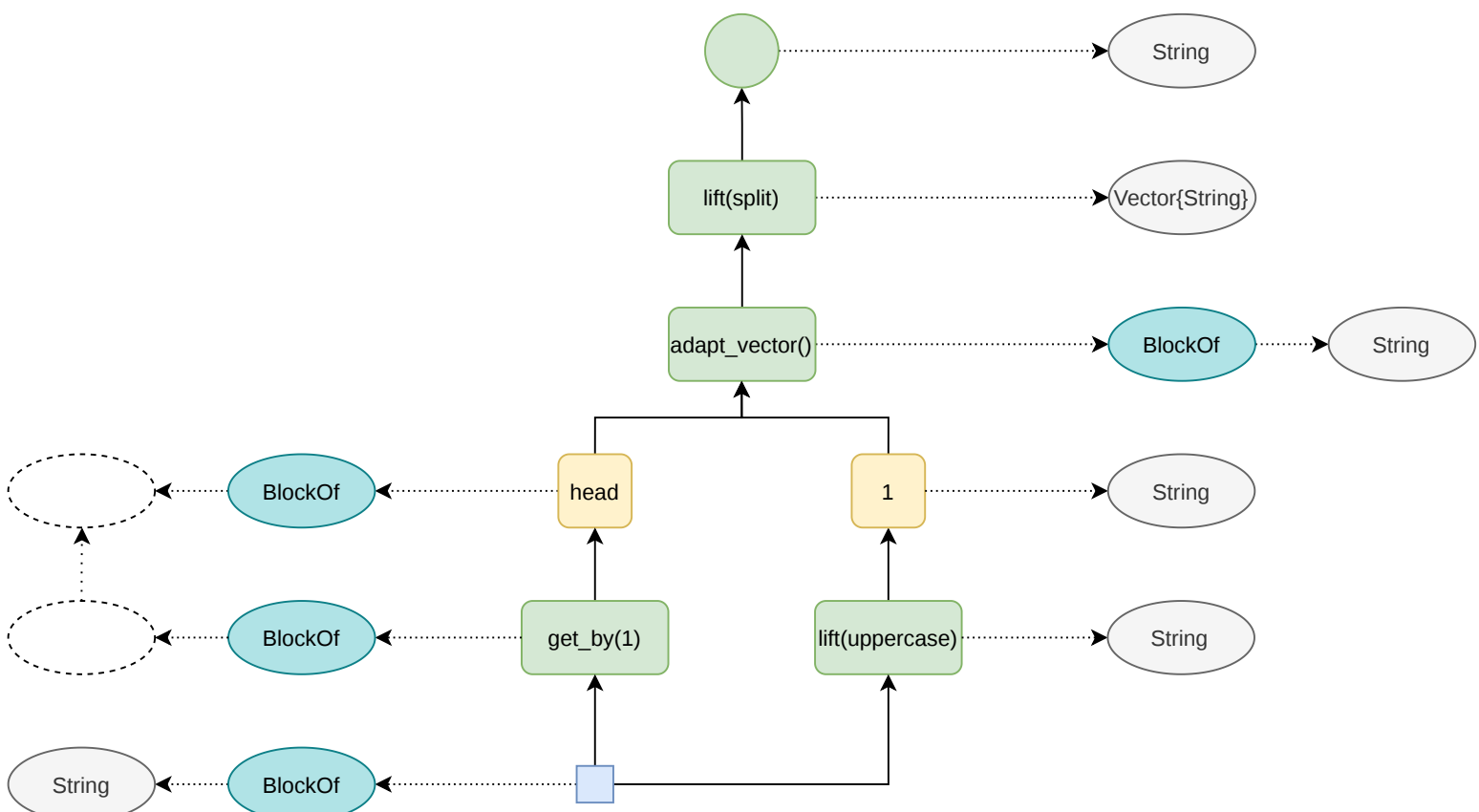


The diagram illustrates the transformation of a flat table into a hierarchical tree structure through four stages, connected by large gray arrows:

- Stage 1:** A flat table with two columns: an index column (1) and a value column ("Hello World").
- Stage 2:** A flat table with two columns: an index column (1) and a value column (String["Hello", "World"]).
- Stage 3:** A hierarchical tree structure. The root node is a box with two children. The left child is a table with two rows: (1, 1) and (2, 3). The right child is a table with two rows: (1, "Hello") and (2, "World").
- Stage 4:** A hierarchical tree structure. The root node is a box with two children. The left child is a table with two rows: (1, 1) and (2, 2). The right child is a table with one row: (1, "HELLO").

