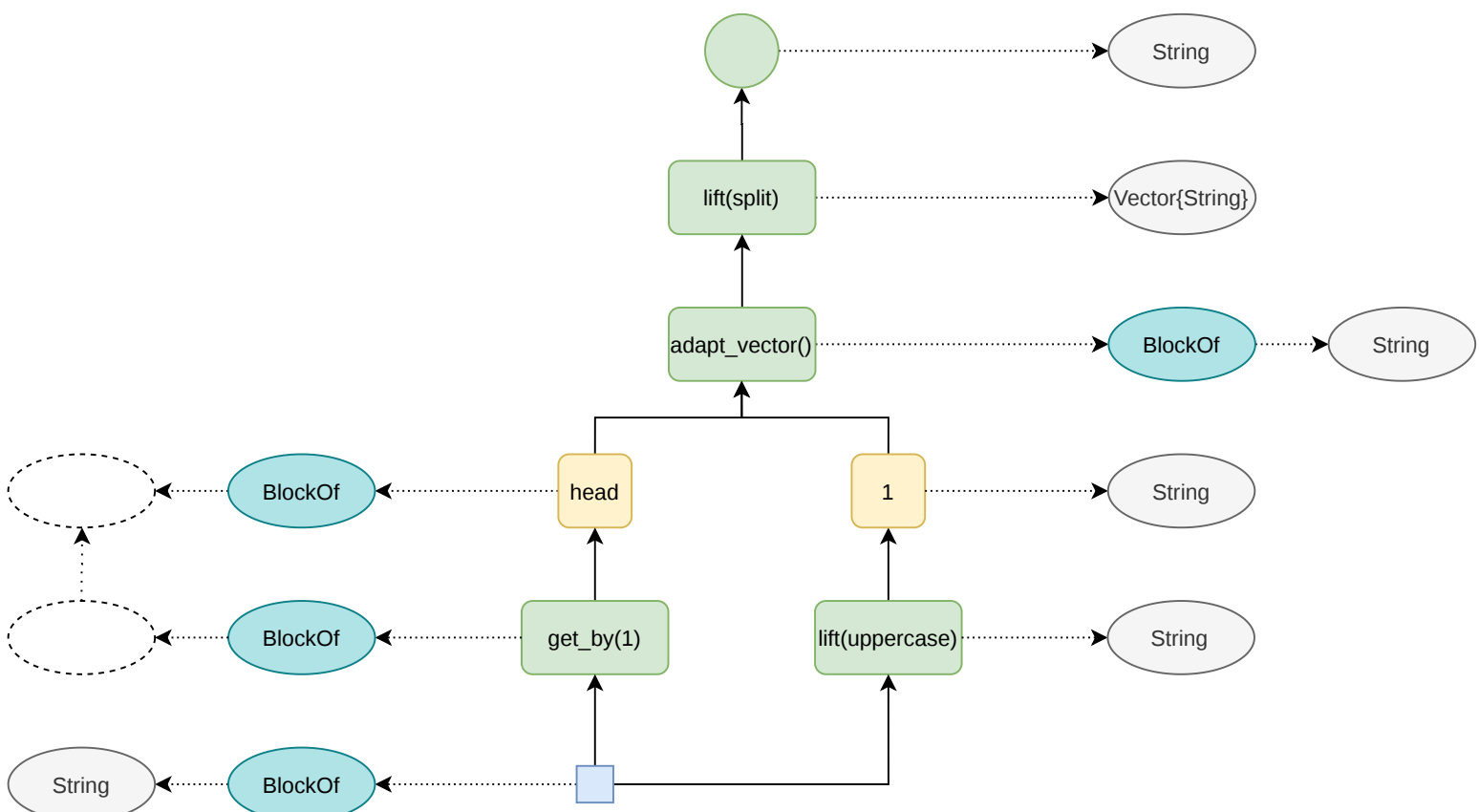
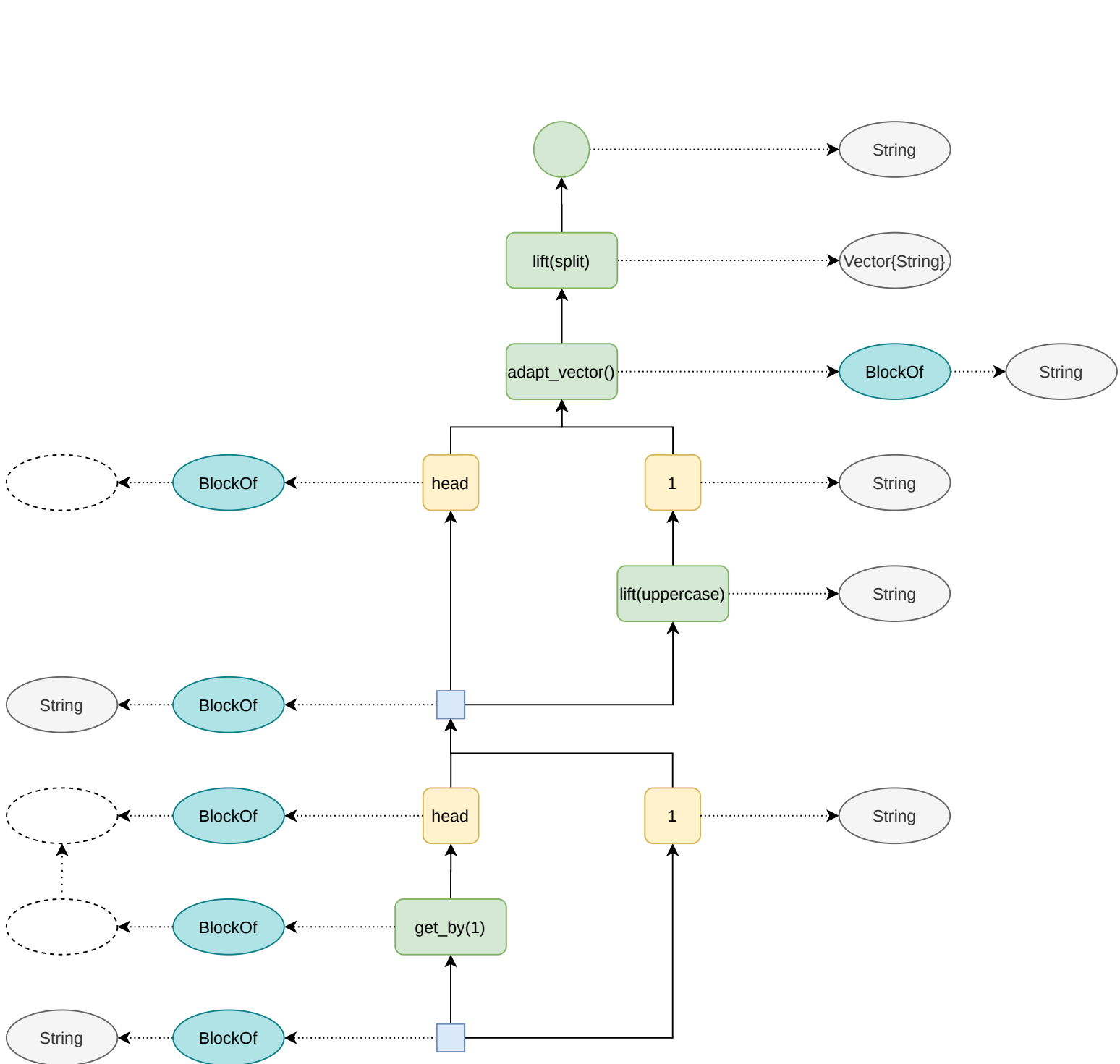
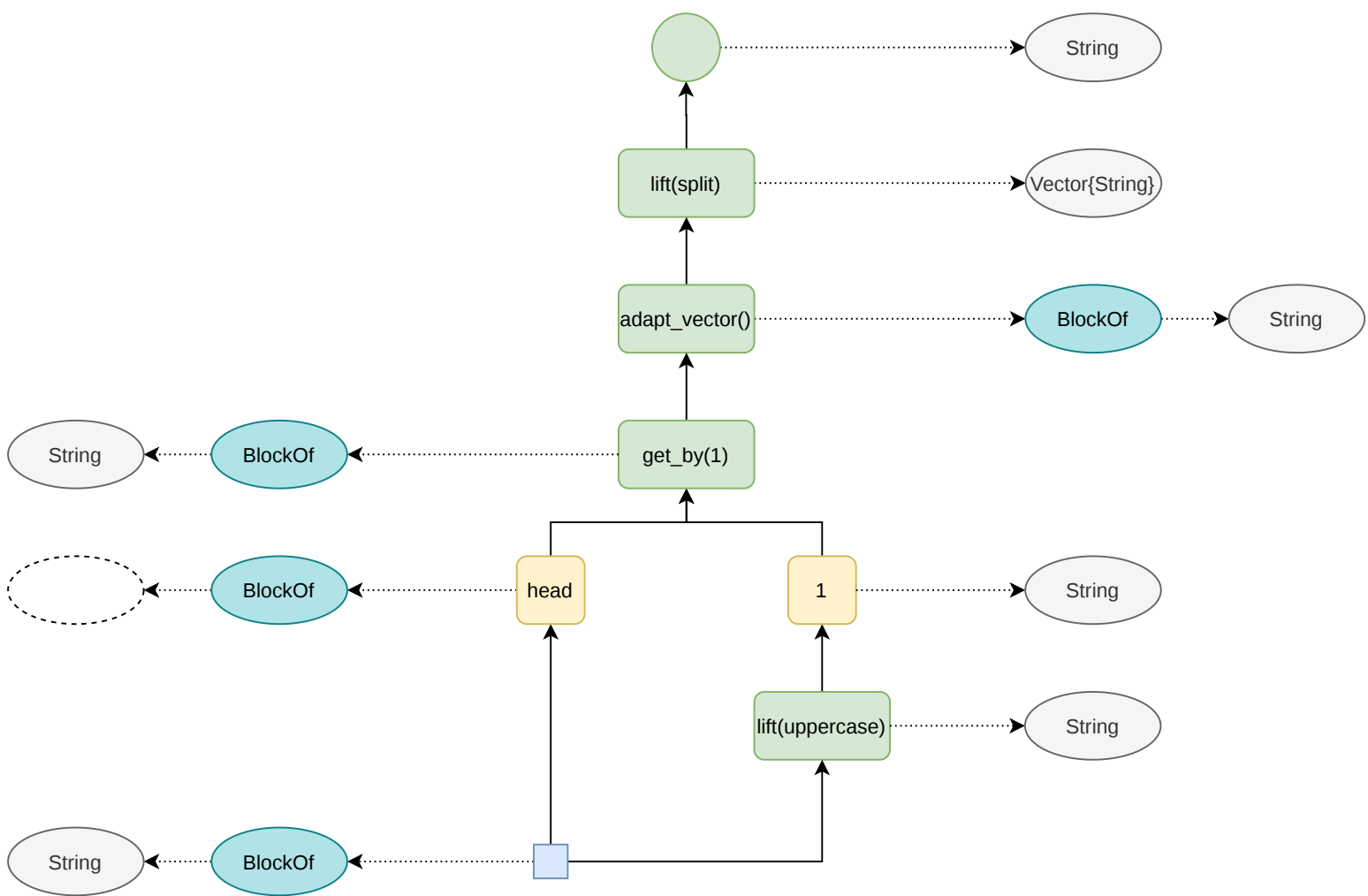
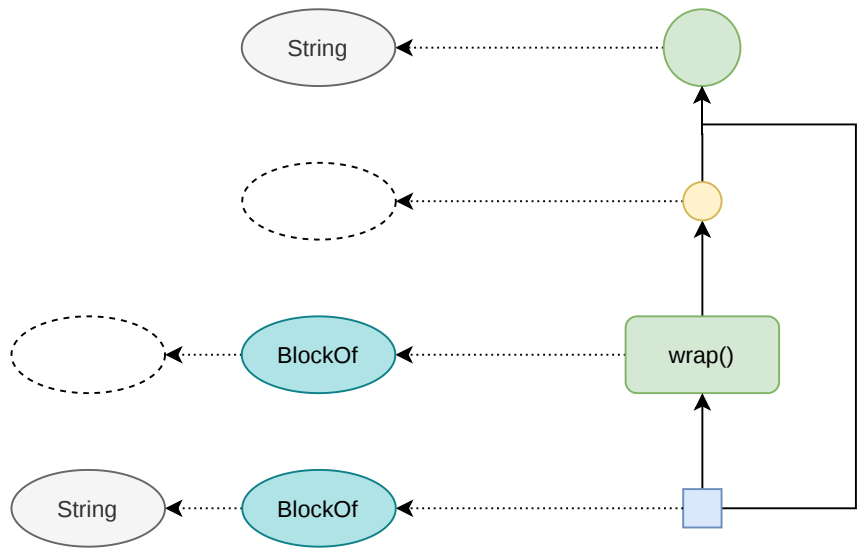
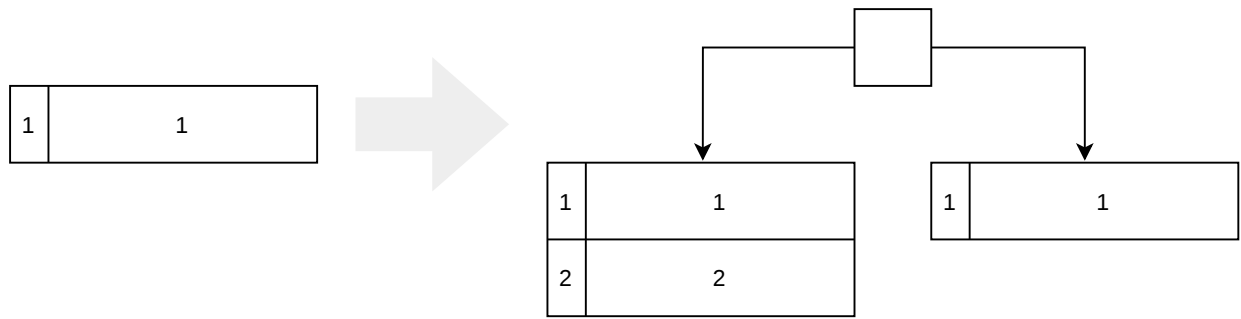
