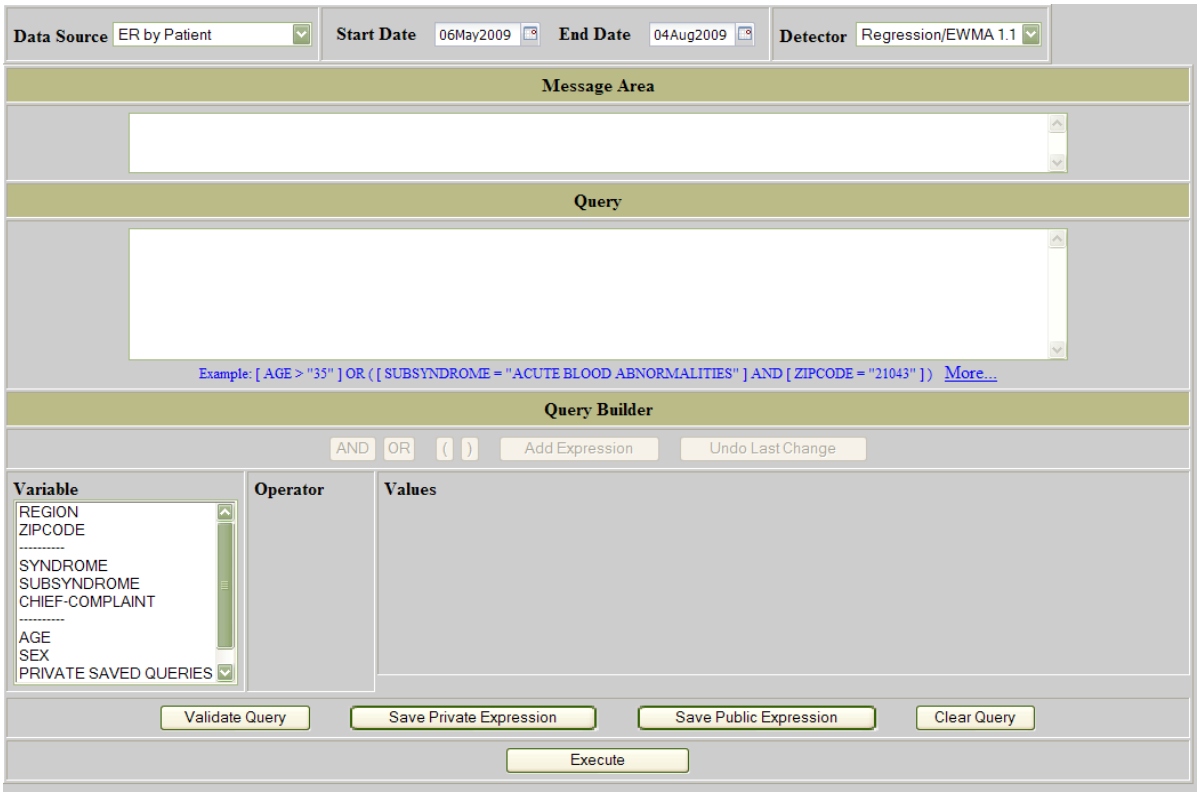

Figure 2. Advanced Querying Tool interface

The screen in Figure 2 is divided into 5 major sections. Starting at the top, the user can filter the data by picking the data source from a dropdown list, start and end date. The surveillance system should supply this list of data sources to the AQT. The next area below is the Message Area where the GUI communicates with the user. Any information, warnings, or error messages are displayed in this section. The next area, the query section, contains the query expression. The users can either directly type the query expression or use the tool to generate the query expression and paste it in this area. Alternatively, they can use a combination of the two methods by typing part of the expression and pasting the rest using the query builder. The query section is followed by the query builder section where the tool provides list boxes, buttons, etc., to direct the user through the process of generating the query expression. The bottom section is where an action on the query is performed. Users can validate the expression's syntax, save the query for their own future use, save it to be shared with others in the user community, clear the query expression and start over, or simply execute the query and get the results.

Data source

As mentioned earlier, the capability to generate queries on data from a variety of sources is one of the objectives of the AQT. Each data source has its own distinctive set of data elements. The interface has to provide a list of data elements pertaining to the chosen data source. For example, the data might represent different geographic regions from one data source to the other. That is, one source might have data identified by zip codes while another source uses some other type of defined region such as hospitals, pharmacies, and schools. Another area where data sources can be different is in medical groupings. For example, office visits often use ICD-9 codes [7], while emergency departments use patient chief complaints. The interface is designed to distinguish valid data elements for each data source and populate the data element list box accordingly.

After selecting a data source the tool populates a list box with a set of associated data elements for the data source. The list box is divided into three major areas:

- The geography system
- The medical grouping system
- Others such as age, sex, saved and shared queries.

Figure 3 shows how the medical grouping systems differ for Emergency Room (right) and over the counter (left) data sources.

