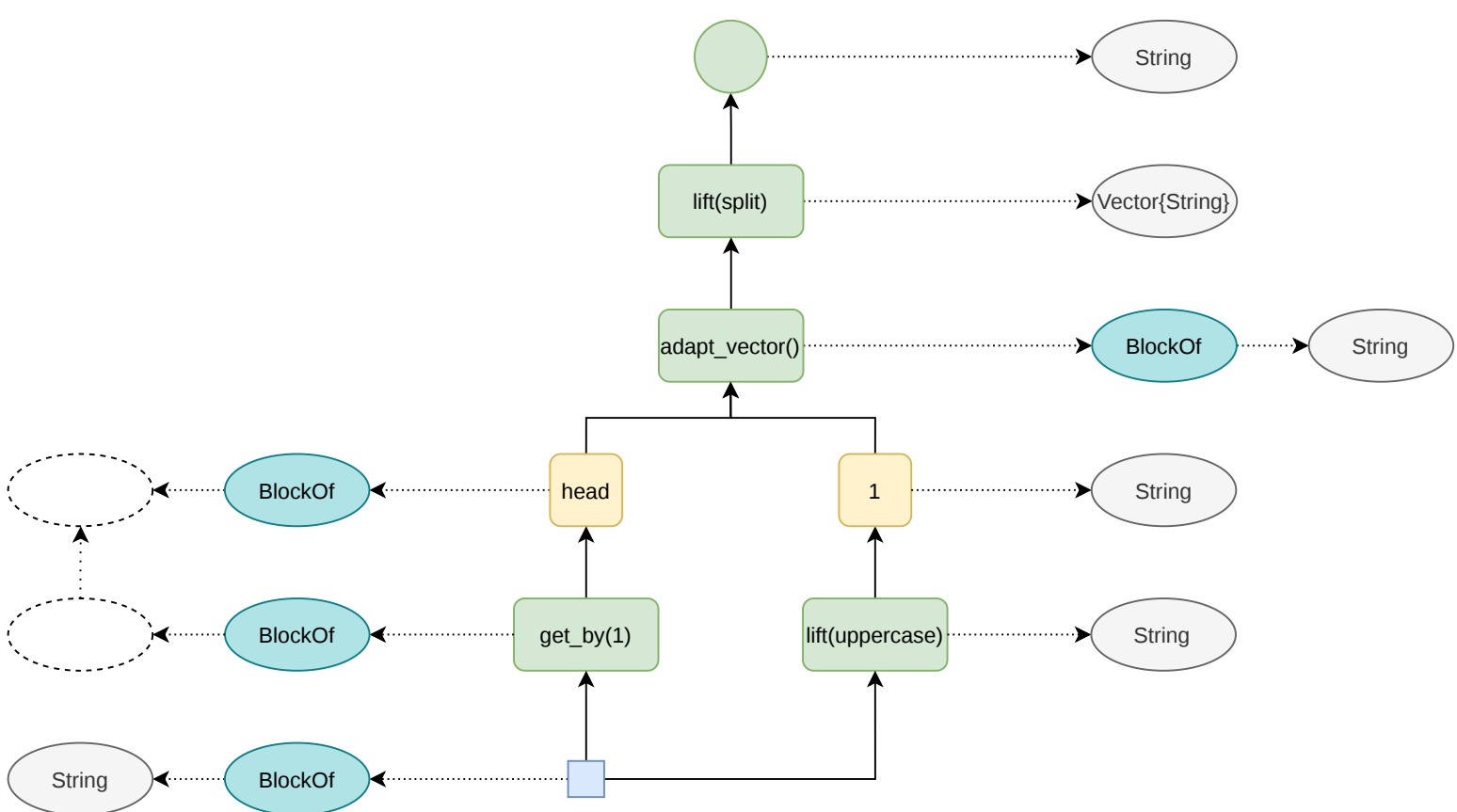




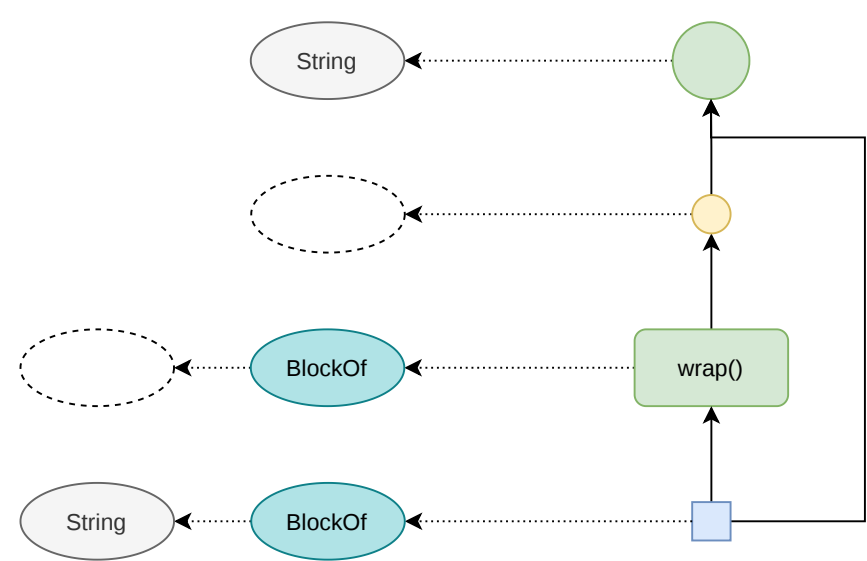
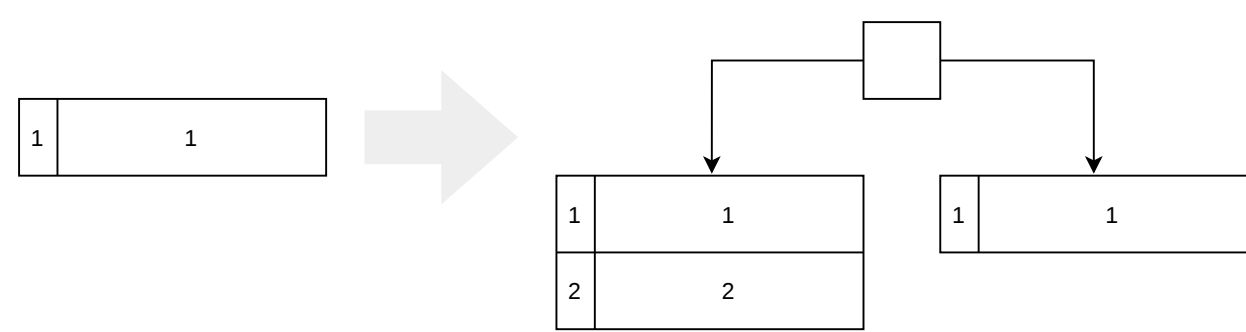
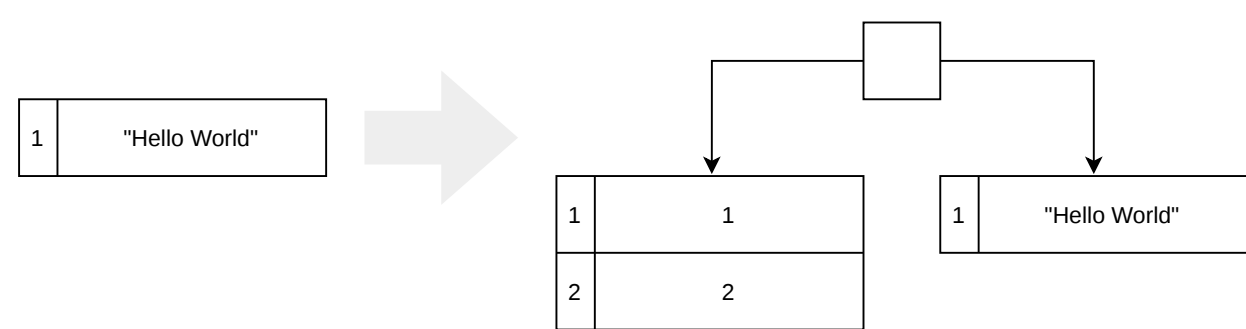
The diagram illustrates the transformation of a flat table into a hierarchical tree structure through four stages, connected by large grey arrows.

- Stage 1:** A flat table with two columns: an index column (1) and a value column ("Hello World").
- Stage 2:** The table is transformed into a hierarchical structure. The root node is a table with index 1 and value "String['Hello', 'World']". It branches into two child nodes:
 - Left child: index 1, value 1.
 - Right child: index 1, value "Hello"; index 2, value "World".
- Stage 3:** The hierarchical structure is further transformed. The root node branches into two child nodes:
 - Left child: index 1, value 1; index 2, value 3.
 - Right child: index 1, value "HELLO"; index 2, value "WORLD".
- Stage 4:** The final stage shows a more complex hierarchical structure. The root node branches into two child nodes:
 - Left child: index 1, value 1; index 2, value 2.
 - Right child: index 1, value "HELLO".