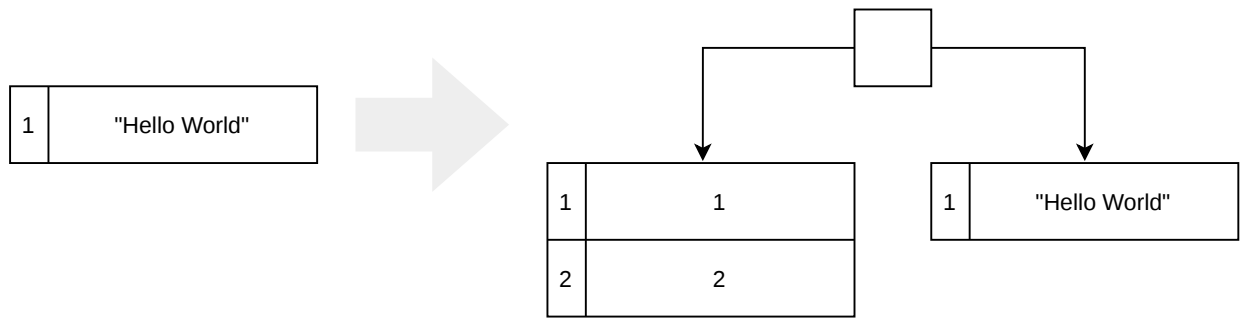


The diagram illustrates the transformation of a flat table into a hierarchical tree structure through a series of steps:

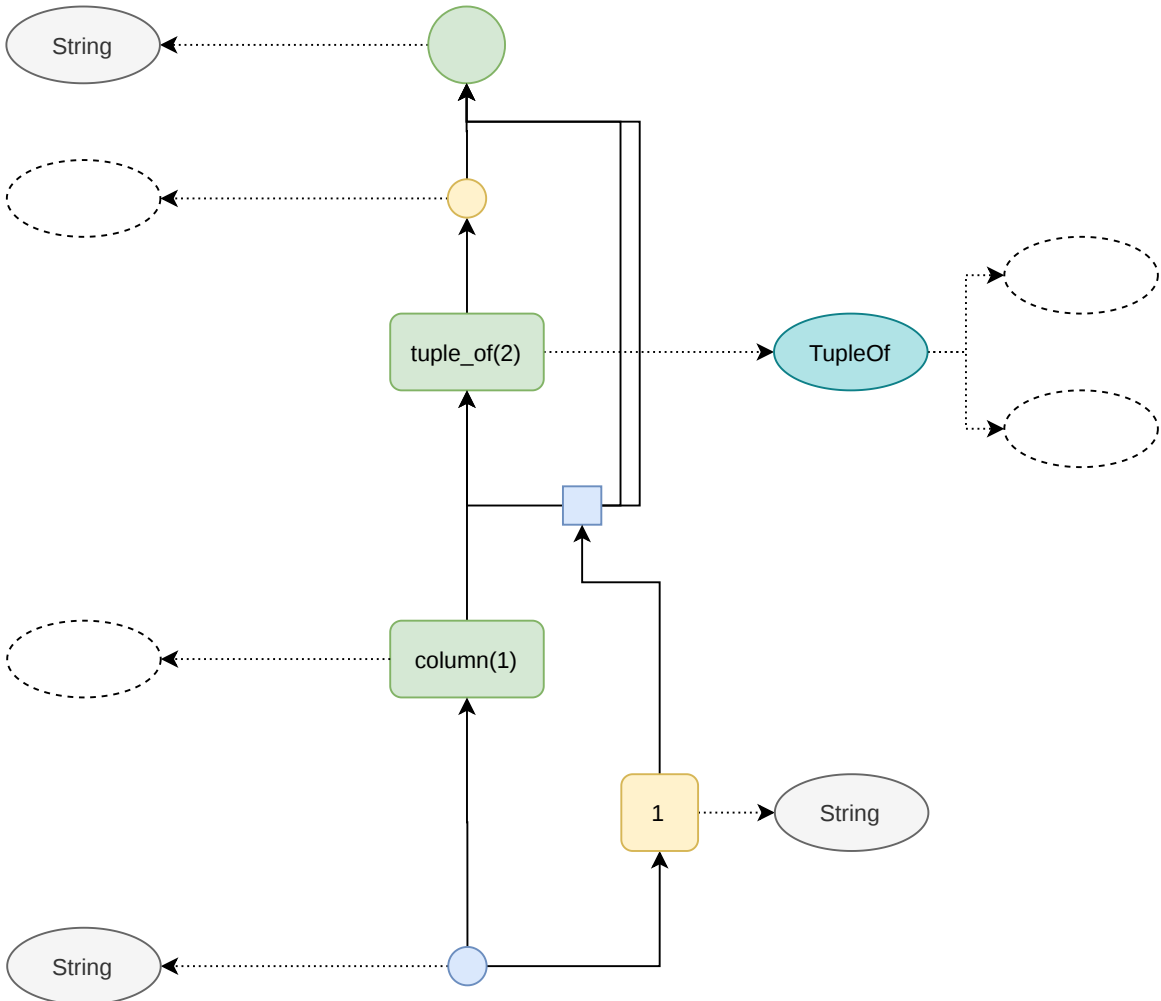