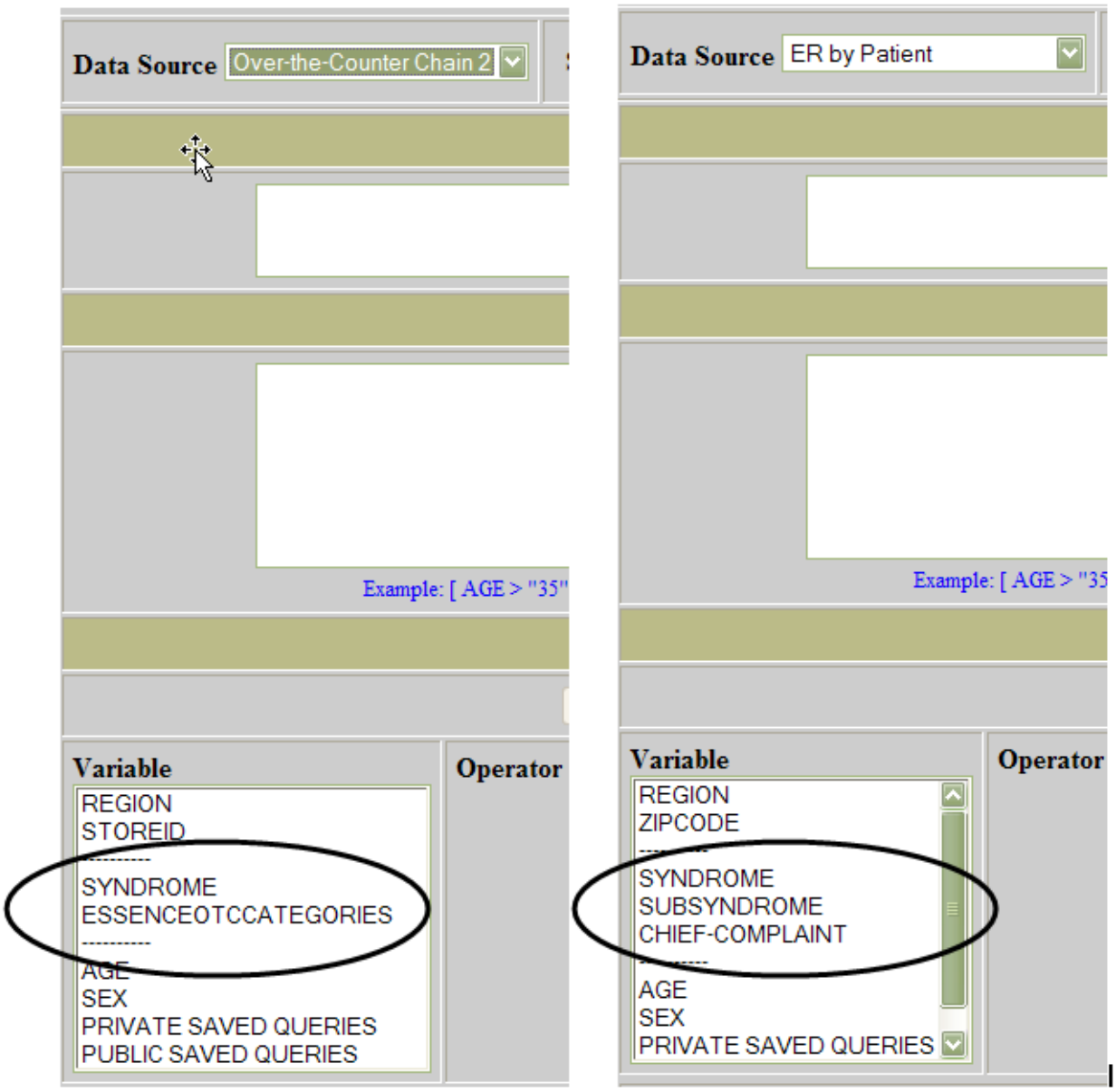

Figure 3. Different data elements for each data source

Flexibility

As mentioned earlier, a main objective of the AQT is to provide an interface that caters to both novice and experienced users. The experienced users simply type the query, while beginners and those who are more comfortable with a guided interface can use list boxes and buttons to generate the queries. In fact, one can type part of the query and use the tool to generate the rest of the query (Figure 4). When a user types a query directly, it is assumed that the user knows the syntax and valid data elements pertaining to the data source, though the tool does check the syntax and provide feedback.

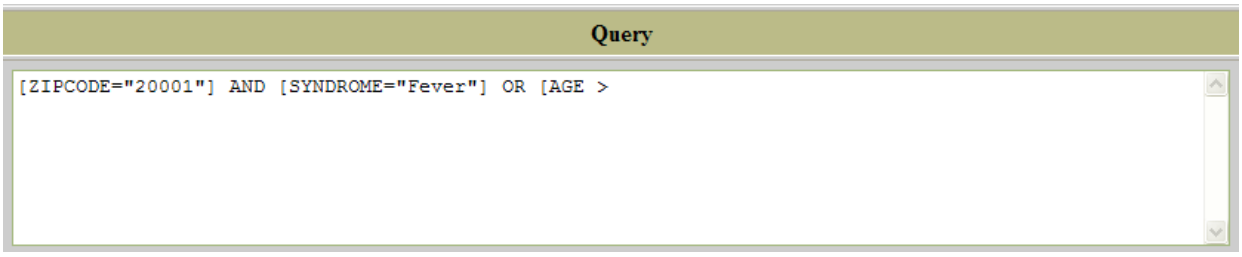

Figure 4. Generate query expression

Because we want the users to define and build their query as if they were writing a logical expression for a mathematical computation, the syntax is simple and close to the "where" clause of a Structure Query Language (SQL) statement. However, one does not need to know SQL to write the expressions. A query consists of one or more simple expressions joined by "AND" and/or "OR," negated by "NOT," and grouped by parentheses. A simple expression is enclosed within square brackets ([]) and defined by a variable, a logical operator, and a value. For example, if an investigator is searching for reported fever cases within a specified zip code, the query then consists of two simple expressions; one which searches for the specified zip code and the other which checks the fever syndrome. The final query may look like the expression below:

[ZIPCODE="21043"] AND [SYNDROME="FEVER"]

If the investigators want to narrow the search into a certain age group they can type or use the tool to add **AND [AGE = "0-4"]** to the above expression. Hence, the users can add more conditions without worrying about the underlying data model.

The most complex part of the syntax occurs when searching for values that contain, start with, or end with a set of characters (Figure 5). In this case, the syntax uses "*" as the wildcard character. For example, a user would type **[chief-complaints = "*head*"]** in the query box if he/she is looking for all the records of chief-complaints that include the word "head." Similarly, if a user types **[chief-complaints = "head*"]** or generates it using the tool (selects the Starts With from the operator list box and types *head* in the text field), the resulting query would search for all the records where the chief-complaints field begins with the word "head."

