CSCI 621
GROUP 3 - H2 Implementation Assignment 3

Abhishek Shah, Harsh Bhansali, Miloni Sangani, Rasika Sasturkar

INTRODUCTION

- For implementing an enhanced feature in terms of query processing and optimization, we decided to implement a custom aggregate function: Quartile to the H2 database. Aggregate functions are functions that combine multiple records and then perform the function and return a single value as result.

- We decided Quartiles because it is a very commonly used statistical method for retrieving meaningful information from the dataset.

- Quartiles are used to summarize a group of numbers from the given data on which we use this function. Instead of looking at a big list of numbers, we look at just a few numbers that give us an insight of what's going on with the data.

- Quartiles split up the data into four equal size groups. Out of them, the three main quartile are as follows:

    - The first quartile (Q1) is defined as the middle number between the smallest number which is the minimum and the median which is the middle of the dataset. Q1 is the number that separates the lowest 25% of the data from the remaining 75% of the data.

    - The second quartile (Q2) is the median of the data set. Q2 is the number in the middle of the group.

    - The third quartile (Q3) is the middle number between the median and the highest value which is the maximum value in

the dataset. Q3 is the number that separates the lowest 75% of the data from the highest 25% of the data.

- Quartiles divide the number of data points evenly. Therefore, the range is not the same between all three quartiles and this range is known as the interquartile range. Quartiles are generally used to calculate the interquartile range, which is a measure of variability around the median.

- The interquartile range is simply calculated as the difference between the first and third quartile that is Q3–Q1. In effect, it is the range of the middle half of the data that shows how spread out the data is.

- The significance of Quartiles is that the upper and lower quartiles can provide more detailed information on the location of specific data points, the presence of outliers in the data set, and the difference in spread between the middle (50%) of the data and the outer data points.

ENHANCED FEATURE IMPLEMENTATION

QUARTILE is an aggregate function that calculates the quartiles among all the values of a column in the table and returns an array [Q1, Q2, Q3] of type “double” where the first position represents the value of the first quartile, the second position represents the value of the second quartile and the third position represents the value of the third quartile.

Files added : Quartile.java
File path : /h2database/h2/src/main/org/h2/api/Quartile.java

In H2, the user can define their own custom aggregate function by implementing the AggregateFunction interface i.e overriding its methods like init, add, getResult, getType. The Quartile.java file was added to the org/h2/api path. Then we execute some commands in the H2 console to

create an aggregate function with the name we want to define it with. This is shown below in the screenshots of the executions.

```java
@Override
public Object getResult() throws SQLException {
    // TODO Auto-generated method stub
    int[] nums = numbers.stream().mapToInt(Integer::intValue).toArray();
    Arrays.sort(nums);
    if (nums.length % 2 == 0) {
        result[1] = ((double) nums[nums.length / 2] + (double) nums[nums.length / 2 - 1]) / 2;
        int splitAt = nums.length / 2;
        result[0] = calculateQuartile(subArray(nums, beg: 0, splitAt));
        result[2] = calculateQuartile(subArray(nums, splitAt, nums.length));
    } else {
        result[1] = (double) nums[nums.length / 2];
        int splitForEndFQ = nums.length/2;
        int splitForBeginTQ = nums.length/2 + 1;
        result[0] = calculateQuartile(subArray(nums, beg: 0, splitForEndFQ));
        result[2] = calculateQuartile(subArray(nums, splitForBeginTQ, nums.length));
    }
    return result;
}

public static <T> int[] subArray(int[] array, int beg, int end) {
    System.out.println(Arrays.toString(Arrays.copyOfRange(array, beg, end)));
    return Arrays.copyOfRange(array, beg, end);
}

private Double calculateQuartile(int[] nums) {
    if (nums.length % 2 == 0)
        return ((double) nums[nums.length / 2] + (double) nums[nums.length / 2 - 1]) / 2;
    else
        return (double) nums[nums.length / 2];
}
```

Fig 1: Quartile class in Quartile.java

ENHANCED FEATURE IN ACTION

For adding the aggregate function in the H2 database, we need to execute the following command in the H2 interface

create aggregate QUARTILE for "org.h2.api.Quartile";

Run | Run Selected | Auto complete | Clear | SQL statement:

```
create aggregate QUARTILE for "org.h2.api.Quartile";
```

```
create aggregate QUARTILE for "org.h2.api.Quartile";
Update count: 0
(4 ms)
```

Now, we will consider the following database with Employee table that has the following fields : id, name and salary

Run | Run Selected | Auto complete | Clear | SQL statement:

