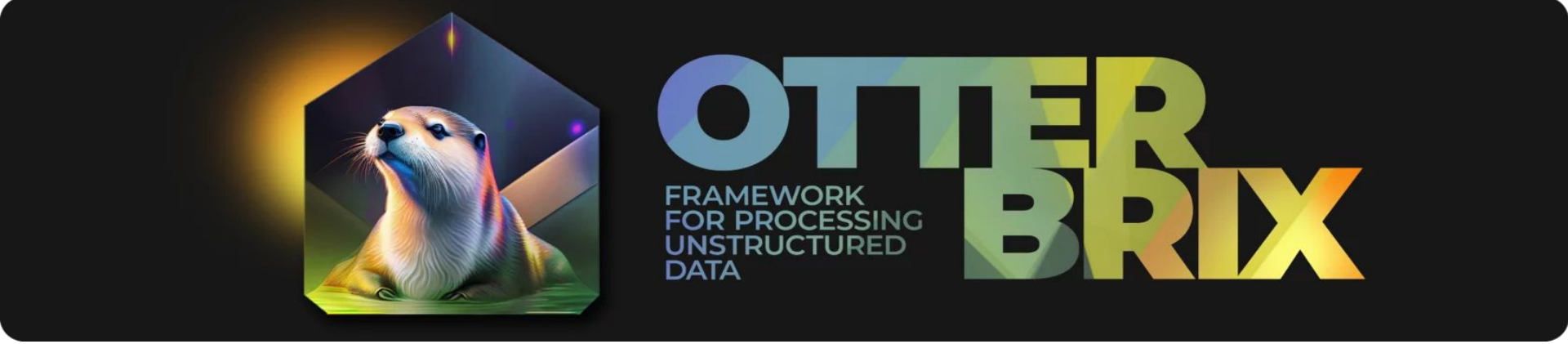
Использование подходов SoA/AoS для работы с многоуровневыми структурами данных