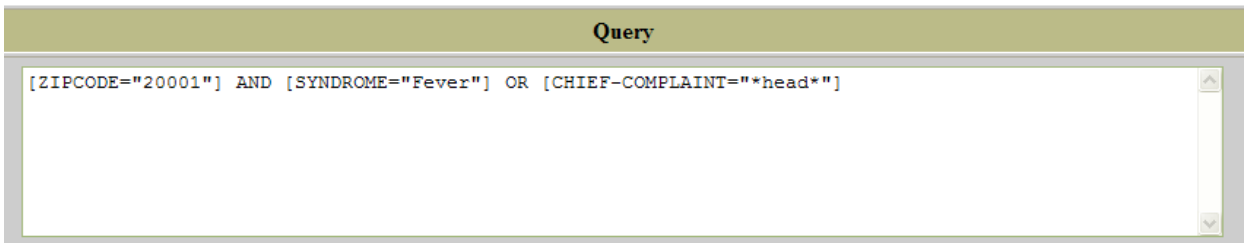

Figure 5. Wildcard in expressions

Natural flow

The procedure for generating expressions follows the same pattern a person would use to create a logical expression. The interface will provide a natural flow to help the users to create an expression as if they are typing it. They may start with selecting a data element or variable such as 'SEX', then a logical operator like '=', and finally a value like 'MALE' or 'FEMALE'. The user can add 'AND' or 'OR' and create the next expression using this same process.

The user can interject expressions in the middle of a query, remove parts of the query, or undo the last change made to the query. As changes are being made, the tool validates the entire query in the background and provides instant feedback. This method of constructing queries is more intuitive to the users than that of creating all the individual expressions first and then joining them together.

Once the data source is selected, a list of core data elements is provided in a list box. From the list box the user can select a data element. Based on the type of the data element, a list of valid logical operators for that data element is placed in another list box. Figure 6 shows the list of valid operators for text fields.

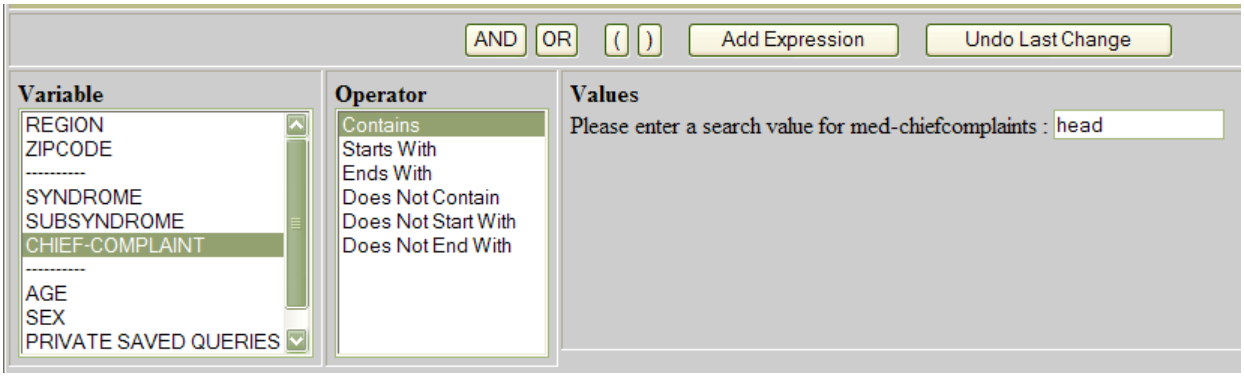

Figure 6. Valid operators for long text fields

In cases such as zip code and syndrome, '=' and '<>' operators are also valid. For age the operators '>', '<', '<=', and '>=' are added to the list. Once the user selects a data element, a list of valid values pertaining to the data element is listed in yet another list box. The user can select one or more of these values, and if more than one value is selected the user can choose to group these values using 'AND' or 'OR'. Note that the AQT generates the expression in a left to right progression in the same manner as one typing the expression (Figure 7).

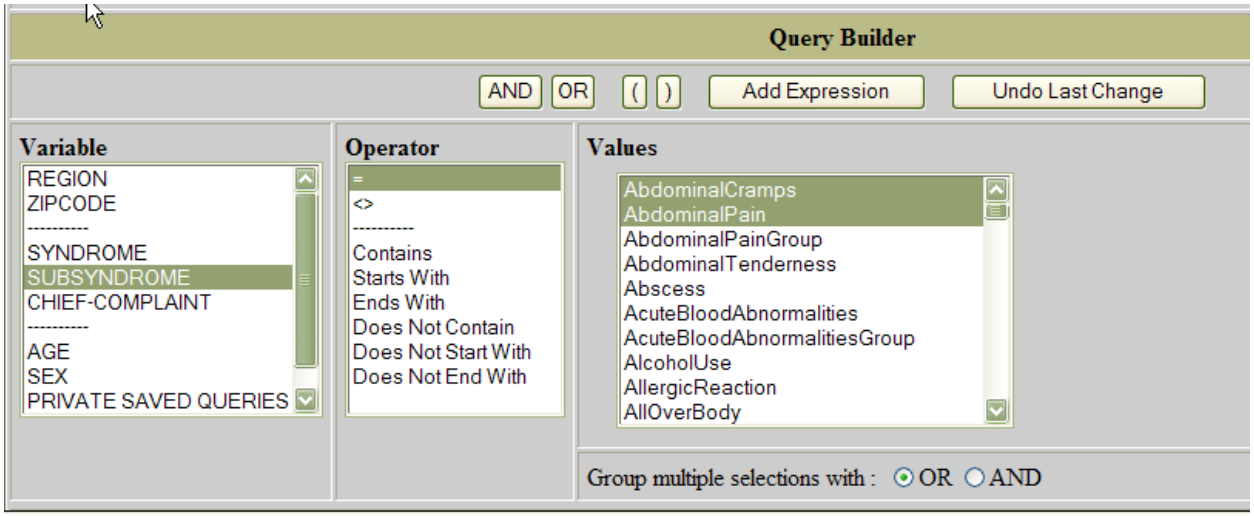

Figure 7. Select multiple values

The next step is to add this expression to the query. By clicking on the "Add Expression" button, the expression is pasted at the cursor location in the query area. One can add more expressions to this query by clicking AND or OR buttons and following the same process (Figure 8).

