
Exercise 1

Download the zip archive from the website and try to make yourself familiar with the classes in `./physAlgEx6/PhysAlgebra.hh`. The classes implement the operators of a physical algebra with the following characteristics:

- push-based (memory flow and control flow)
- tuple-at-a-time processing
- processes tuples in row format
- assumes data is stored in row format

The files `Relation.hh` and `Relation.cc` implement the storage layer of a simple row store. You don't have to look at the details of this code unless you're interested in it. It allows you to store data together with a schema in the `database` sub directory that will be loaded to main memory when you create a `Relation` object.

Exercise 1 a)

Implement the `init()`, `step()` and `fin()` functions of the `Selection` operator of the physical algebra. If you get stuck, you may want to have a look at the solution code.

Exercise 1 b)

In the sub directory `database` you can store relations that you can query using the physical algebra. To store a new relation `R`, provide a file `R.data` containing the data and a file `R.schema` containing the schema. See `database/test.data` and `database/test.schema` for an example. Also make yourself familiar with the `crimeInAtlanta2017` data set given in the same directory. For details on the data set, see

<https://www.kaggle.com/priscillapun/crime-in-atlanta-2017>

Note that attributes that are nullable are of type string and that the empty string represents a null value.

You can implement queries by manually building their query plan using the operators of the physical algebra. Then, run the query by invoking the `run` method of the `tablescan`.

Implement the following queries:

- (i) `SELECT name`
`FROM test`
`WHERE name = 'Olivia';`
- (ii) `SELECT *`
`FROM crimeInAtlanta2017`
`WHERE neighborhood = 'Peachtree_Hills';`
- (iii) `SELECT location , MaxOfnum_victims`
`FROM crimeInAtlanta2017`
`WHERE x BETWEEN -84.36 AND -84.35`
`AND y BETWEEN 33.73 AND 33.74;`

You can just put your solution into the main function in `main.cc`. You can also find an example query there.

Exercise 2

You are given the following database relation:

PERSONS		
id	name	haircolor
0	Tom	brown
1	Olivia	blond
2	Esteban	black
3	Gaspar	black
4	Davy	brown
5	Jade	red
6	Daniel	brown
7	Clemens	blond

Exercise 2 a)

Assume that the value of the attribute `haircolor` is either black, brown, blond or red. The attribute is not nullable. How many bits do you need to encode these hair colors? Give a concrete encoding, i. e., a mapping $enc : \{\text{black, brown, blond, red}\} \rightarrow \{0, 1\}^k$, where k denotes the number of bits needed for the encoding.

Exercise 2 b)

Execute the following query by hand:

```
SELECT name
FROM Persons
WHERE haircolor <> 'blond';
```

Use the *BitSliceH* method presented in the lecture (Script, pp. 61 ff.) to retain only the hair colors that fulfill the selection predicate. Use a register size of $w = 16$. Show all intermediate steps. Indicate which tuples qualify.

Recall the three steps of BitSliceH:

1. Calculate the result bit vector Z using the formula applicable to the selection predicate.
2. Extract the indicator bits.
3. Convert the bit vector of indicator bits to indices (row identifiers).

For this exercise, it is sufficient to only perform step 1.

Hints

- Given a register size (word size) of $w = 16$ bit and k bit needed for the encoding the hair colors, how many encoded values fit into one 16-bit word?
- Use a padding of zeros if the rightmost value does not completely fit into the current word.
- Write down intermediate results in order not to get confused with more complex bit manipulation expressions.