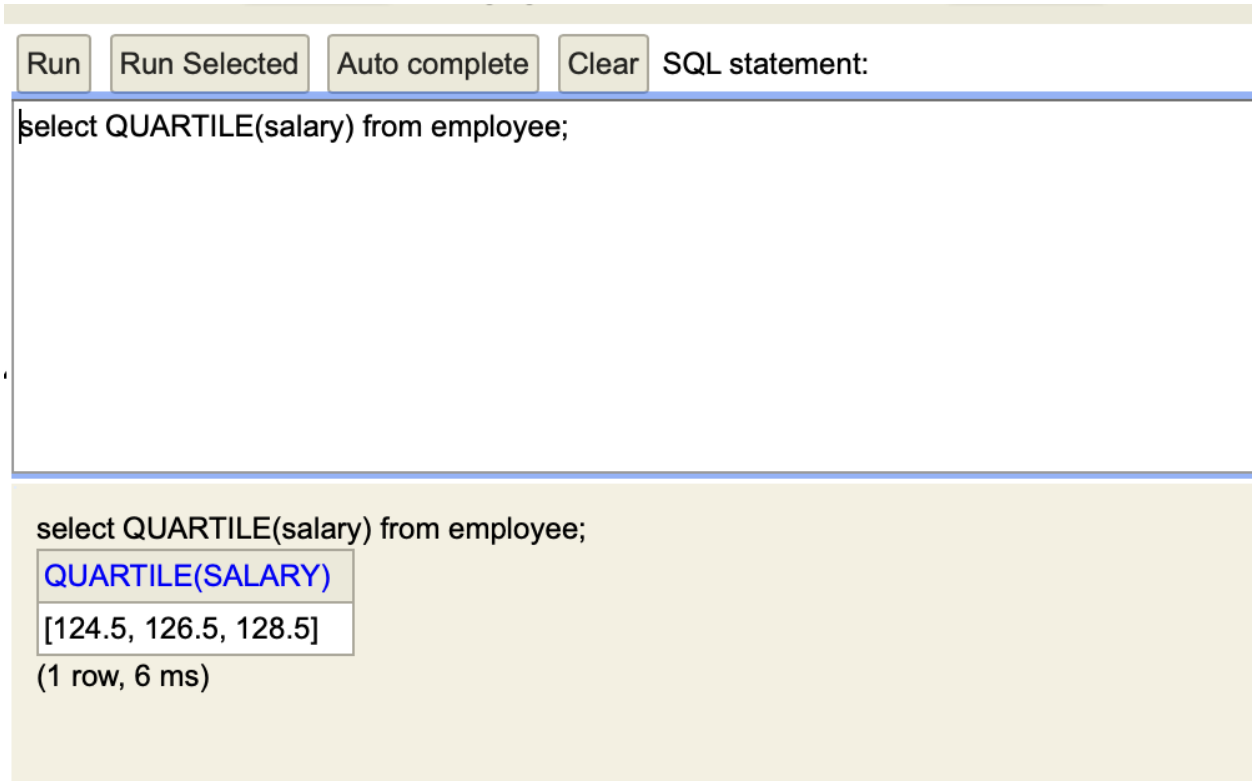
```
SELECT * FROM employee;
```

SELECT * FROM employee;

ID	NAME	SALARY
1	A	123
2	B	124
3	C	125
4	D	126
5	E	127
6	F	128
7	G	129
8	H	130

(8 rows, 17 ms)

Displaying the results of the aggregate function results



WORKLOAD DISTRIBUTION

Everyone

All team members read up about aggregate functions that already exist in H2. Then we brainstormed about the new aggregate function to be added to the source code which would enhance the database. We came up with the Quartile aggregate function.

Abhishek Shah

Worked primarily on the documentation of the assignment i.e writing of the introduction and explaining the reason why we chose Quartile as an aggregate function to add. Also helped with the test cases and cross checking the answers returned by the function.

Rasika Sasturkar
Worked on executing the final implementation and taking screenshots for the documentation. For that she created the aggregate function on the H2 console, populated the database with test data and ran many test cases to ensure that the Aggregate function was returning correct answers.

Miloni Sangani
Helped with code checking and writing of the documentation of the assignment. Explained the steps to implement the enhanced feature in the H2 source code and also write a description of what is happening in each screenshot of execution.

Harsh Bhansali
Worked mainly on implementation of the code in Quartile.java file. The methods like calculatedQuartile, getResult, add and init were coded by him. Helped in the documentation and also on the initial idea of implementing Quartile as an aggregate function.