A jQuery-like Platform for Standardized Dataset Processing Logic

Marat Zhanikeev*
School of Management, Tokyo University of Science, Japan
**Corresponding Author: maratishe@gmail.com*

Received 25 February 2020; received in revised form 18 May 2020; accepted 3 June 2020

Abstract

The processing logic for datasets (including bigdata) cannot be easily standardized and shared among researchers. This is primarily due to the lack of a modeling language which can cover a wide range of selection and processing logic. For example, UML, even in its database schema sections, cannot fulfill this goal. This paper first notices the similarities between processing DOM elements in web applications and tuples in datasets, and the proposes to borrow the jQuery notation for dataset processing. Most concepts, such as (1) selectors, (2) methods, (3) processing chains, (4) callbacks for asynchronous (time consuming) tasks, and others enjoy a direct mapping to the proposed functionality. This paper offers several examples which emphasize the usefulness of the proposed method and describes a recent non-browser implementation which focuses on performance optimization.

Keywords: Dataset processing, bigdata, jQuery, selectors, noSQL databases, web survey data

1. Introduction and Problem Statement

Let us start by describing a practical situation. Several people work on the same data that can come from a web survey (Chambers & Skinner, 2003) or can even be bigdata. Since all the involved people work on the same survey data, there exists a need for one person to explain how the data are processed. In a similar fashion, one might want to recreate someone else's results from a description of the undertaken processing steps. The larger issue of accountability aside, this situation is a perfect example when a common modeling language can be helpful. For example, if all people in a team know how to create and interpret UML action diagrams (Larman, 2002), descriptions in this form are readily exchangeable across members.

When it comes to processing raw data from web surveys, in spite of a strong methodological background (Bethlehem & Biffignandi, 2012; Biemer & Lyberg, 2003), there is currently no standard way to describe the process or save it in a format which can be read, understood, and re-implemented by other people. The same is applicable to bigdata storage and analysis.

It can be argued that an algorithm can serve the intended purpose, however, as will be shown below, the proposed method is much more efficient in describing the processing logic. Moreover, the proposed modeling language has the advantage of a simple visual representation. The list of features proposed in this paper is as follows:

- a model for visualization and scripting processing logic for an arbitrary dataset (dataset from this point on) is proposed;
- the model is simple and has only two components -- selectors and methods;
- the modeling language is very powerful regardless of the simplicity, for example, hierarchical grouping of raw survey data is possible with such concepts as sets, subsets, and parent sets;
- the proposed model can implemented as a web application and has built-in visualization and animation functionality, it is also implemented as a standalone performance-optimized script interpreter;
- the proposed model is already implemented based on jQuery (Lindley, 2009) -- a language well known in web application development community and, incidentally, the best match for the intended usecase;
- the model is extremely easy to extend by adding new methods to jQuery, in fact, it is even possible to extend selectors.

This paper is written as follows. After reviewing existing UML and related modeling possibilities and establishing terminology, this paper discusses unique features shared across all datasets. Parallels are then found between jQuery features and the needs of dataset processing, which makes it possible to formulate the proposal using jQuery terminology. The proposal is then formulated first in scripting and then in visual components. Example code in this paper originates from real experience by this author.

2. Related Work

Universal Modeling Language (UML) (Larman, 2002) is currently the default modeling language across many areas of research. Diagramming concepts like usecases, state and/or action diagrams, and others, are the most commonly used UML tools. None of these diagrams are applicable to dataset processing. UML was at some point extended to cover database design, which remains in common use today. Such diagrams define table structures, data types of individual columns in tables, relations/dependencies across tables, etc. Although datasets often come in table form (noSQL form is rapidly becoming more common), in which case database design diagrams are not applicable. There are many less common diagrams which include state-and-transition diagrams, data flow diagrams, dependency graphs or graphs in general. Again, none of these are applicable.

Although survey data processing (Patterson, 2002)(Sills, 2002) does not share common features with any of the diagrams explained above, some similarity with existing processes can be offered. Survey data processing shares common features with *the processing done using SQL*, which is mostly about querying databases and processing the output of queries (Sills, 2002). However, it should be noted that even SQL lacks functionality necessary for processing datasets.

In recent years, a new kind of rigid statistical processing called *data streaming* has been proposed and remains an active research until now (Zhanikeev, 2014). Data streaming offers an advanced modeling toolkit, by rigidly defining memory footprint, processing workload, and, more importantly, the data structure produced by the analysis. The latter is referred to as a *sketch* and is, when described in detail, can serve as a medium when sharing the processing method among researchers (Zhanikeev, 2014). However, in data streaming, only sketches are rigidly defined, while the processing logic itself can be interpreted and implemented differently by different researchers. Note that this paper solves this very problem by allowing to share both data structure and processing logic via the proposed modeling method.