Боргардт Александр

non-profit/community-driven

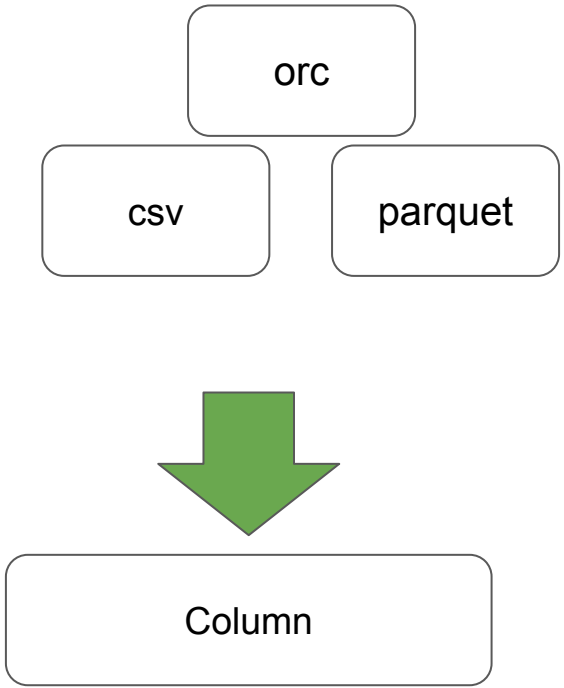
Telegram

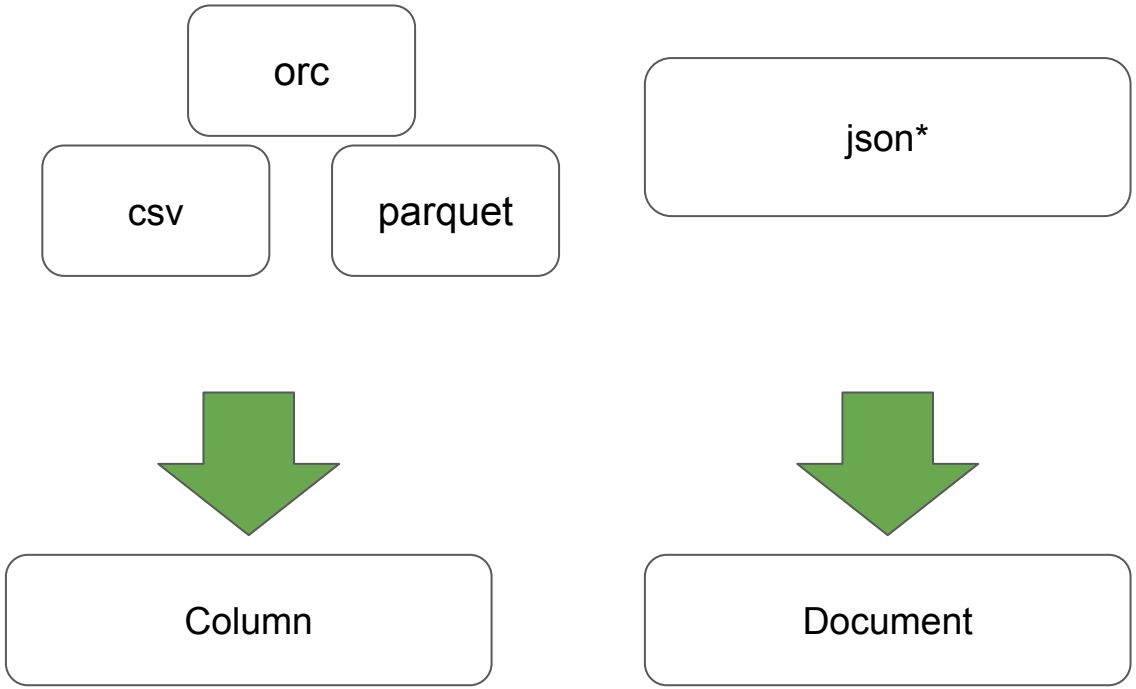


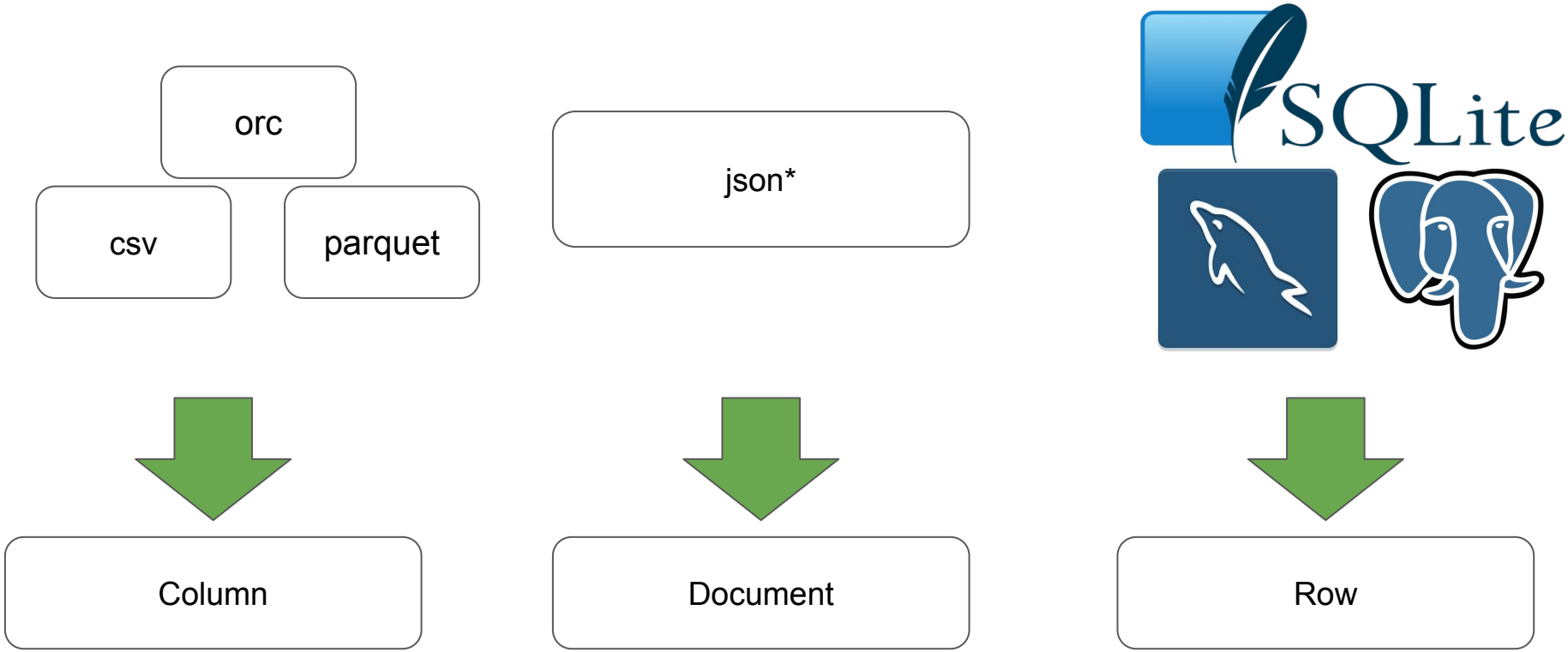
otterbrix.com

Типовое использование Ottebrix

```
from otterbrix from Client
client = Client()
client.execute("..................copy csv.....1 gb.........................")
client.execute("...................copy json 5 gb.....................")
client.execute("..............copy....parquet. 10 gb..................")
client.execute("..............join csv and parquet to save new table ..................")
client.execute("..............read and save dump sqlite ..................")
```