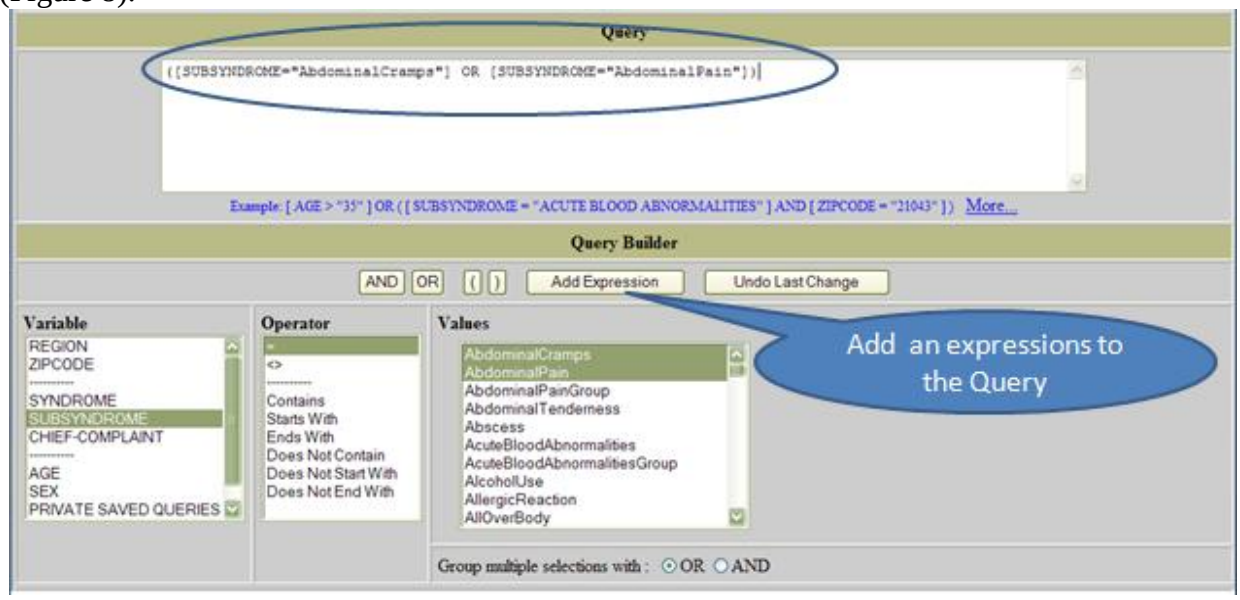

Figure 8. Add expressions to the query

The AQT helps users quickly identify limits for variables with large sets of values. Because data elements such as zip codes and ICD-9 codes have a lot of values for dropdown lists, finding a particular value in these list boxes is very cumbersome. The tool provides an intermediate step for filtering these options into a more manageable list (Figure 9). For example, if the investigators are interested in data from certain zip codes in a state, they can reduce the options by typing the first two digits of the zip code and thereby filtering the list.

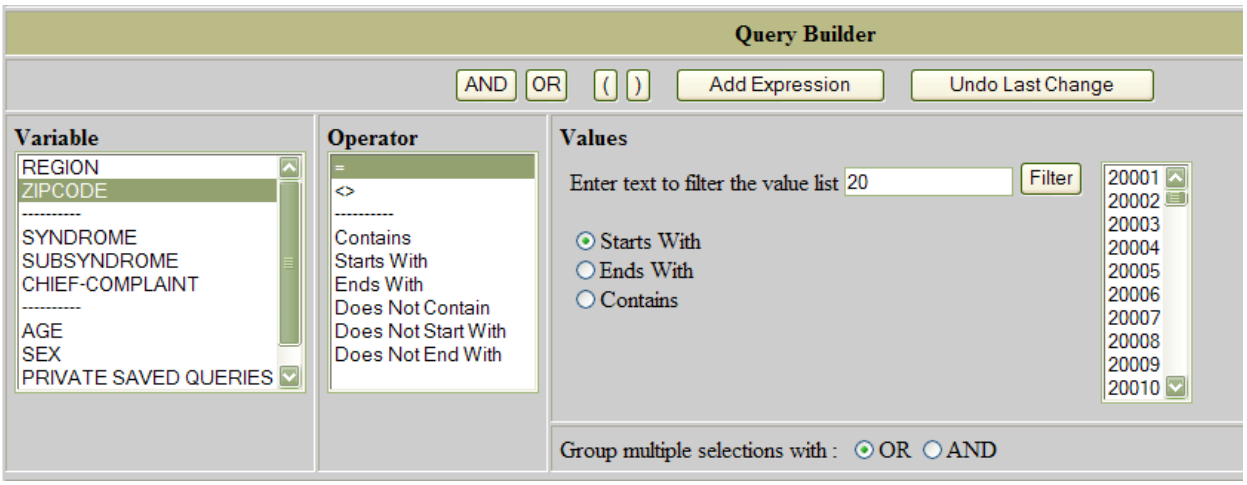

Figure 9. Filter value list

Validation

The tool will generate valid expressions and provide a mechanism to check the query expressions when a user types parts or all of them. Every time an expression is generated by the tool and the *Add Expression* button is clicked, the tool examines the entire query expression, checking it against the syntax rules. Before saving or executing the expression the AQT automatically checks the syntax and if it detects any syntax errors it will provide meaningful error messages in the message area (Figure 10). Additionally, at any point the user can click on the validate button and check the syntax.

