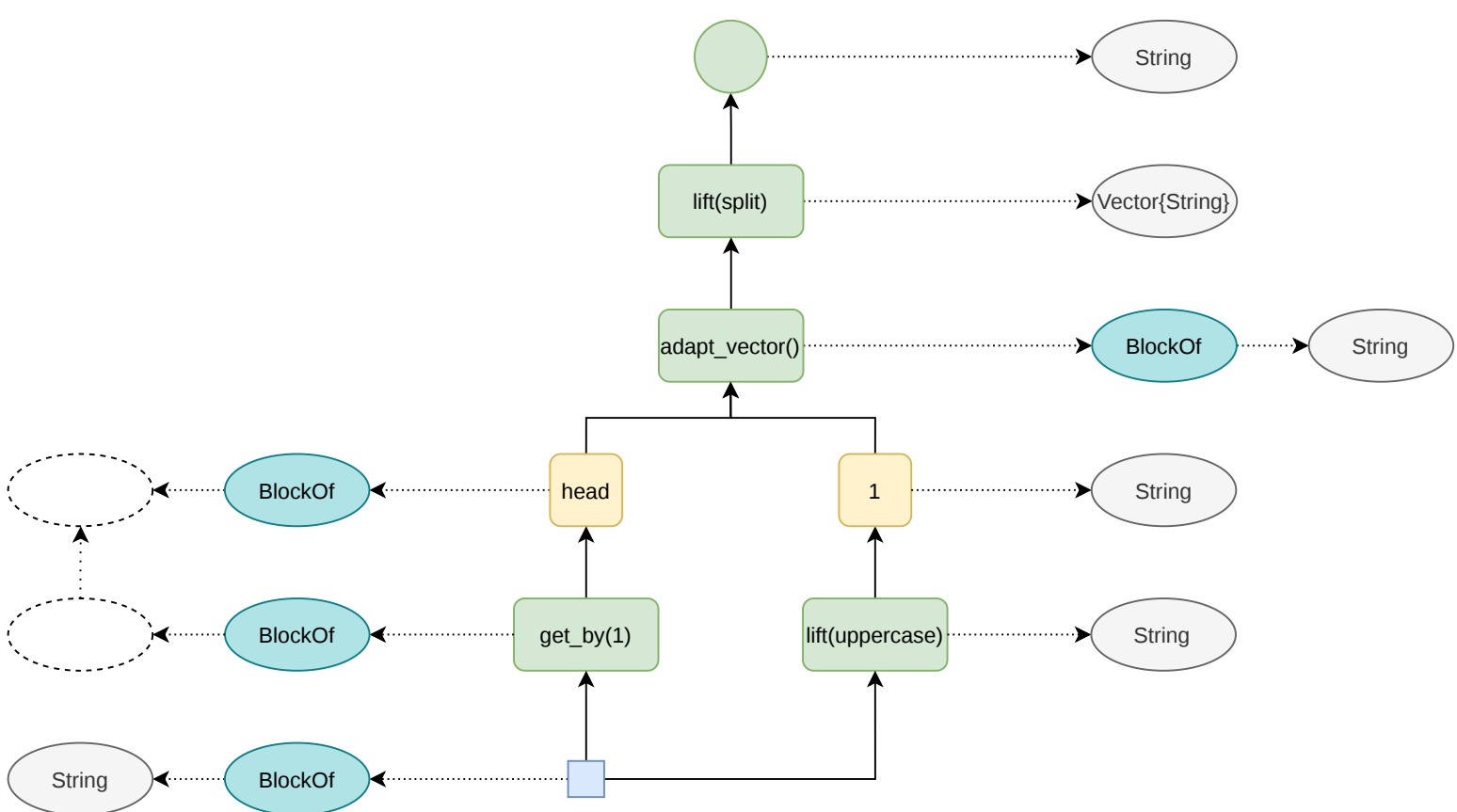


The diagram illustrates the transformation of a flat table into a hierarchical tree structure through a series of steps:

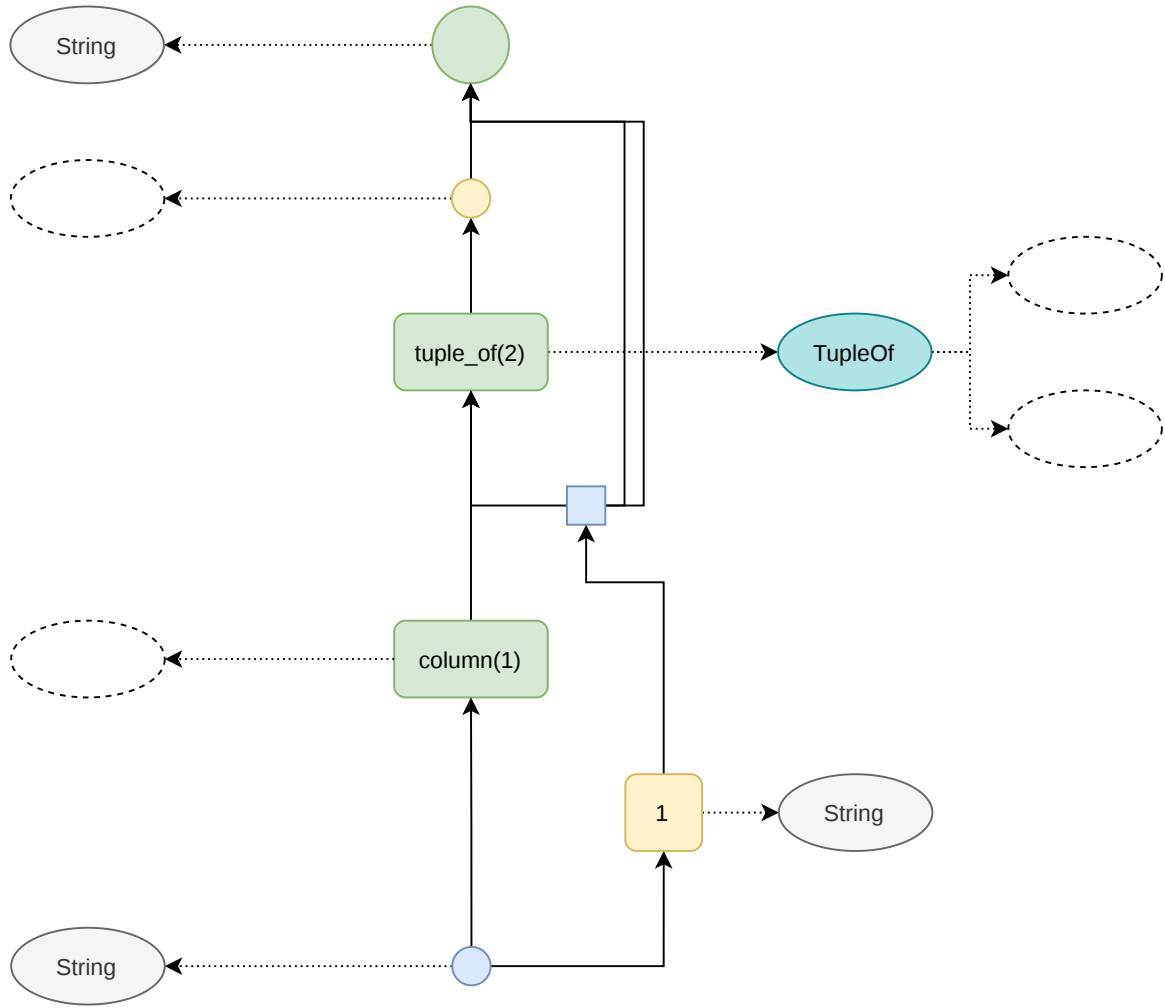
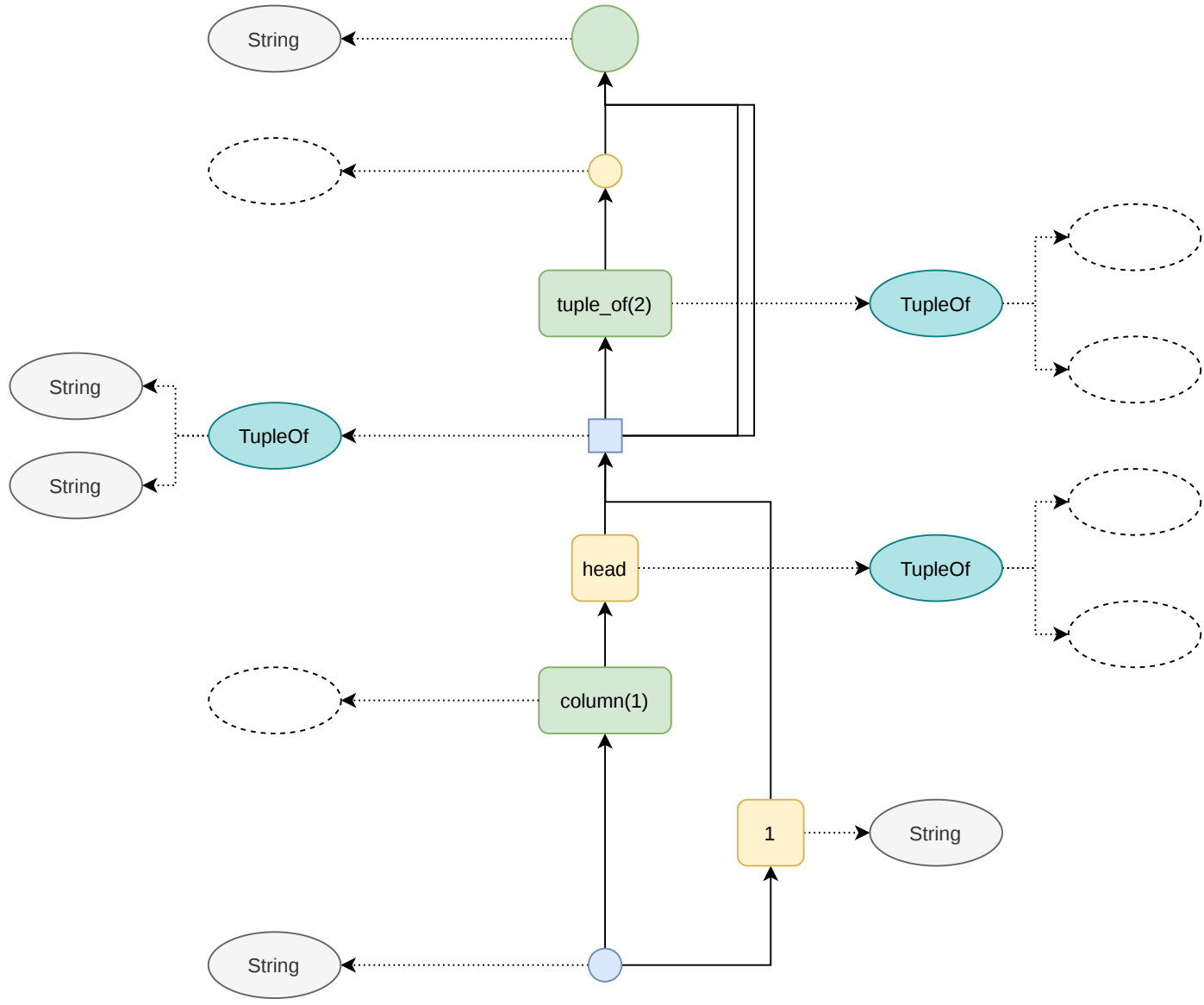
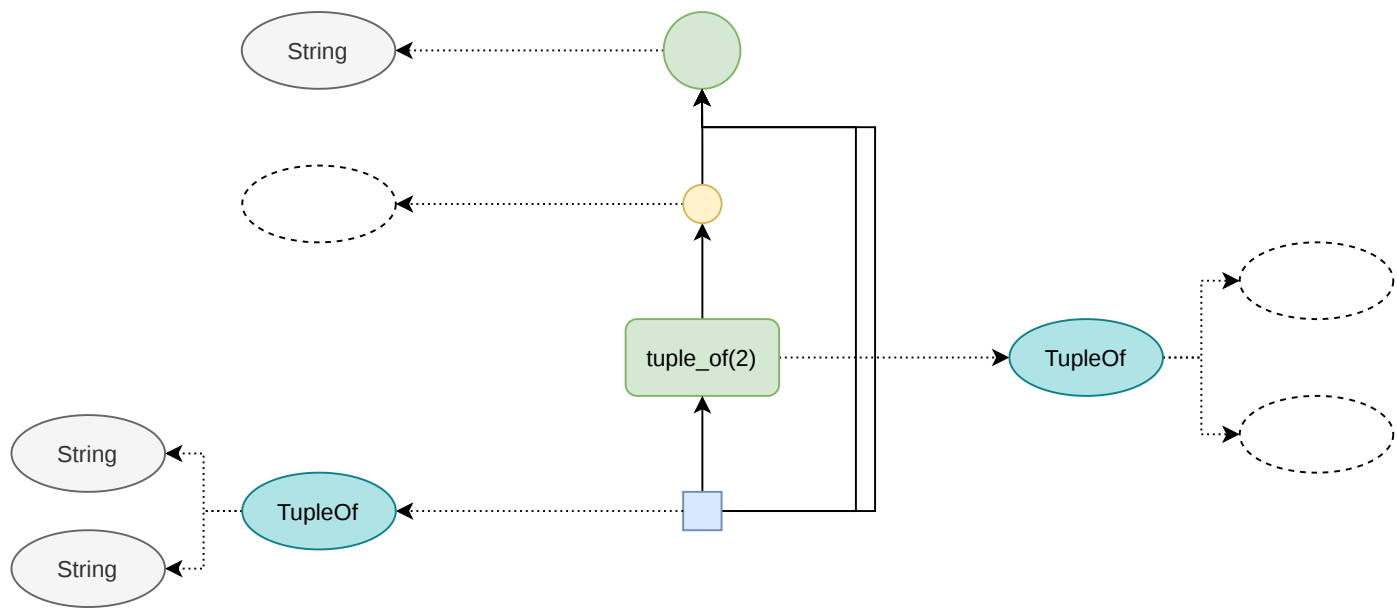
- Initial Flat Table:** A single table with two columns: an index (1) and a value ("Hello World").
- String Splitting:** The value "Hello World" is split into an array of strings: ["Hello", "World"].
- Indexing and Splitting:** The array is indexed (1: "Hello", 2: "World"). A new root node is created, branching into two child nodes corresponding to the indexed elements.
- Case Folding:** The string values are converted to uppercase: "HELLO" and "WORLD".
- Final Hierarchical Structure:** The structure is further processed, resulting in a tree where the root branches into two nodes, each of which branches into two more nodes, representing a more complex hierarchy.