orc
csv
parquet
Column

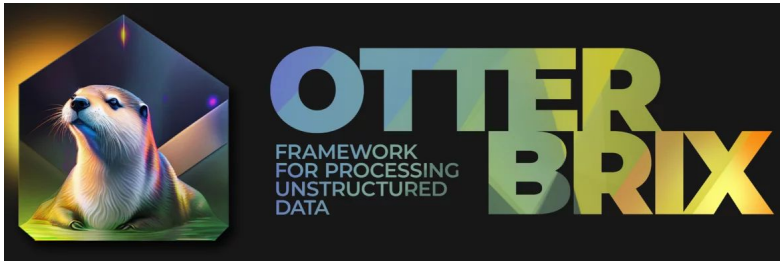

orc
csv
parquet
json*
Column
Document


orc
csv
parquet
json*
SQLite
Column
Document
Row

Column

Document

Row



Какую построить систему обработки для Multi-Modal Storage ?
row column document

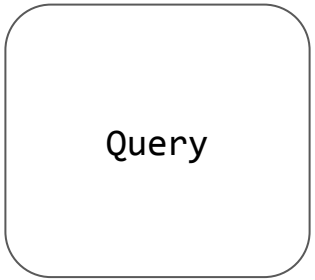
Pipeline
Обработки

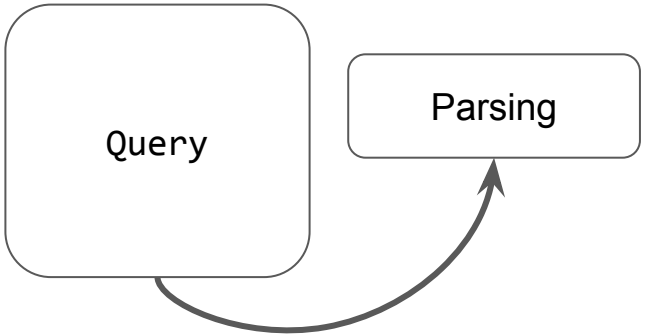
Сиситема титов

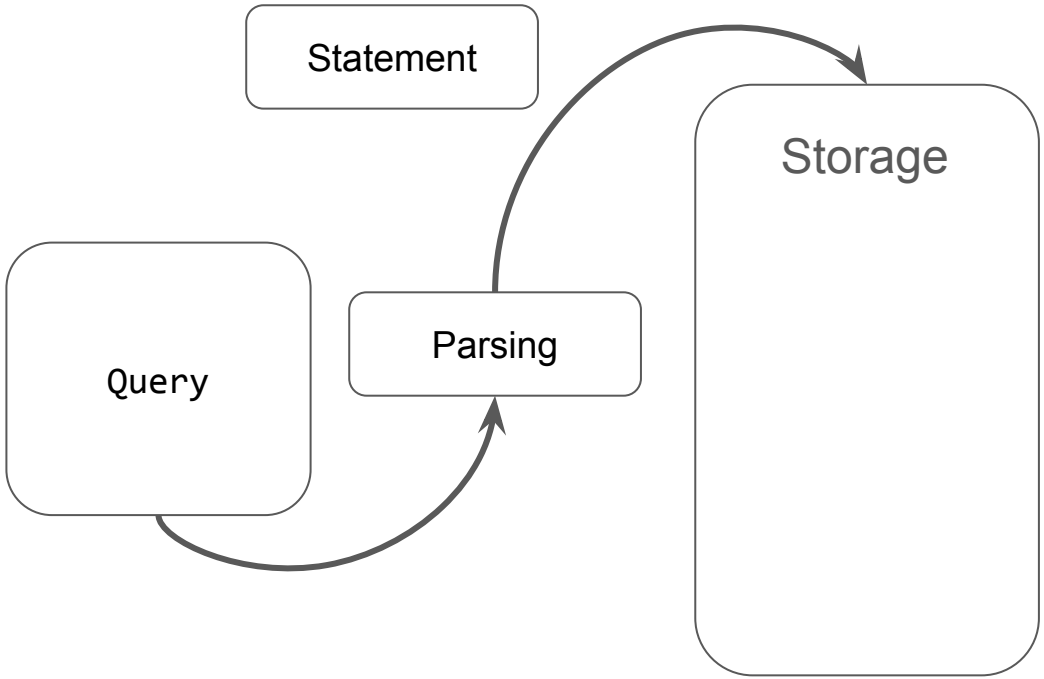
Какую построить систему обработки для Multi-Modal Storage ?
row column document