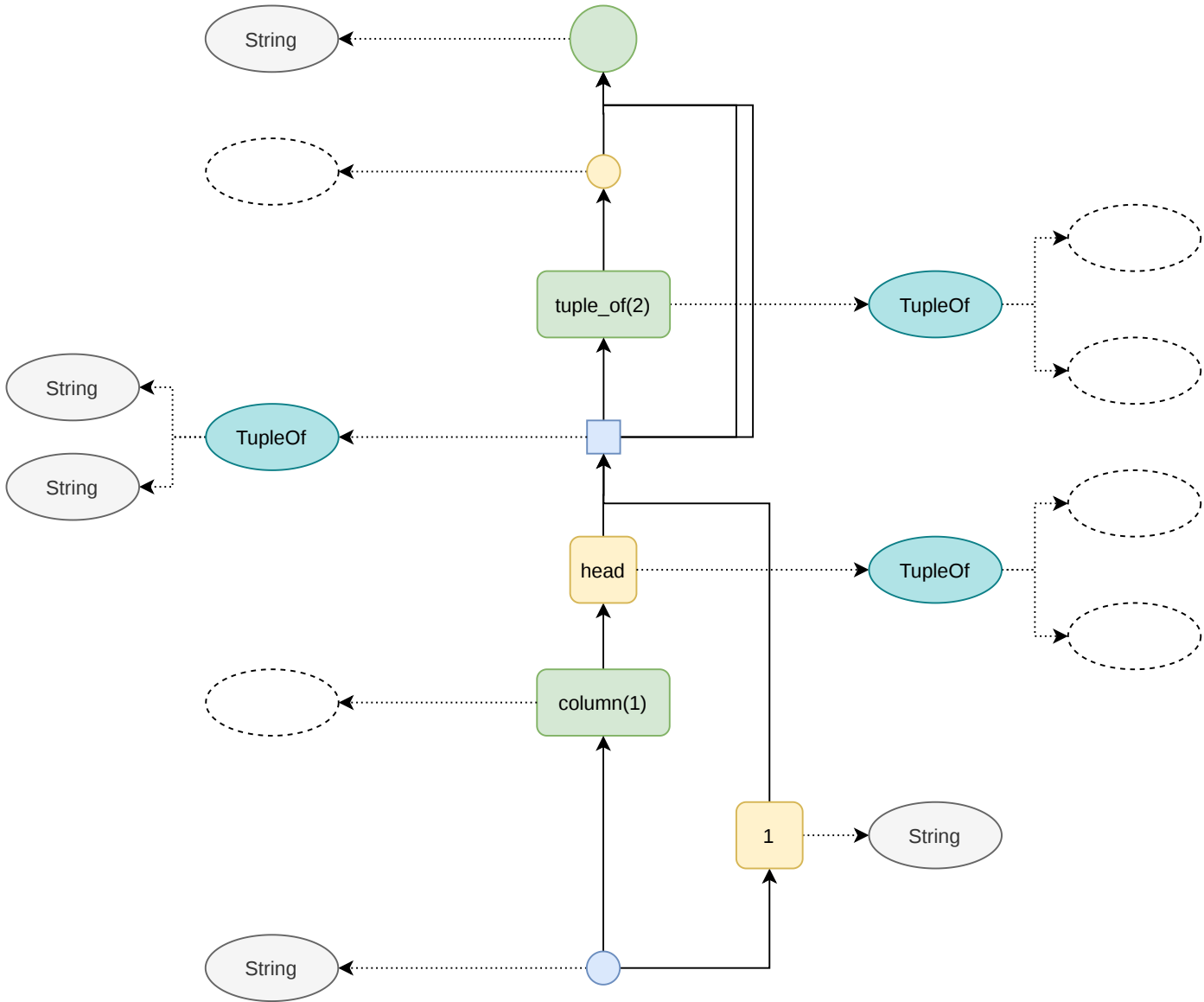
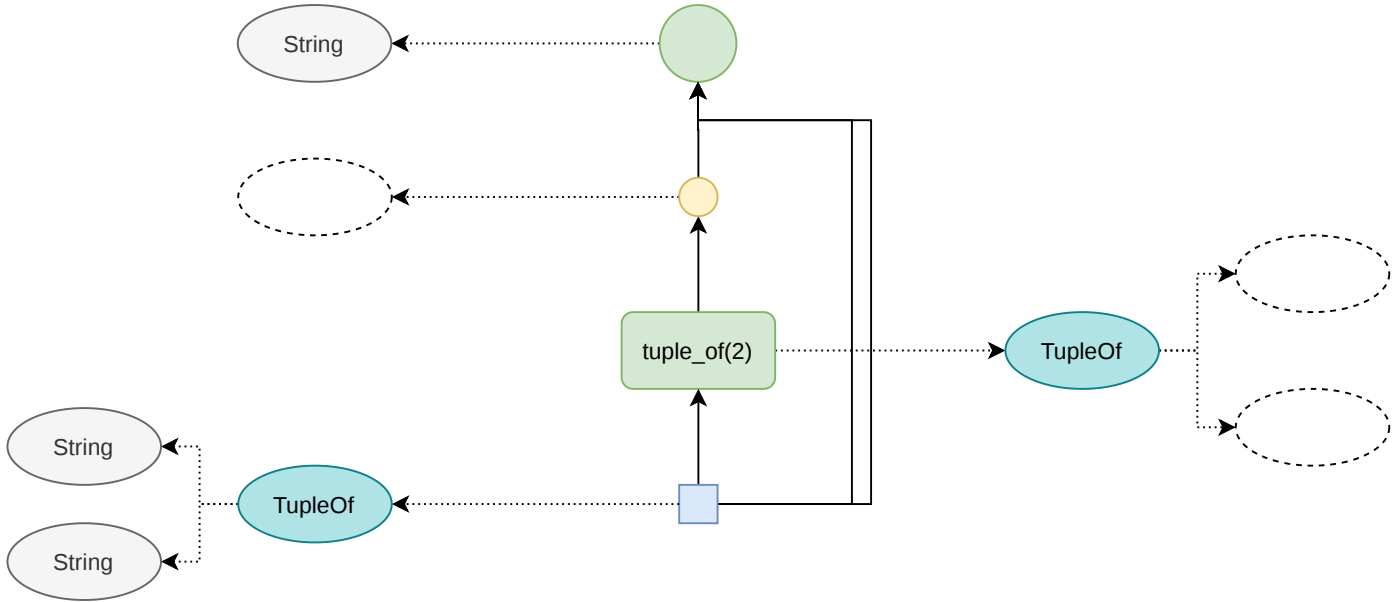
- Initial State:** A flat table with two columns: an index (1) and a value ("Hello World").