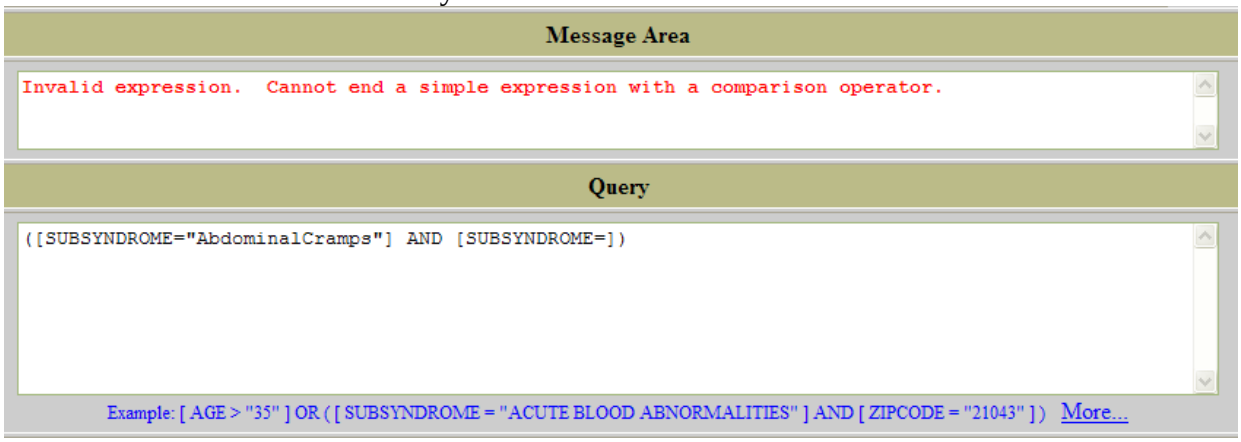

Figure 10. Validate the query expression

Save and share queries

Frequently, investigators want to execute a query over time, run the same query with different values, or use the query inside more complex queries. Similarly as all the other data elements (zip code, syndrome, region, etc.), the permanent storage and retrieval of queries (File system, database, or any other mechanism) are the responsibility of the disease surveillance system. The AQT is merely an interface to assist the investigators with their research by hiding the complexity and inner workings of the underlying data model.

Once the users define the desired query they can click on [Save Public Expression] or [Save Private Expression] buttons. If the query is valid, the screen provides an area to enter a unique name for the query (Figure 11).

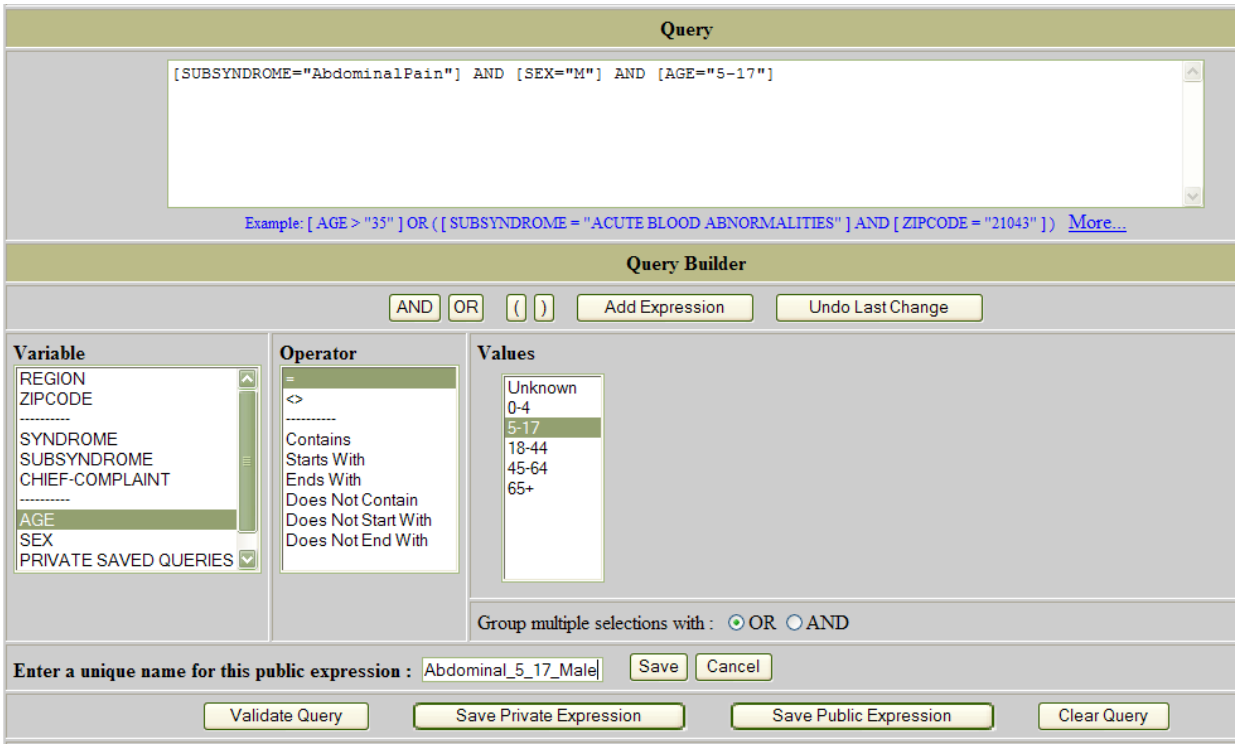

Figure 11. Saved and shared queries

If the query is successfully validated the AQT passes the name and query expression to the surveillance system. It is the surveillance system's responsibility to confirm that the query's name is unique and provide feedback to the AQT the success or failure of the save operation. Based on the feedback received the AQT provides an appropriate message in the message area.