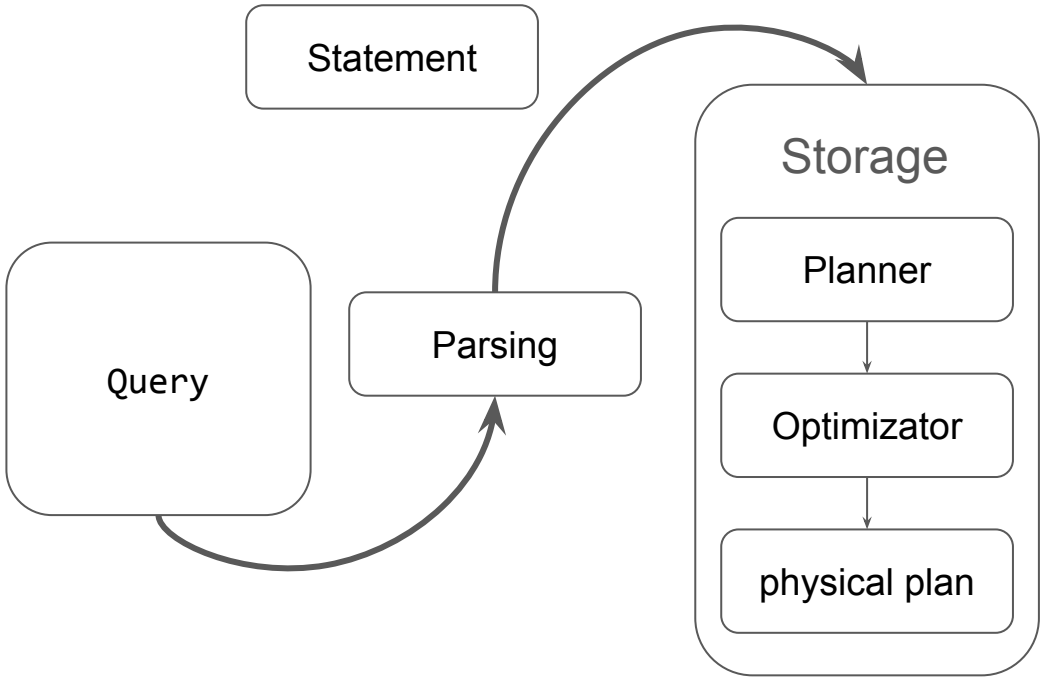
Pipeline
Обработки

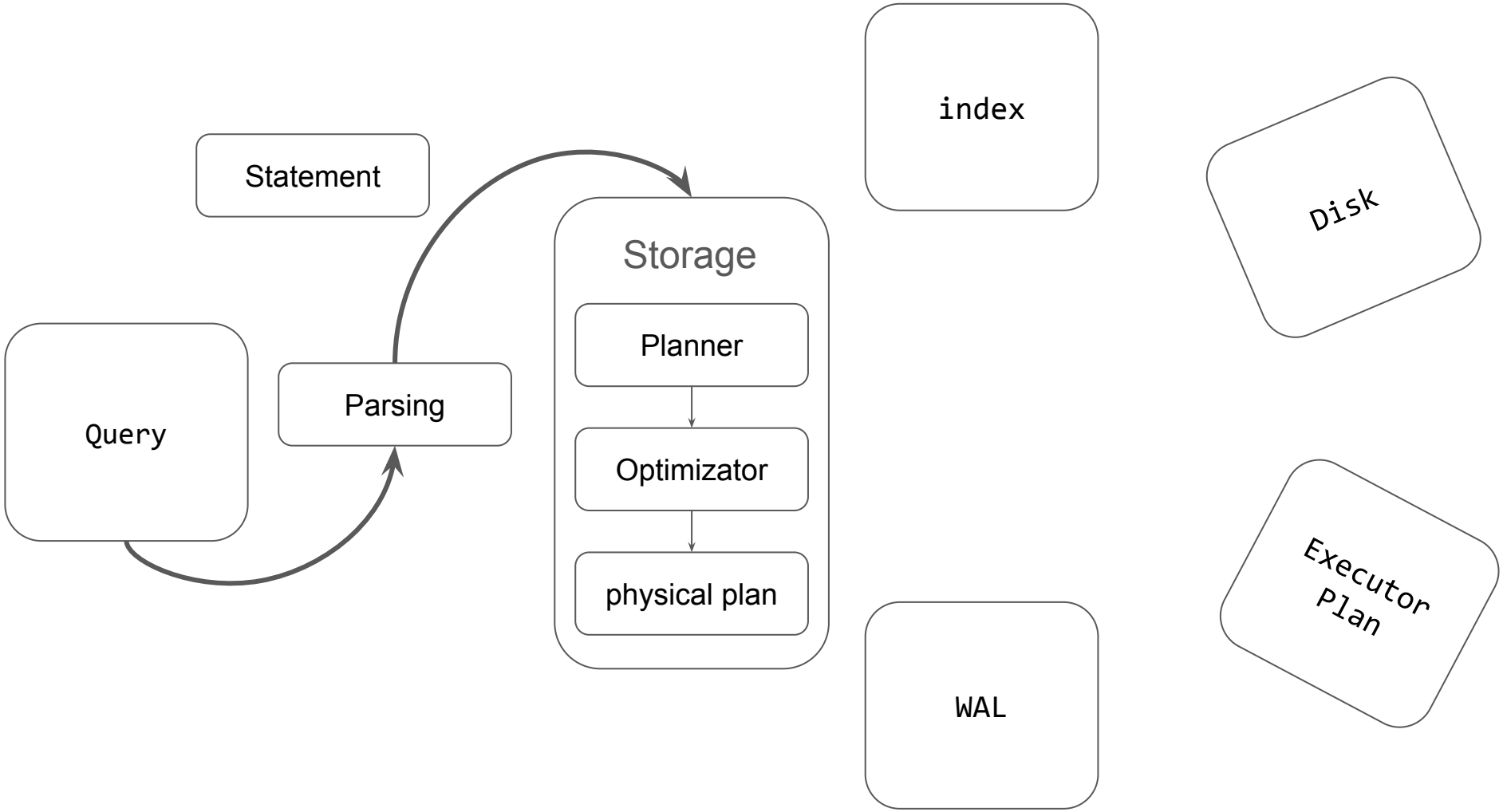
Сиситема титов

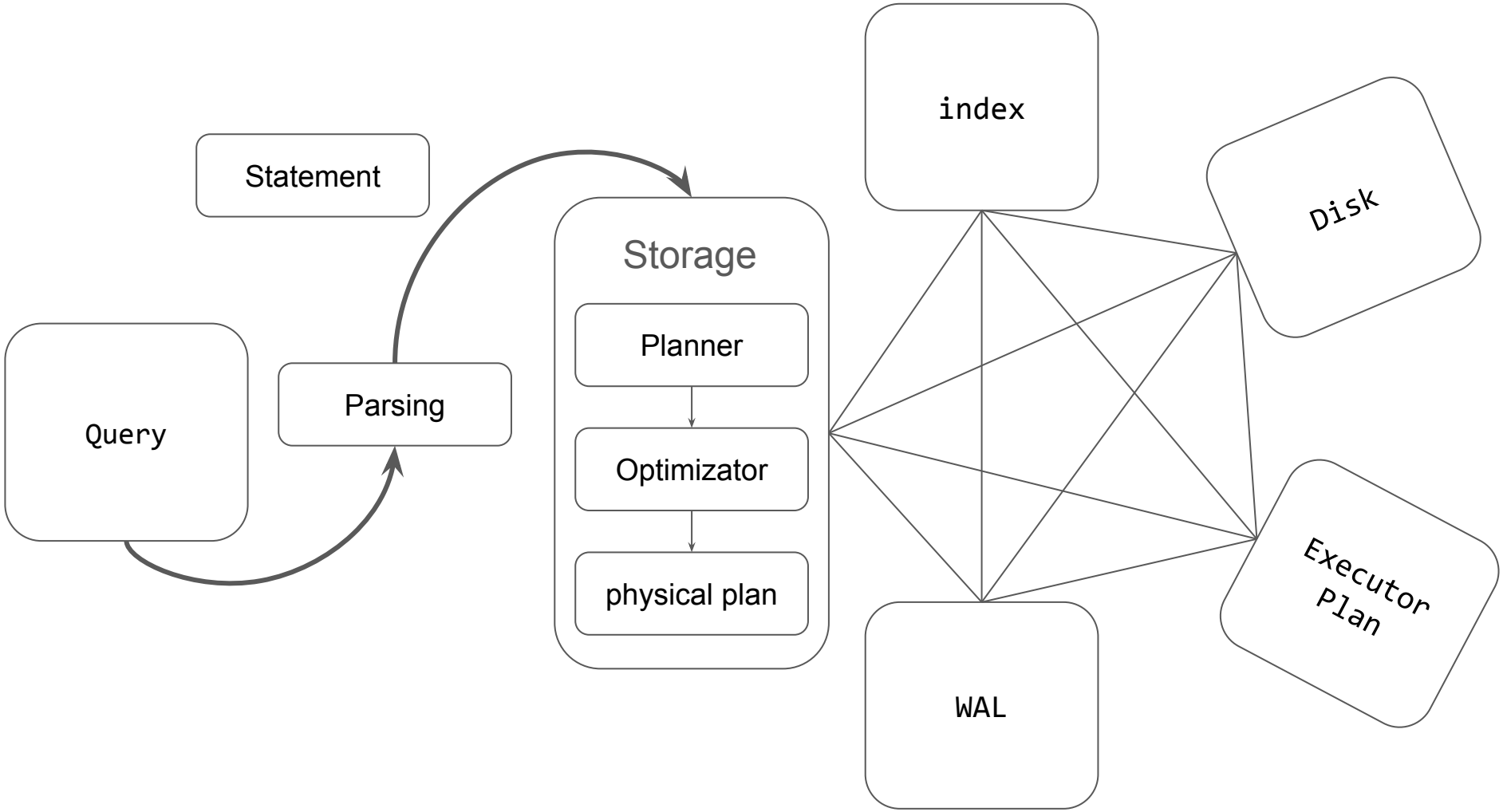

Query


Query
Parsing

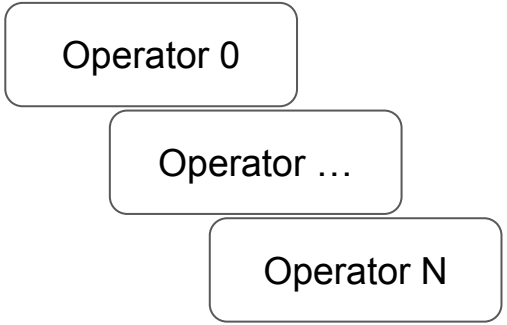

Statement
Storage
Query
Parsing


Statement
Storage
Planner
Optimizator
physical plan
Query
Parsing

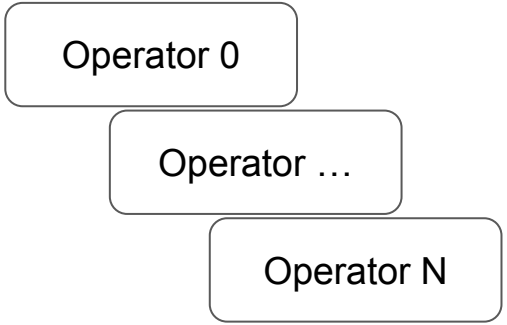

Statement
Query
Parsing
Storage
Planner
Optimizator
physical plan
index
Disk
Executor
Plan
WAL


Statement
Query
Parsing
Storage
Planner
Optimizator
physical plan
index
Disk
Executor
Plan
WAL