Data streaming is not specific to survey data analysis and is, in fact, often applied to big data processing (hence the memory footprint and efficiency analysis). Traditionally, modeling concepts in bigdata processing are extremely primitive. For example, *key-value* pairs and *counting* of their occurrence in data is a standard way to process big data (Rowstron, 2012). Naturally, this methodology is too simple to share among researchers. However, in recent years, there is research on richer data structures and processing logic in big data. For example, performance when processing schema-less (the noSQL method) data tuples using Lucene was analyzed and proved efficient (Gao, 2011). In (Das, 2010), an attempt was made to implement statistical processing scripts written in the R language on a big data processing platform. Lucene, again, suffers from the problem of rigidly specifying data structure but not the processing logic. The R method, however, is very close in spirit to this method as it makes it possible to rigidly specify both structure and processing parts. R, however, has another problem, which is that it deals primarily with numeric data (both in input, output and processing proper), while this paper defines flexible selectors which can work with both numbers and textual data.

As a summary of the above overview, this paper is original in that it brings together several elements which currently exist in unrelated tools and implements it in an easily recognizable form, e.g. the jQuery language (Bibeault, 2010). Specifically, this proposal offers a rigid data structure (as in data streaming or Lucene) and standardizes processing logic (as in R), while also offering a flexible way to select input data for each processing stage using jQuery selectors, which far exceeds the search function in Lucene.

3. Terminology

Datasets in this paper refer to raw data that needs to be processed to become useful. The modeling instrumentation proposed in this paper refers to the actions during this processing stage. This paper will use the database term *tuple* to refer to one row of data. Multiple tuples are referred to as *sets*. There can also be *subsets*, while the term *parent* represents the ability to return to the larger (parent) set from its subset. Given the chain-processing functionality presented in this paper, set-subset-parent cycle is an essential feature.

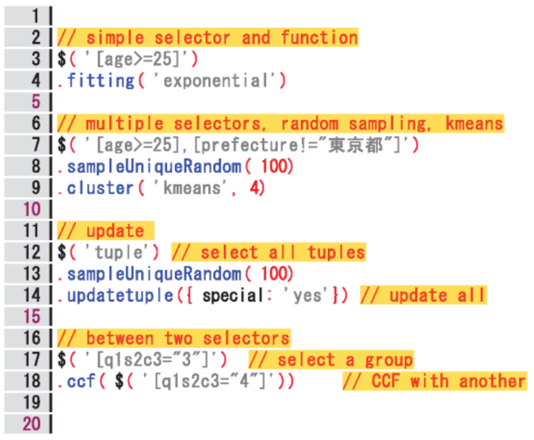
The term *selector* refers to a set of conditions which are used to select a set of tuples. In conventional software selectors can be implemented as *if... then...* constructs but in this paper a much simpler construct will be presented. The term *method* refers to operations performed on sets of tuples. Several examples will be presented in this paper.

4. The Javascript/jQuery Model

It turns out that there already is a programming language which completely matches the unique features and requirements listed in the previous sections. The language is *jQuery* -- an open source built-up on top of Javascript. The natural habitat for jQuery is the client side of web applications, i.e. the browser. The language currently supports all the major browsers, namely Firefox, Opera, Chrome, Internet Explorer, etc.

jQuery is extremely popular today and is used much more in professional web programming than its underlying Javascript. The main reason for this is the extreme simplicity of the language. This simplicity is attained the hard way by the language through creating a high level abstraction over the plain Javascript. The abstraction, in a nutshell, is exactly what is needed for processing of survey data -- selectors and methods (jQuery Selectors, 2019).

The main purpose of jQuery is to be able to select multiple HTML tags (DOM elements) and apply the same processing logic to each of them in turn. For example, one might want to process the table in such a way that all even lines in a table have one background color and all odd lines have another. With jQuery this can be done in two lines of code using *:odd* and *:even* selectors. If one uses *parent()* to return to the main set, the entire processing logic can actually be implemented in a single uninterrupted chain. Drawing a parallel, complex selectors are akin to snowball sampling (Handcock & Gile, 2011) in dataset processing.