In a collaborative environment users would like to share their findings and queries with others. Providing the capability to save and share the queries for collaborative use enables others in the user community to run these queries as they are or to make the modifications necessary to help with their own investigations. The AQT facilitates saving public queries by providing an interface similar to saving private queries (Figure 11). The surveillance system should implement the inner workings of the permanent storage and retrieval of public queries.

The next step is retrieving these saved queries. There are two options in the data element list box in the query builder section of the AQT: one option is for retrieving the private saved queries, and the other option is for retrieving public saved queries (Figure 12). Upon selection of either one, a list of corresponding queries will be presented to the users. This list includes the text of the query and the unique name given to that query. By clicking on the query name the saved query will be added to the expression in the Query area.

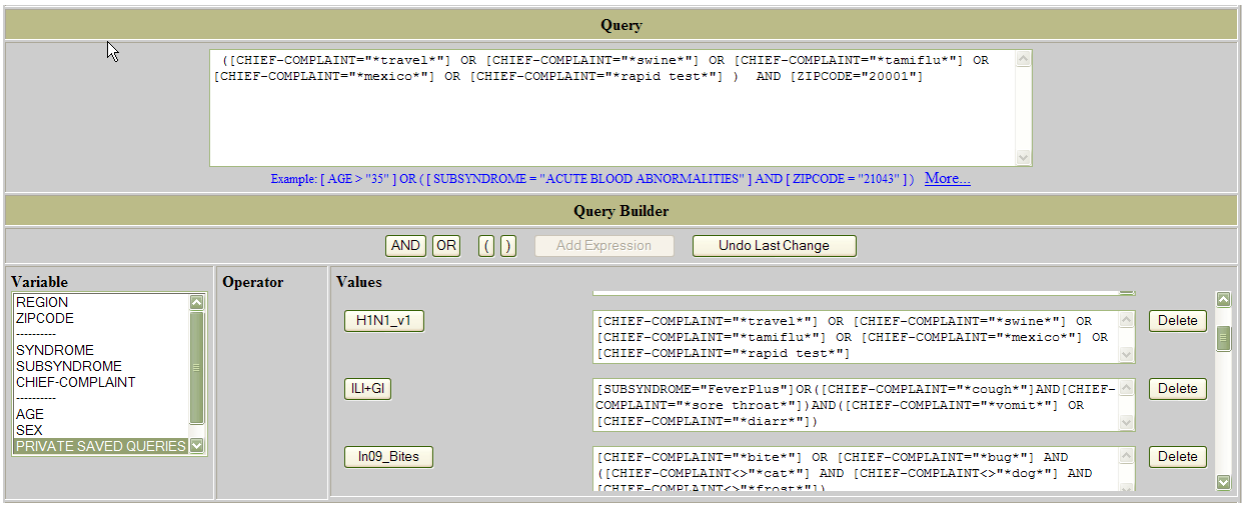

Figure 12. Retrieve saved public and private queries

At this point users can add more conditions to the same query, such as specifying a zip code, changing the value for age, etc.

Portability

The final objective of this project is for the AQT to have the capability to be used with most web-based surveillance systems. One can think of the AQT as a widget, or an add-on with some defined interfaces. The back end can be implemented in a variety of popular technologies such as .NET, Java Servlet, or any other server technology as long as it can communicate via an http protocol. The surveillance system has to provide the interfaces that supply values for the different parts of the screen, and the functionality to parse the final query text and run it against the underlying database.

Making the tool adaptable to many web-based systems requires the AQT to contain all the processing dynamically, including validating the query syntax and changing the contents of the list boxes. In a web-based environment, this means using browser components such as HTML, Cascading Style Sheets (CSS) [13], JavaScript, and the Document Object Model (DOM) [14] to implement application logic. In developing AQT, we utilized HTML, JavaScript, and AJAX (Asynchronous JavaScript and XML) and placed all the processing on the local machine to avoid any server dependency.

We used JavaScript to apply validation, data handling, and screen processing on the browser side, and AJAX for communicating with server applications. AJAX is used for creating interactive web applications and is a cross-platform technique usable on many different operating systems, computer architectures, and web browsers, because it is based on open standards such as JavaScript and XML. The intent of this technique is to make web pages more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This feature increases the web page's interactivity, speed, functionality, and usability. AJAX is asynchronous in that loading does not interfere with normal page loading.

The AQT uses AJAX calls to obtain required data for populating the different list boxes on the screen. For example, when the user selects a data source the tool calls the surveillance system, passes the selected data source, gets a list of data elements from the server (the surveillance system), and then populates the data element list box. The communication to the server is done by an AJAX call, and the JavaScript processes the returned data and populates the list.

Implementation

ESSENCE has been one of the early adaptors of AQT. Although the capability to create efficient custom queries for emergency room chief complaints data existed prior to the AQT, the query building process was cumbersome and not user-friendly. It was easy to make syntax errors while typing a query, and there was no mechanism to validate the logic of the query statement. Furthermore, while "AND" and "OR" and "ANDNOT" expressions were possible, there was no method to construct complex Boolean operations with parentheses to clarify the order of operations. The previous capability allowed the user to base the custom query on Data Source, Geography System, or Medical Grouping System, however, since the selections were not part of the query statement they could not be modified without returning to the pre-selection screens and re-starting the query process. Additionally, the original capability did not allow for querying of data beyond the fundamental chief complaints-level. The following screen shot shows the query options that were available with the original feature. A sample chief complaints query designed to capture *Influenza-Like-Illness* is shown in Figure13.