life cycle physical plan



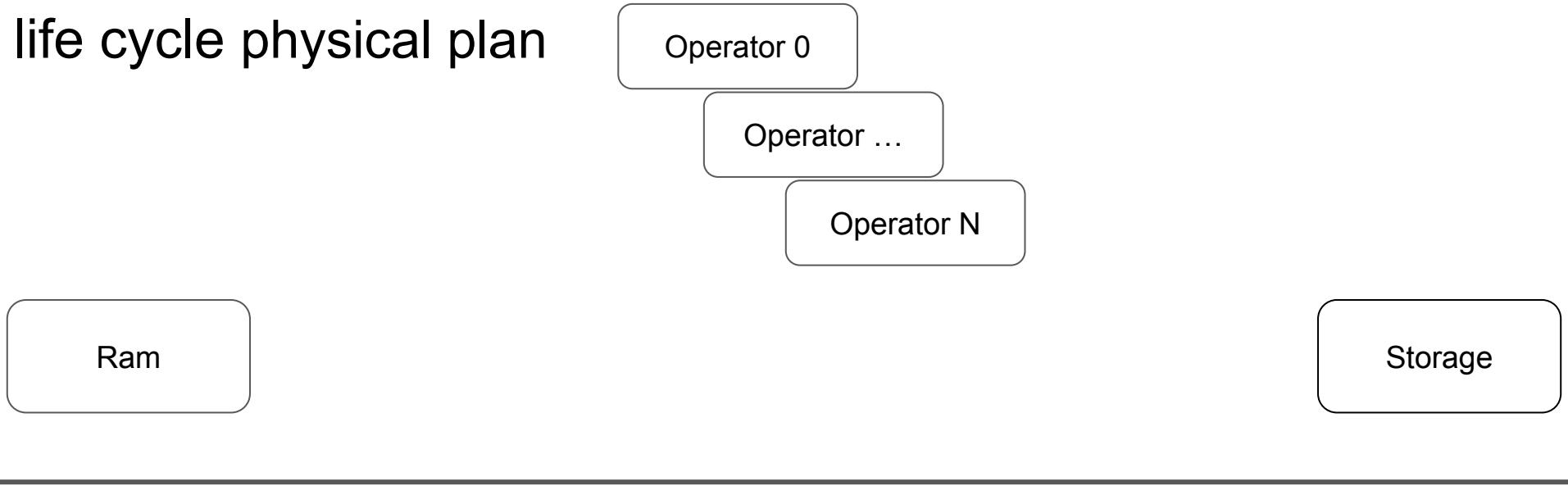
life cycle physical plan



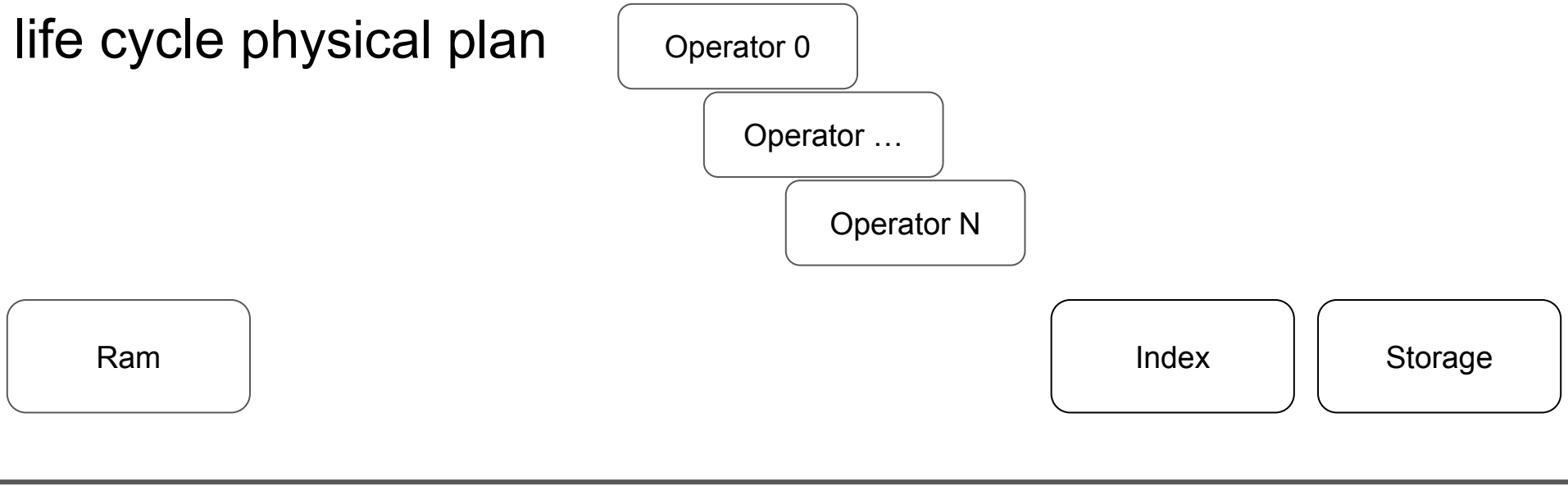
Ram