Below the Stage 3 tree, a large gray arrow points down to a single table with two rows: (1, "HELLO") and (2, "WORLD").

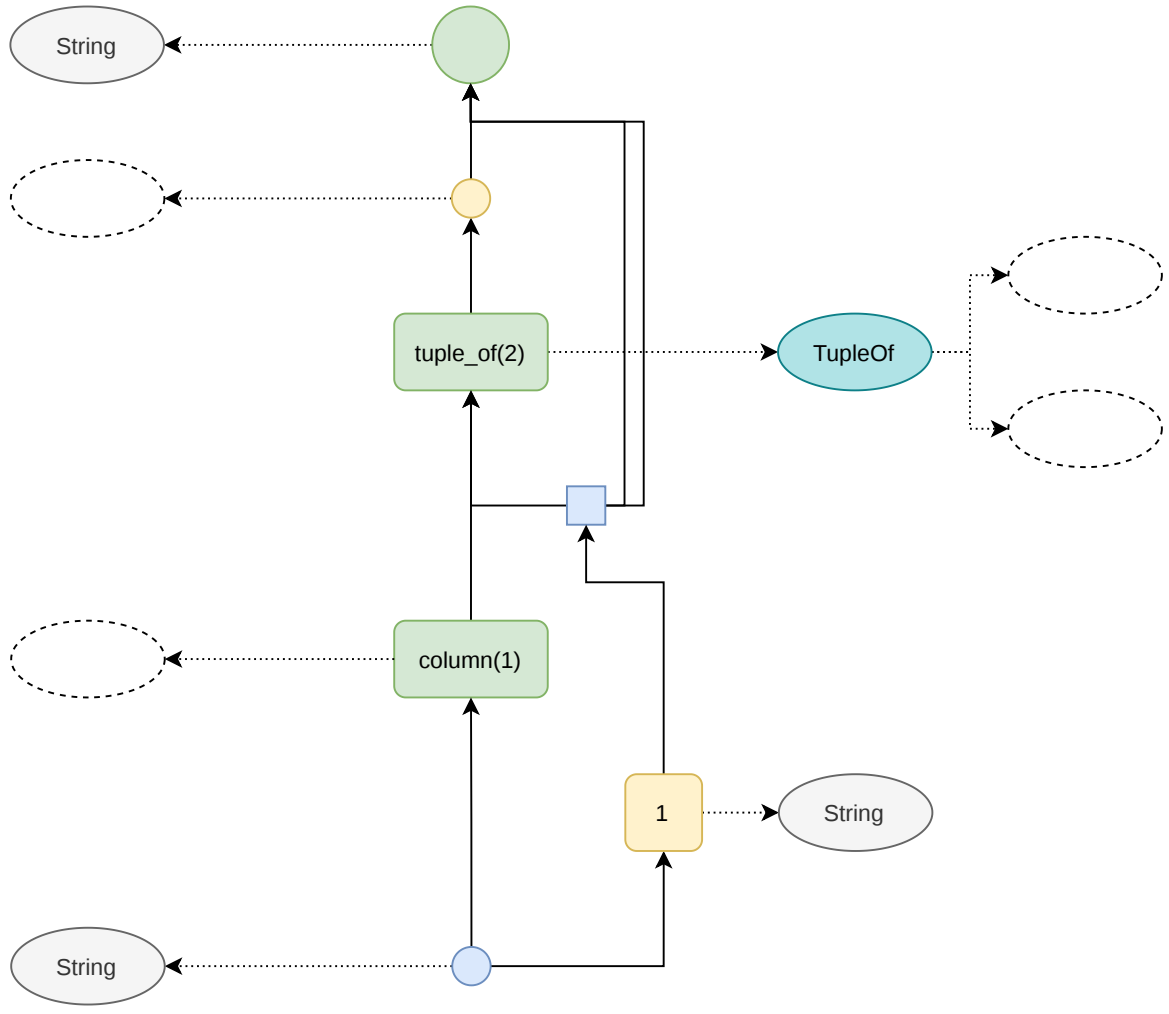
Below the Stage 4 tree, a large gray arrow points down to a single table with two rows: (1, 1) and (2, 2).



