xAPI Reporting Tutorial

Tutorial Contents

xAPI, LRS, and xAPI Wrapper

To-do: brief overview of xAPI Wrapper and related software

xAPI Dashboard

Prerequisites for developers

- Web browser (preferably Chrome)
- Suitable text editor with syntax highlighting
- Experience with either JavaScript or any other C-like language
- Basic xAPI knowledge desired, but not required
- Basic SQL knowledge is desired, but not required

Prerequisites for other participants

- Web browser (preferably Chrome)
- Basic xAPI knowledge desired, but not required
- Basic SQL knowledge is desired, but not required

Introduction

The xAPI Dashboard is a Javascript library used to extract and visualize aggregate data from Experience API statements. It is composed of two parts. The first part, the Collection class, allows developers to run SQL-like queries, filters, and aggregations over xAPI data. The second part, the XAPIDashboard class, can generate numerous types of charts and visualizations based on that aggregated xAPI data.

examples/livedata.html file contains examples of the types of visualizations we'll be building. It currently pulls live data from the ADL LRS and is able to tell us:

- The number of times each verb was used in the past 30 days in the form of a bar chart
- Rate of new statement activity over the past 30 days in the form of a line chart

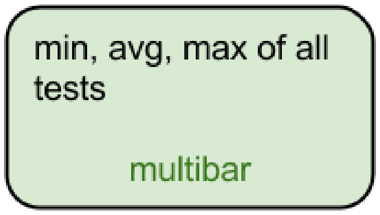
- The 10 most active actors over the past 30 days

Tutorial Steps

1. Let's build our own reporting dashboard! For this tutorial, we'll be asking relevant questions about data stored in our local LRS.
   a. **Question #1**: It'd be interesting to learn a bit more about class assessments. What are the minimum, average, and maximum scores achieved on each test?
   b. **Questions #2**: Given a test, what is the number of students in each traditional letter grade range (90-100, 80-90, ...)?
   c. **Question #3**: Given a score range and a test, what are the names of the students in that range and what were their exact scores on that test?
2. Download all inclusive release package
   a. https://github.com/adlnet/xAPI-Dashboard/
3. If you're a developer and want to program in parallel, you may open the *template.html* file in both a web browser and a text editor. This file
4. Instantiate and configure xAPI wrapper and dashboard objects
   a. To-do: elaborate
5. Use dashboard's fetchAllStatements convenience function to retrieve all xAPI statements
   a. dashboard.fetchAllStatements(query, callback)
6. **CHECKPOINT 1**
7. Let's create our first chart! A chart created using the XAPIDashboard class must be given configuration options that determine its look and behavior. Of these options, groupBy and aggregate are the most important and are both required.
   a. pre is used to filter out undesirable elements from the dataset before performing any operations
   b. groupBy is analogous to SQL's GROUP BY and it functions as its name implies. Given a dataset (random people), you can specify on which property to perform a groupBy (occupation), and groupBy will generate groups of people that share the same property value (teachers, engineers, bankers, etc.).
   c. The aggregate functions built into the Dashboard are also analogous to many of the ones present in SQL. Once a groupBy has been performed, an aggregate function is run and will generally produce a single value per group (the exception to this is multiAggregate, which is only used for a MultiBarChart). For example:

        i. ADL.count() will count the number of statements in each group
        ii. ADL.min(*property*) will return the minimum value of a given property per group
    d. post is used to massage the data **after** it has been aggregated. During this step, you're able to limit size of the aggregate dataset, order by a certain property, or perform any other arbitrary operations.

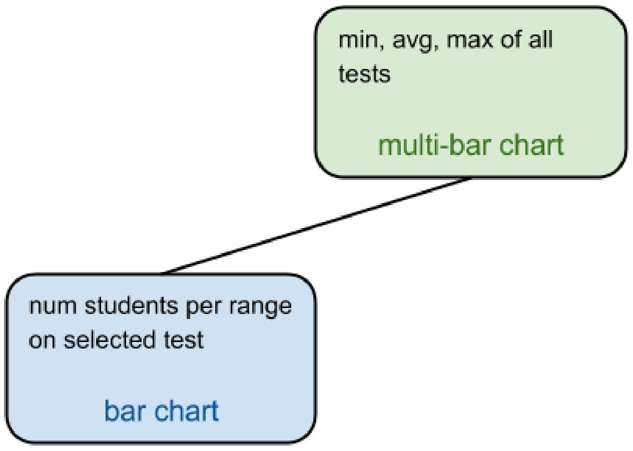
8. Determine which type of chart is best for displaying the results of **Question #1**
    a. This question is really three different questions in one. Minimum, average, and maximum are all operations that can be performed on the exact same dataset and as such, it makes sense to combine their output into a single multi-bar chart
9. Use dashboard's built-in *createMultiBarChart* function and determine the proper options to pass into it
    a. Options:
        i. groupBy: the name of each test
        ii. pre: retrieve only statements that actually have a score, order by test name
        iii. aggregate: multiple functions - min, average, max
        iv. container: the svg element on which this chart will be drawn
    b. Call draw on the newly created instance of MultiBarChart
10. Refresh the page and you should see an interactive multi-bar chart that is also very aesthetically pleasing. Below is an image of our current (and very simple) chart hierarchy:



