Databases for Developers - Assignment 2

Dora Dimitrova   Martin Vestergaard   Anders Kalhauge

Spring 2021

The following tasks should be handed in on Peergrade.

Task 1 - investigation

Produce a small writeup (around 300 words) answering the following questions.

1. What is point of NoSQL databases?
2. What is the CAP theorem?
3. What are ideal use cases of HBase?

Task 2 - Bloom filters

Bloom filters are used in hbase as an incredible optimization. Solve the following.

1. What is a bloom filter?
2. What is an advantage of bloom filters over hash tables?
3. What is a disadvantage of bloom filters?
4. Using your language of choice, implement a bloom filter with `add` and `check` functions. The backing bit-array can simply be a long (64 bit integer).
5. If you are to store one million ASCII strings with an average size of 10 characters in a hash set, what would be the approximate space consumption?
6. The following equation gives the required number of bits of space per inserted key, where $\mathcal{E}$ is the false positive rate.

$$b = 1.44 log_2(1/\mathcal{E}) \qquad (1)$$

7. How many bits per element are required for a 1% false positive rate?

8. How many bits per element are required for a 5% false positive rate?
9. If you are to store one million ASCII strings with an average size of 10 characters in a bloom filter, what would be the approximate space consumption, given an allowed false positive rate of 5%?.

Task 3 - Huffman coding

HBase internally uses a compression that is a combination of LZ77 and Huffman Coding.

1. Generate Huffmann Code (and draw the Huffmann Tree) based on the following string: "beebs beepps!!!!! their eerie ears hear pears"
2. How many bits is the compressed string? How many bits is the raw ASCII string?
3. Compress "pete is here" with the Huffmann tree from before.
4. Write your own 10 word sentence. Generate the Huffmann Code (a new Huffmann Tree), and write a new compressed message (ie. in binary). Swap with one of your fellow students, and decompress each other's message.

Task 4 - Map and Reduce

Solve the following using Javascript, for example in your browser's developer console.

1. Map the list of numbers to a list of their square roots: [1, 9, 16, 100]
2. Map the list of words so each is wrapped in a ¡h1¿ tag: ["Intro", "Requirements", "Analysis", "Implementation", "Conclusion", "Discussion", "References"]
3. Use map to uppercase the words (all letters): ["i'm", "yelling", "today"]
4. Use map to transform words into their lengths: ["I", "have", "looooooong", "words"]
5. Get the json file `comics.json` from the course site. Paste it into your browser's Javascript console. Use map to get all the image urls, and wrap them in img-tags.
6. Use reduce to sum the array of numbers: [1,2,3,4,5]
7. Use reduce to sum the x-value of the objects in the array: [{x: 1},{x: 2},{x: 3}]

8. Use reduce to flatten an array of arrays: [[1,2],[3,4],[5,6]]

9. Use reduce to return an array of the positive numbers: [-3, -1, 2, 4, 5]

10. Optional: The accumulator function can obviously use objects outside of itself. Use reduce to implement groupBy. For example:

    ```
    people = [
      {name: 'Rikke', age: 46},
      {name: 'Michael', age: 47},
      {name: 'Mathias', age: 46}
    ];
    ```

    should be turned into

    ```
    groupedPeople = groupBy(people, 'age');
    /*
    groupPeople:
    {
      46: [
        {name: 'Rikke', age:46},
        {name: 'Mathias', age:46}
      ],
      47: [
        {name: 'Michael', age:47}
      ]
    }
    */
    ```