wrap()



chain_of(tuple_of(2), column(1))



sieve_by()

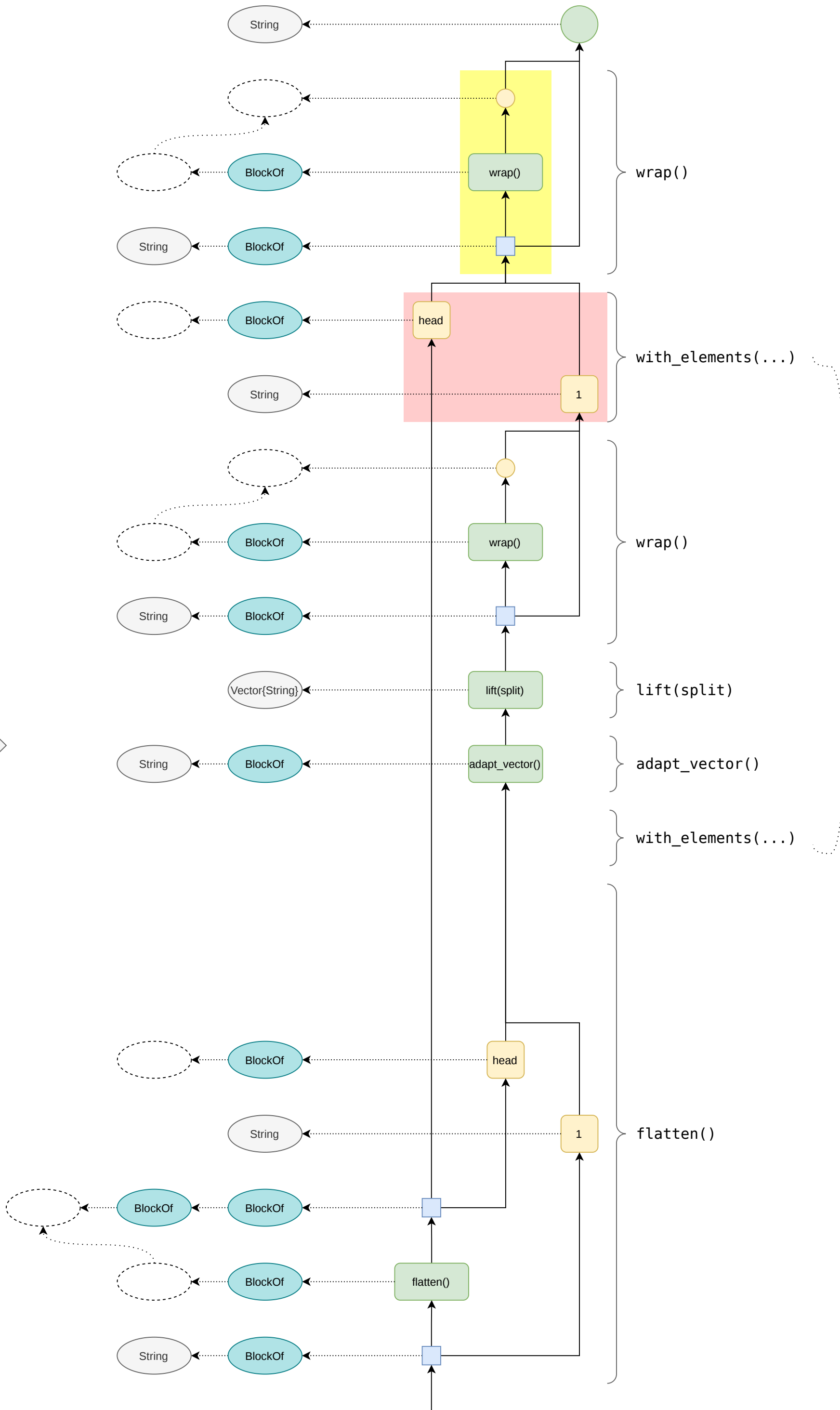


chain_of(wrap(), block_length())



@query "Hello World" split(it)

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



untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline,Vector{NodeRef}}



@query "Hello World" split(it)

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







untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline,Vector{NodeRef}}

$$\{it^2, (it^2)^3\}$$


```
untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline, Vector{NodeRef}}
```

@query "Hello World" split(it)

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