wrap()



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



sieve\_by0



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)

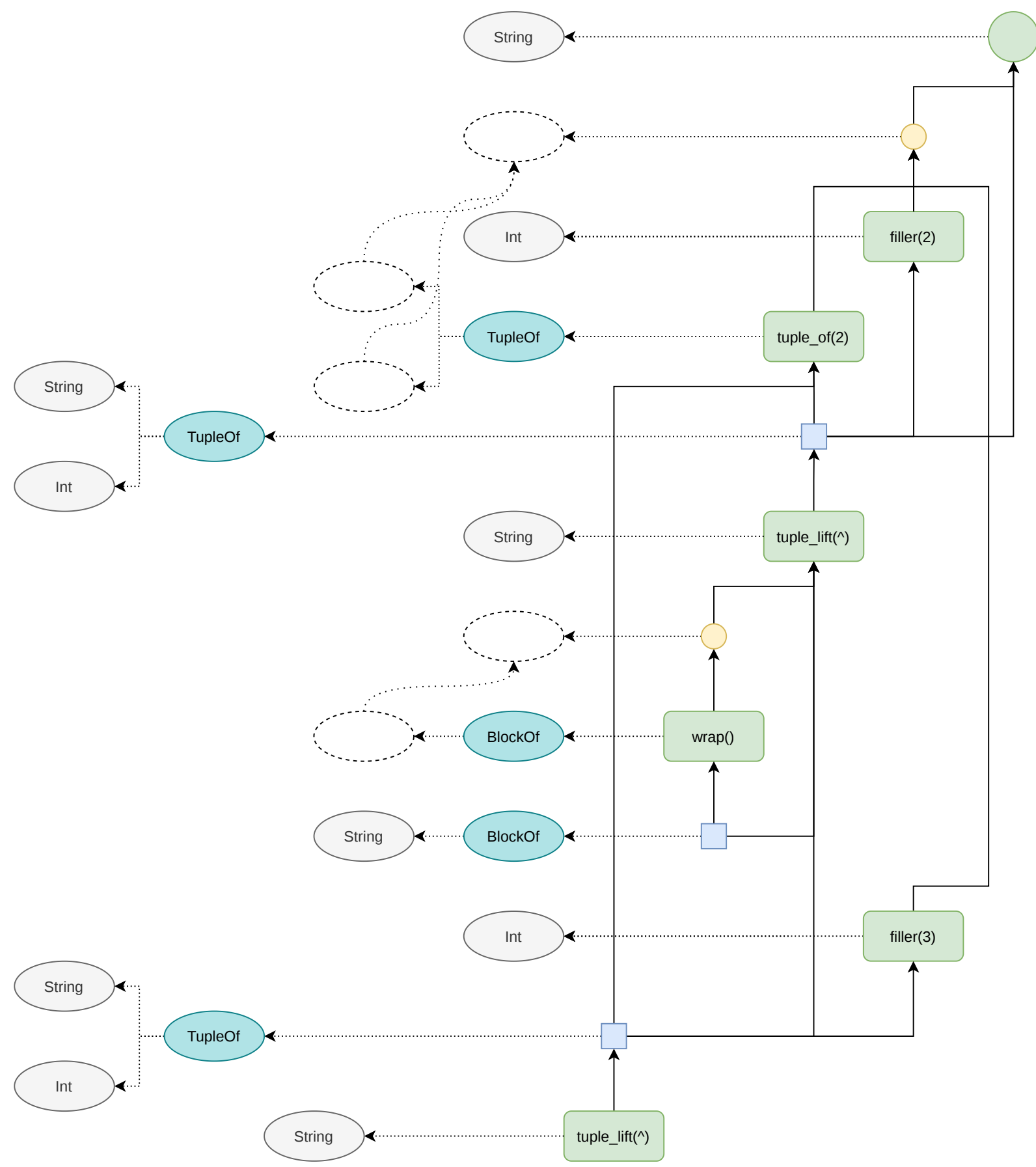
```
chain_of(
  wrap(),
  with_elements(
    chain_of(
      wrap(),
      lift(split),
      adapt_vector()),
    flatten())
```







`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())
```