- Transformation:** The value "Hello World" is split into an array: `String["Hello", "World"]`.
- Indexing:** The array is processed into a tree structure. The root node branches into two child nodes:
  - Node 1: Index 1, Value 1
  - Node 2: Index 1, Value "Hello"; Index 2, Value "World"
- Normalization:** The tree structure is normalized. The root node branches into two child nodes:
  - Node 1: Index 1, Value 1; Index 2, Value 3
  - Node 2: Index 1, Value "HELLO"; Index 2, Value "WORLD"
- Final State:** The tree structure is further processed. The root node branches into two child nodes:
  - Node 1: Index 1, Value 1; Index 2, Value 2
  - Node 2: Index 1, Value "HELLO"



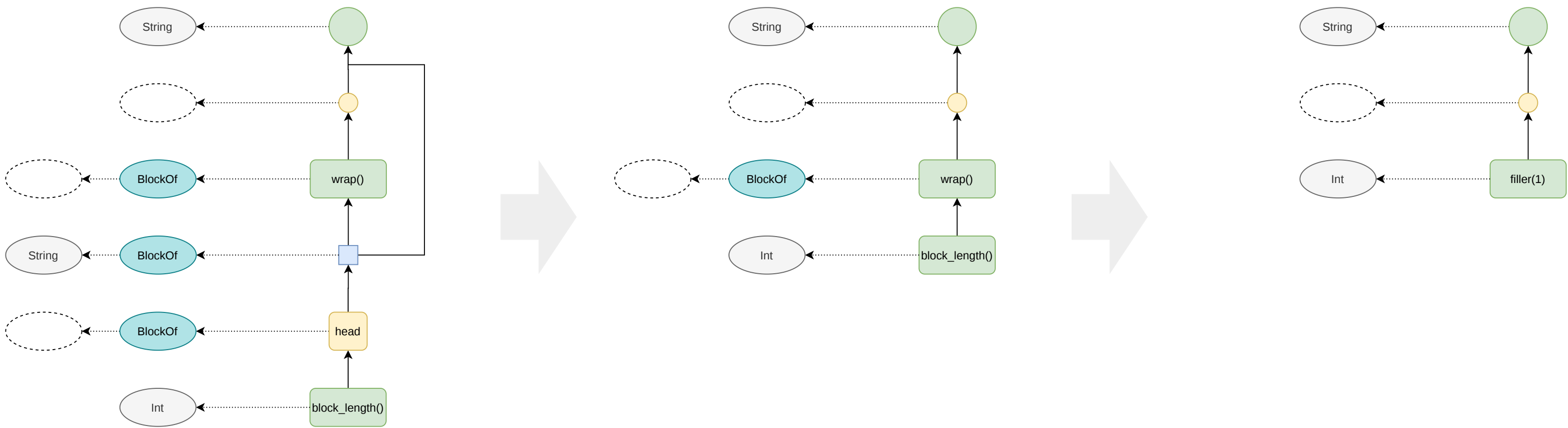
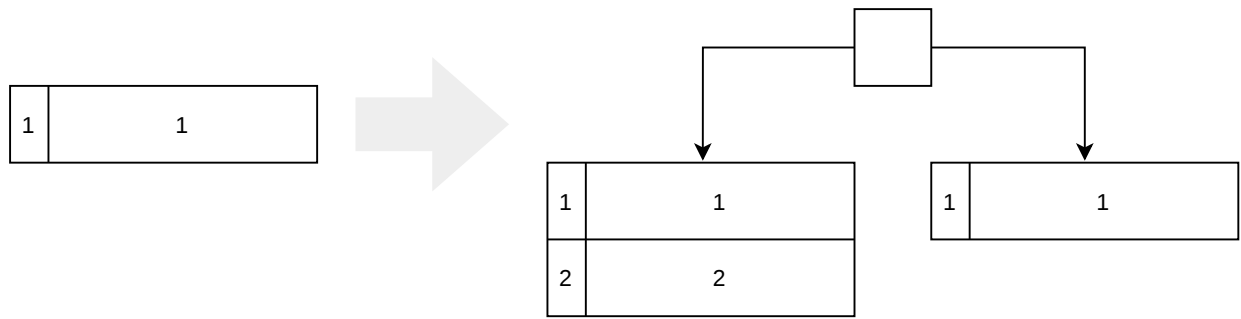
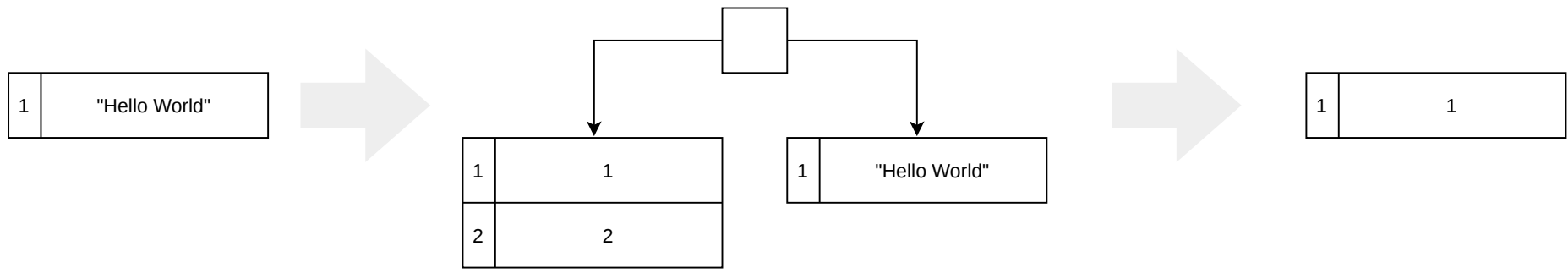
wrap()



chain\_of(tuple\_of(2), column(1))



chain\_of(wrap(), block\_length())



@query "Hello World" split(it)

```
chain_of(
  wrap(),
  with_elements(
    chain_of(
      wrap(),
      lift(split),
      adapt_vector()),
    flatten())
```