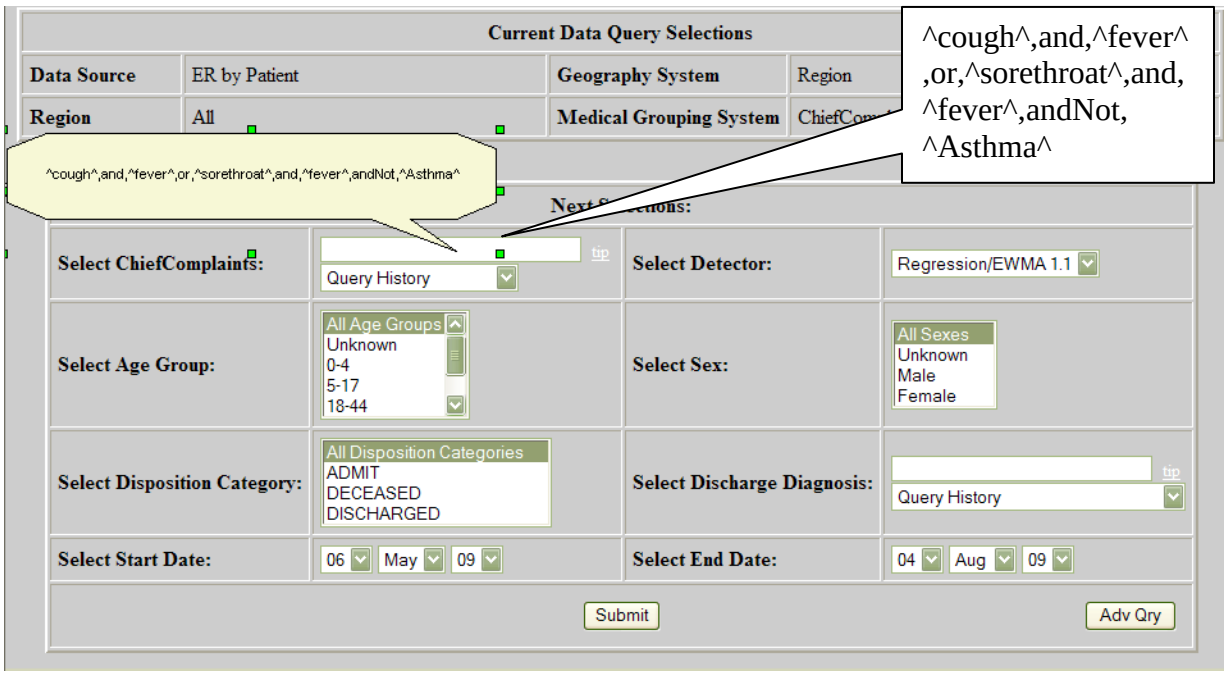

Figure 13. Influenza-like-illness query

The AQT not only contains several capabilities that were not previously available, but also provides an intuitive user-friendly interface that allows the user to build simple or highly

complex queries more easily. Two new features in the AQT are parentheses, which allow the user to clarify the order of operations, and the ability to select variables such as Region, Zipcode, Hospital, Syndrome, Sub-syndrome, Chief Complaint, Age, and Sex, as part of the query statement. This allows for easy query modifications. Additionally, the AQT lets the user query data beyond the fundamental chief complaints level into a more sensitive Sub-syndrome or Syndrome level. It also allows users to develop queries that contain combinations of chief complaints, syndromes, and sub-syndromes into one query. The query can also contain combinations of different geographies such as zipcodes and regions. This capability is not available without AQT.

During the query building process the AQT automatically validates the logic of query expression as it is created, and the user has the option to conduct a final validation prior to executing the query. This feature allows the user to quickly identify syntax errors and correct them before adding further complexity or executing the query. The following screen shot (Figure 14) shows the query options available within the AQT feature. A sample chief complaints query designed to capture *Influenza-Like-Illness* in Region_A is shown.