The diagram demonstrates how a flat table can be transformed into a hierarchical tree structure, which is a common requirement in data processing and analysis.



wrap()



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



sieve_by()

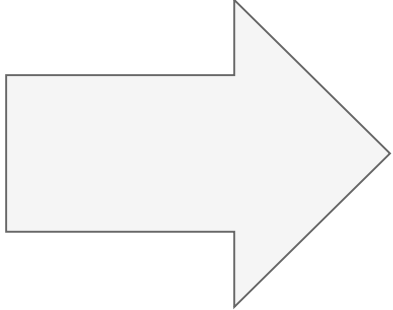


chain_of(wrap(), block_length())



@query "Hello World" split(it)

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



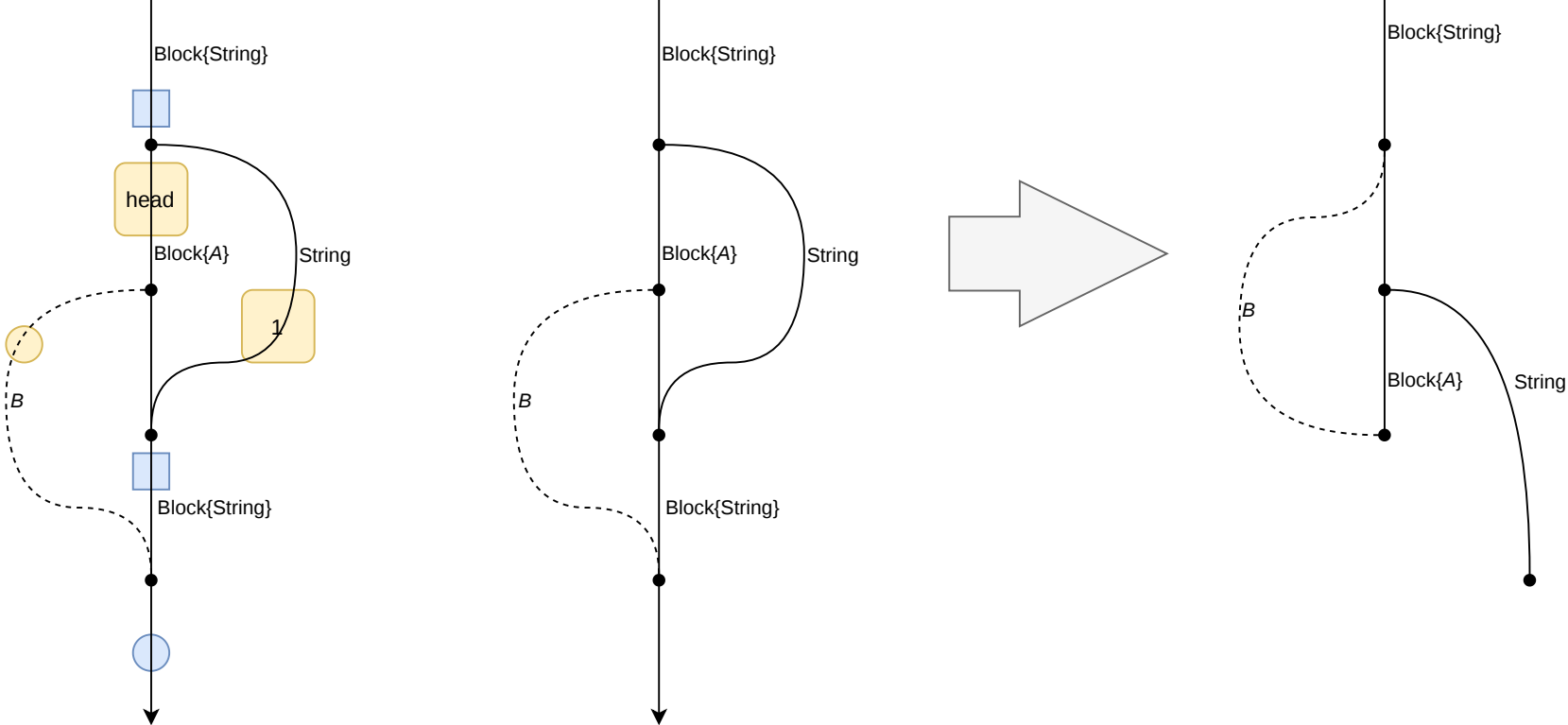
untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline,Vector{NodeRef}}





@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}` `chain_of(f, tuple_of(g, h)) => tuple_of(chain_of(f, g), chain_of(f, h))`



`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())
```