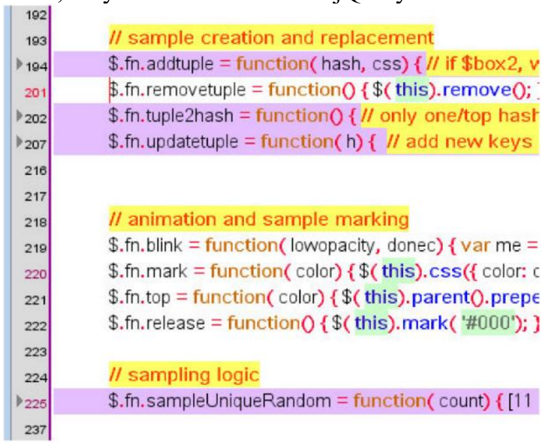
```

// simple selector and function
$( '[age>=25]')
.fitting( 'exponential')

// multiple selectors, random sampling, kmeans
$( '[age>=25],[prefecture!="東京都"]')
.sampleUniqueRandom( 100)
.cluster( 'kmeans', 4)

// update
$( 'tuple') // select all tuples
.sampleUniqueRandom( 100)
.updatetuple({ special: 'yes'}) // update all

// between two selectors
$( '[q1s2c3="3"]')  // select a group
.ccf( $( '[q1s2c3="4"]'))     // CCF with another


```

Figure 1: Some Examples of Dataset-like Functionality Implemented via Selectors and Functions

Figure 1 shows some example selector-method pairs, specific to dataset processing. For example, in the first block, all tuples with attribute age equal or greater than 25 are selected, and then exponential fitting is performed on values from all tuples. The second example in figure 1 has two selectors which are merged using AND logic, then 100 tuples are randomly selected and clustered using the k-means method with 4 clusters.

The last example in figure 1 exhibits a method which works on two tuples sets. One set is created by the selector before the method and the other selector is used as method argument. The method can then use two sets of values to calculate, for example, cross-correlation function or even more advanced features (Patterson, Dayton, & Graubard, 2002).

As is show in figure 2, current implementation of the proposed framework has only few methods (all lines starting from $.fn are methods), but it will be extended in the future to contain the standard analysis kit. In the meantime, a few words on the internal structure of methods are necessary. While loops are not part of the proposed framework, they exist in internals of jQuery methods.



```

    // sample creation and replacement
    $.fn.addtuple = function( hash, css) { // if $box2, v
    $.fn.removetuple = function() { $( this).remove();
    $.fn.tuple2hash = function() { // only one/top hash
    $.fn.updatetuple = function( h) {  // add new keys


    // animation and sample marking
    $.fn.blink = function( lowopacity, donec) { var me =
    $.fn.mark = function( color) { $( this).css({ color: c
    $.fn.top = function( color) { $( this).parent().prepe
    $.fn.release = function() { $( this).mark( '#000'); }

    // sampling logic
    $.fn.sampleUniqueRandom = function( count) { [11

```

Figure 2: Current jQuery Extension for Survey Data Processing

Selectors by default return sets of tags, so, each jQuery extension has to create the logic for processing each tag separately. The common way to do it in jQuery is to call. Each (callback) on the selected set, and wait for jQuery to call callback method for each tuple.

5. Implementation

The first implementation of the proposal was written actually using the jQuery language and, therefore, was running in the browser. However, with the need for nested contexts and deeper data structures than the flat *key-value* tuples implemented as HTML tag attributes, it was decided that a standalone implementation of the processing engine was required. The current implementation is a standalone script interpreter which implements all the jQuery features (refer to (Bibeault, 2010) for the complete list of jQuery capabilities) in a simplistic command line environment.