11. CHECKPOINT 2

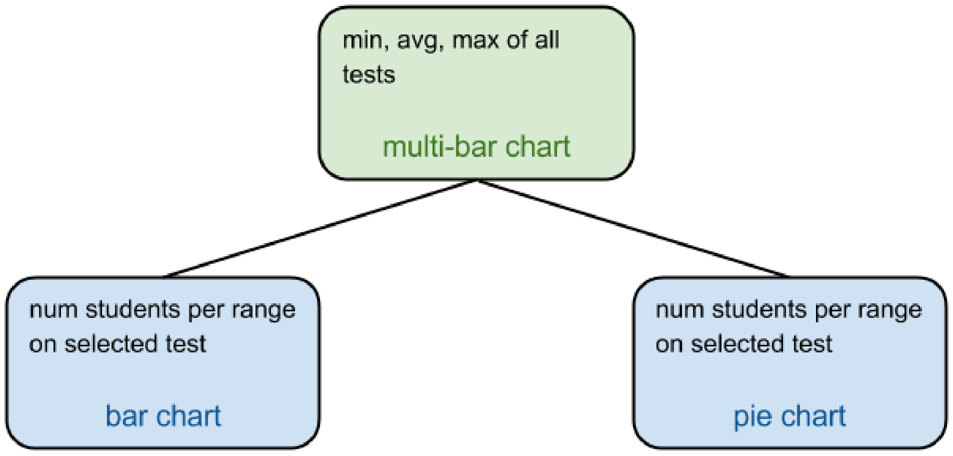
12. To answer **Question #2**, we will have to drill down into the details of the newly created chart. The way we do this with the dashboard is to assign a "child" chart to our existing multiBarChart.

13. Determine which type of chart is best for displaying the results of **Question #2**.
    a. Count is a single operation that produces a single value per group. A bar chart is suitable for representing this kind of data. Just for fun, we will also create a pie chart.
14. Use dashboard's built-in createBarChart function and determine the proper options to pass into it
    a. groupBy: each test needs to be placed in a group with other tests containing the same score
    b. range: because there can be a very large number of groups (one for each possible score), we will limit the number of groups by using group ranges (30-35, 35-40, …, 90-95, 95-100)
        i. min: 30, max: 100, increment: 5
    c. pre: only include statements that actually have a score and use the event argument to restrict statements to only those containing the same name as the clicked test
    d. aggregate: count
    e. container: the svg element on which this chart will be drawn
15. Refer back to the multiBarChart and add a "*child*" field to its options. Child should be an array of references to the charts that we want to set as children of the current chart. After adding the child chart, our hierarchy looks like this:


min, avg, max of all tests
multi-bar chart
num students per range on selected test
bar chart

16. CHECKPOINT 3

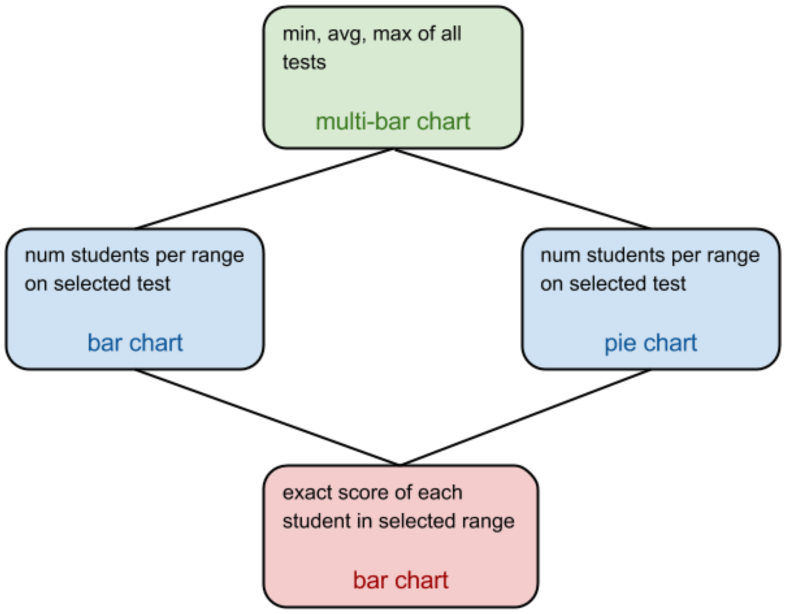
17. You may now repeat steps 11 and 12 using the dashboard’s built-in *createPieChart* function to create a pie chart instead of a bar chart. Add it as a second child to the multi-bar chart.
18. Our hierarchy now looks like this:

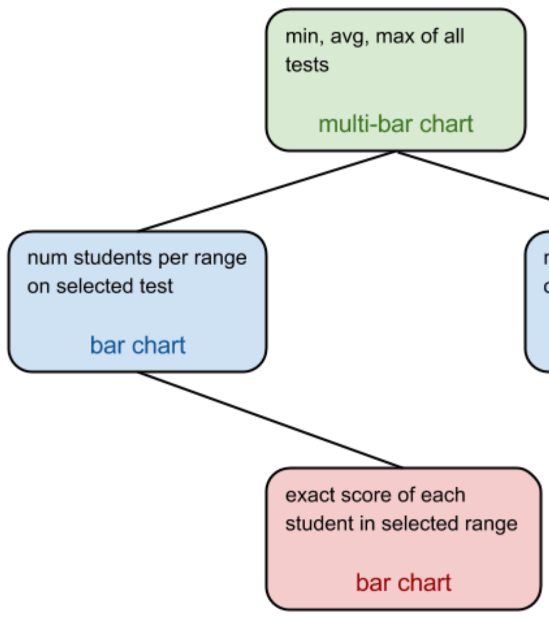

min, avg, max of all tests
multi-bar chart
num students per range on selected test
bar chart
num students per range on selected test
pie chart

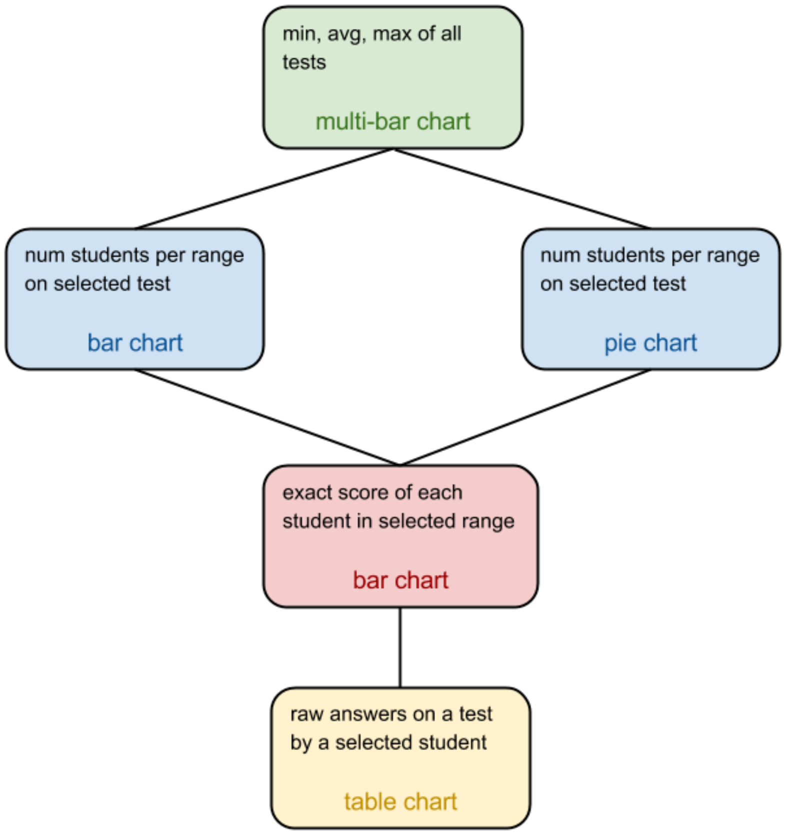
19. CHECKPOINT 4

20. As with **Question #2**, answering **Question #3** will require us to allow the user to “drill down” even further.
21. Determine what type of chart is most appropriate for displaying individual scores
    a. A table would be optimal, but our library doesn't do that yet. So we'll use a bar chart to show the same data
22. Use the dashboard’s *createBarChart* function and determine the proper options to pass into it
    a. groupBy: group by actor name
    b. pre: include only statements with a score that is within the range that was clicked on from the test currently being viewed by the user. Note that charts only receive event information from its immediate parent, so this chart will only receive range information. Use a different approach to determine which test is currently being viewed.
    c. aggregate: select score (further aggregation isn’t necessary because each bar maps to exactly one statement)
    d. post: make sure there are not too many bars to read, and sort them

e. customize: rotate the x axis labels (student names) 45 degrees so they don't overlap


min, avg, max of all tests
multi-bar chart
num students per range on selected test
bar chart
num students per range on selected test
pie chart
exact score of each student in selected range
bar chart


min, avg, max of all tests
multi-bar chart
num students per range on selected test
bar chart
num students per range on selected test
pie chart
exact score of each student in selected range
bar chart


min, avg, max of all tests
multi-bar chart
num students per range on selected test
bar chart
num students per range on selected test
pie chart
exact score of each student in selected range
bar chart
raw answers on a test by a selected student
table chart

ElasticSearch and Kibana