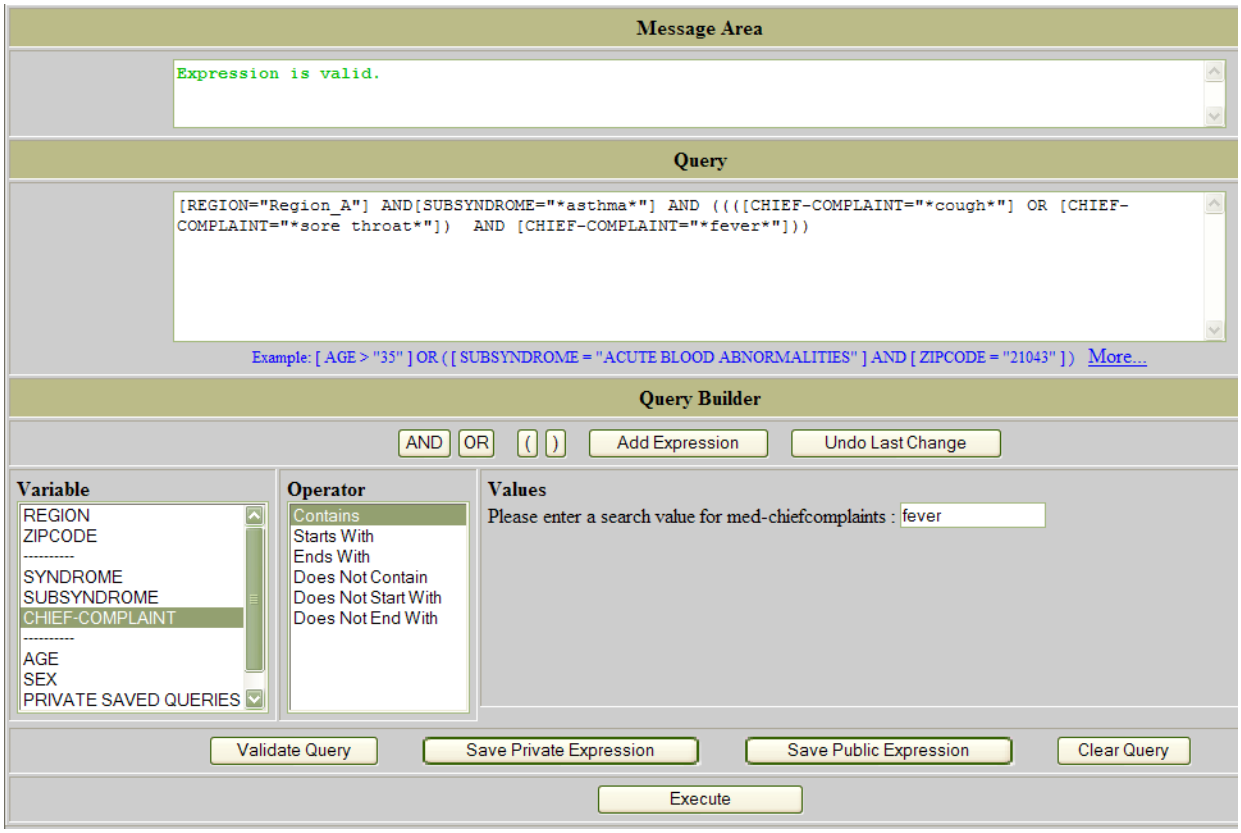

Figure 14. Influenza-like-illness query for region A

Conclusions

We believe that the AQT will provide an interface that can assist public health investigators in generating complex and detailed case definitions. The interface supports saving queries for future use and sharing queries with others in the user community. The interface is intuitive and accommodates both novice and experienced users. Finally, the AQT is a self-contained tool that can be plugged into most web-based disease surveillance systems with relative ease.

Acknowledgements

The author would like to express his appreciation to Colleen Martin and Jerome Tokars of the U.S. Centers for Disease Control and Prevention, to Sanjeev Thomas of Science Applications International Corporation, and to Wayne Loschen, Joseph Lombardo, Jacqueline Coberly, Rekha Holtry, and Steven Babin of the Johns Hopkins University Applied Physics Laboratory.

Conflict of Interest Statement

The author declares that he has no competing interests.

Summary Table

What was already known on the topic

- Early detection of known and emerging illnesses is becoming vital with the increased rate at which diseases spread world-wide.
- Most automated disease surveillance systems analyze data by syndrome and look for disease outbreaks within a community, hence overlooking the diseases that fall outside of the broad syndrome categories.

What this study added to our knowledge

- Electronic disease surveillance systems need a more sophisticated querying tool to assist public health investigators in conducting inquires across multiple data sources.
- Superior analytic flexibility through the use of data elements contained within electronic medical records enables the public health community to rapidly identify specific high risk patients.
- The Advanced Querying Tool (AQT) was designed as a flexible and simple graphical user interface (GUI) that allows users to develop, investigate, and share complex case definitions.

References

[1] World Health Organization. The World Health Report 2007 - a safer future: global public health security in the 21st century. Geneva, Switzerland, World Health Organization Press, 2007, p. 38.
[2] Vacalis T, Bartlett C, Shapiro C. Electronic communication and the future of international public health surveillance. Emerg. Infect. Dis. 1995; 1(1):34-35. http://dx.doi.org/10.3201/eid0101.950108
[3] Teutsch S, Thacker S. Planning a public health surveillance system. Epidemiol. Bull. 1995; 16(1):1-6.
[4] Farrington C, Andrews N, Beale A, Catchpole M. A statistical algorithm for the early detection of outbreaks of infectious disease. J. R. Statist. Soc. A. 1996;159(3):547-563. http://dx.doi.org/10.2307/2983331
[5] Zhong N, Zheng B, Li Y, Poon L, Xie Z, Chan K, et al. Epidemiology and cause of severe acute respiratory syndrome (SARS) in Guangdong, People‟s Republic of China, in February, 2003. Lancet. 2003; 362:1353-1358. http://dx.doi.org/10.1016/S0140-6736(03)14630-2
[6] Shih F-Y, Yen M-Y, Wu J-S, Chang F-K, Lin L-W, Ho M-S, et al. Challenges faced by hospital healthcare workers in using a syndrome-based surveillance system during the 2003 outbreak of Severe Acute Respiratory Syndrome in Taiwan. Infect. Control and Hosp. Epidemiol. 2007; 28(3):354-357. http://dx.doi.org/10.1086/508835
[7] Hart A, Hopkins C, editors. 2003 ICD9CM Expert for Hospitals, 6th ed. Salt Lake City (UT): St. Anthony Publishing; 2003.
[8] Bates D, Gawande A. Improving safety with information technology. New England Journal of Medicine. 2003; 348: 2526-2534. http://dx.doi.org/10.1056/NEJMsa020847
[9] Hillestad R, Bigelow J, Bower A, Girosi F, Meili R, Scoville R, et al. Can electronic medical record systems transform health care? Potential health benefits, savings, and costs. Health Affairs. 2005; 24(5):1103-1117. http://dx.doi.org/10.1377/hlthaff.24.5.1103
[10] Miller R, Sim I. Physicians‟ use of electronic medical records: barriers and solutions. Health Affairs. 2004; 23(2):116-126. http://dx.doi.org/10.1377/hlthaff.23.2.116
[11] Lombardo J, Burkom H, Pavlin P. ESSENCE II and the framework for evaluating syndromic surveillance systems. MMWR Morb Mortal Wkly Rep. 2004 Sep 24; 53(suppl.):159-165.
[12] Standardizing Clinical Condition Classifiers for Biosurveillance. Available at http://www.cdc.gov/Biosense/files/PHIN2007_SubsyndromesPresentation-08.22.2007.ppt
[13] McFarland DS. CSS: the missing manual. Sebastopol (CA): O‟Reilly; 2006.
[14] Flanagan D. JavaScript: the definitive guide, 5th ed. Sebastopol (CA): O‟Reilly; 2006.

Author

Mohammad R. Hashemian, MS (Computer Science)
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723 USA
mohammad.hashemian@jhuapl.edu
Fax Number: 443-778-3686