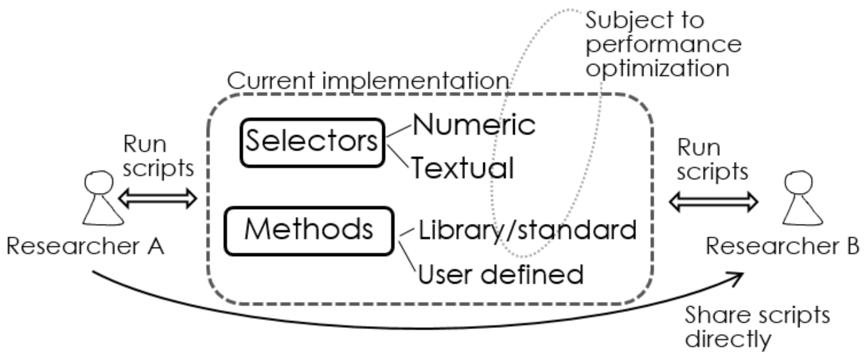


Figure 3: Current Implementation of the Engine for Processing Data Scripts

The need to migrate to a standalone implementation also came from the requirement to optimize performance in several parts of the proposed scripting engine. Figure 3 shows the currently adopted design. The focus is still on the idea that researchers can share processing logic directly, but this time they can use the new implementation to run their scripts and verify output independently. As in jQuery, the two main modules are selectors and methods. Except for customized, user-defined selectors, they exist in either numeric or textual form, which can be easily optimized for performance. For example, lookup time in textual selectors is improved by having an ordered list of values for each attributed used in the script (run once and cached for future repeated use). Numeric attributes are optimized by using fixed lengths (in bytes) and caching intervals in values.

As far as methods are concerned, only a limited list of standard processing methods are offered in the optimized form as part of the scripting engine's library. User-defined method are unknown in advance and thus cannot be optimized. Hopefully, the volume provided by the library will grow with time, if the interest to this proposal in research community is sufficient.

6. Conclusion

The framework proposed in this paper makes it possible to share between researchers not only datasets but also processing logic, which is achieved by proposing a selector-method framework implemented on top of the jQuery language (Bibeault, 2010) traditionally used in web applications. The proposal is unique in that it not only rigidly defines the data structures used in processing as is found in Lucene (Gao, 2011), R (Das, 2010) and data streaming (Zhanikeev, 2014) environments, but also can be used to share processing logic (only found in R today). A feature unique to this proposal comes from jQuery and is the first known example when *selecting* input data using jQuery-like selectors is part of the rigidly defined processing logic.

Early implementation of the proposal was written actually using jQuery and was running in web browser, but current implementation comes in form of a standalone scripting engine, as was explained in Section 5. In future work, the framework will be developed further by, for example, increasing the list of built-in processing methods, this making it possible to offer them in a performance optimized form. There is also a work-in-progress to extend selectors to include advanced search functionality not found in traditional jQuery but which can be useful for statistical data analysis. The progress on these elements will be reported in future publications on this topic.

References

Bethlehem, J., & Biffignandi, S. (2012). *Handbook of web surveys*. John Wiley and Sons.

Bibeault, B., & Katz, Y. (2010). *jQuery in action* (Second Edition). Manning.

Biemer, P., & Lyberg, L. (2003). *Introduction to survey quality*. John Wiley and Sons.

Chambers, R., & Skinner, C. (2003). *Analysis of survey data*. John Wiley and Sons.

Das, S., Sismanis, Y., Beyer, K. S., Gemulla, R., Haas, P. J., & McPherson, J. (2010, June). Ricardo: integrating R and Hadoop. In *proceedings of the 2010 ACM SIGMOD international conference on management of data* (pp. 987-998).

Gao, X., Nachankar, V., & Qiu, J. (2011, November). Experimenting lucene index on HBase in an HPC environment. In *proceedings of the first annual workshop on high performance computing meets databases* (pp. 25-28).

Handcock, M., & Gile, K. (2011). On the concept of snowball sampling. *Journal of sociological methodology*, *41*(1), 367-371.

jQuery Selectors (2019). http://api.jquery.com/category/selectors

Larman, C. (2002). *Applying UML and patterns: an introduction to object-oriented analysis and design, and the unified process*. Prentice Hall.

Lindley, C. (2009). *jQuery cookbook: Solutions and examples for jQuery developers*. O'Reilly Media.

Patterson, B., Dayton, C., & Graubard, B. (2002). Latent class analysis of complex sample survey data: Application to dietary data. *Journal of the American statistical association*, *97*(459), 1-21.

Rowstron, A., Narayanan, D. (2012). *Nobody ever got fired for using hadoop on a cluster.* 1st International Workshop on Hot Topics in Cloud Data Processing (HotCDP).

Sills, S. J., Song, C. (2002). Innovations in survey research: An application of web-based surveys. *Social science computer review*, *20*(1), 22-30.

Zhanikeev, M. (2014). *Replayable bigdata for multicore processing and statistically rigid sketching.* Internet Conference, Hiroshima, Japan.

About Author

Marat Zhanikeev received M.S. and Ph.D. in Global Information and Telecommunications Studies from Waseda University in Tokyo, Japan, in 2003 and 2007, respectively. His research interests include network measurement, network monitoring, and network management, but also extend to practical applications related to these topics as well as non-traditional applications of information technology in general. He is presently an Associate Professor at Tokyo University of Science and is a Regular Member of IEEE, IEICE and IPSJ.