Disk

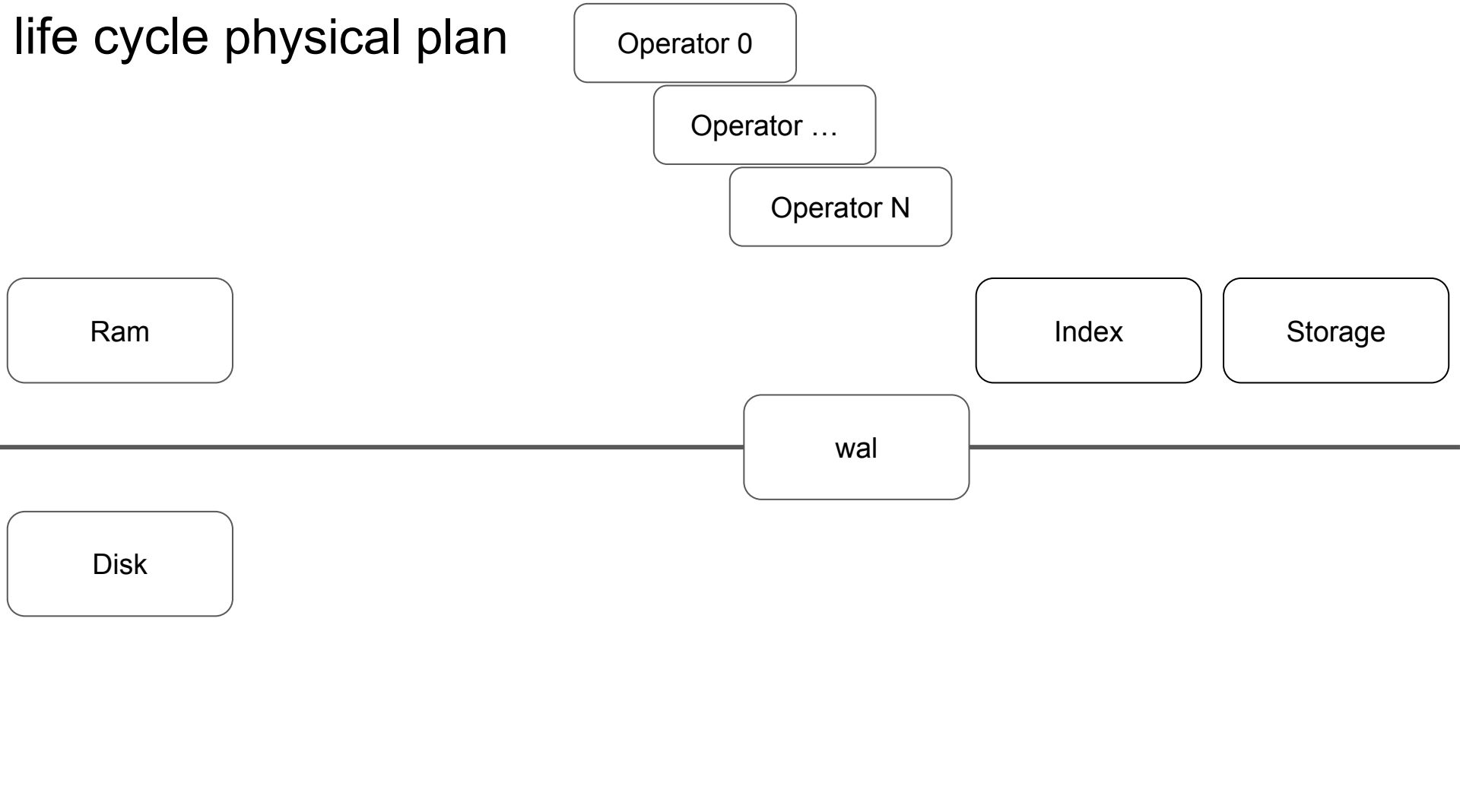
life cycle physical plan



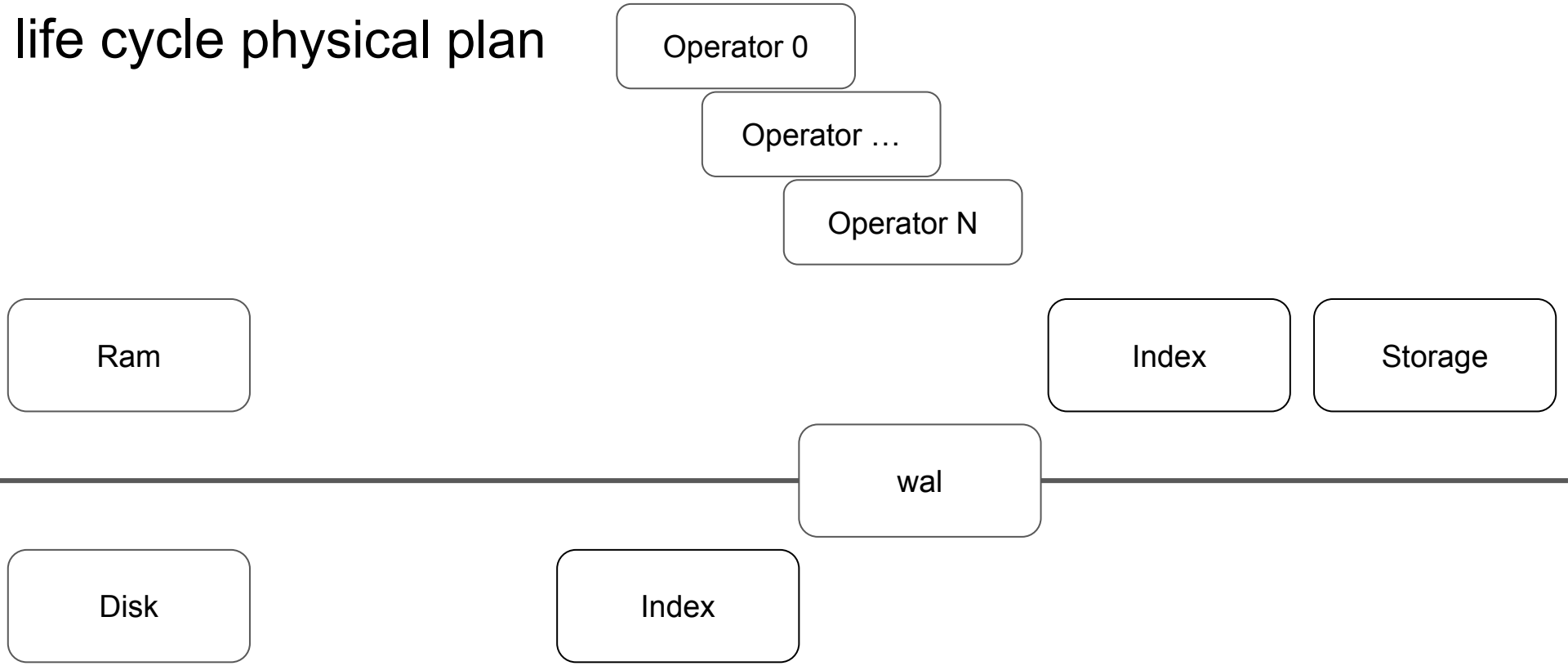
life cycle physical plan



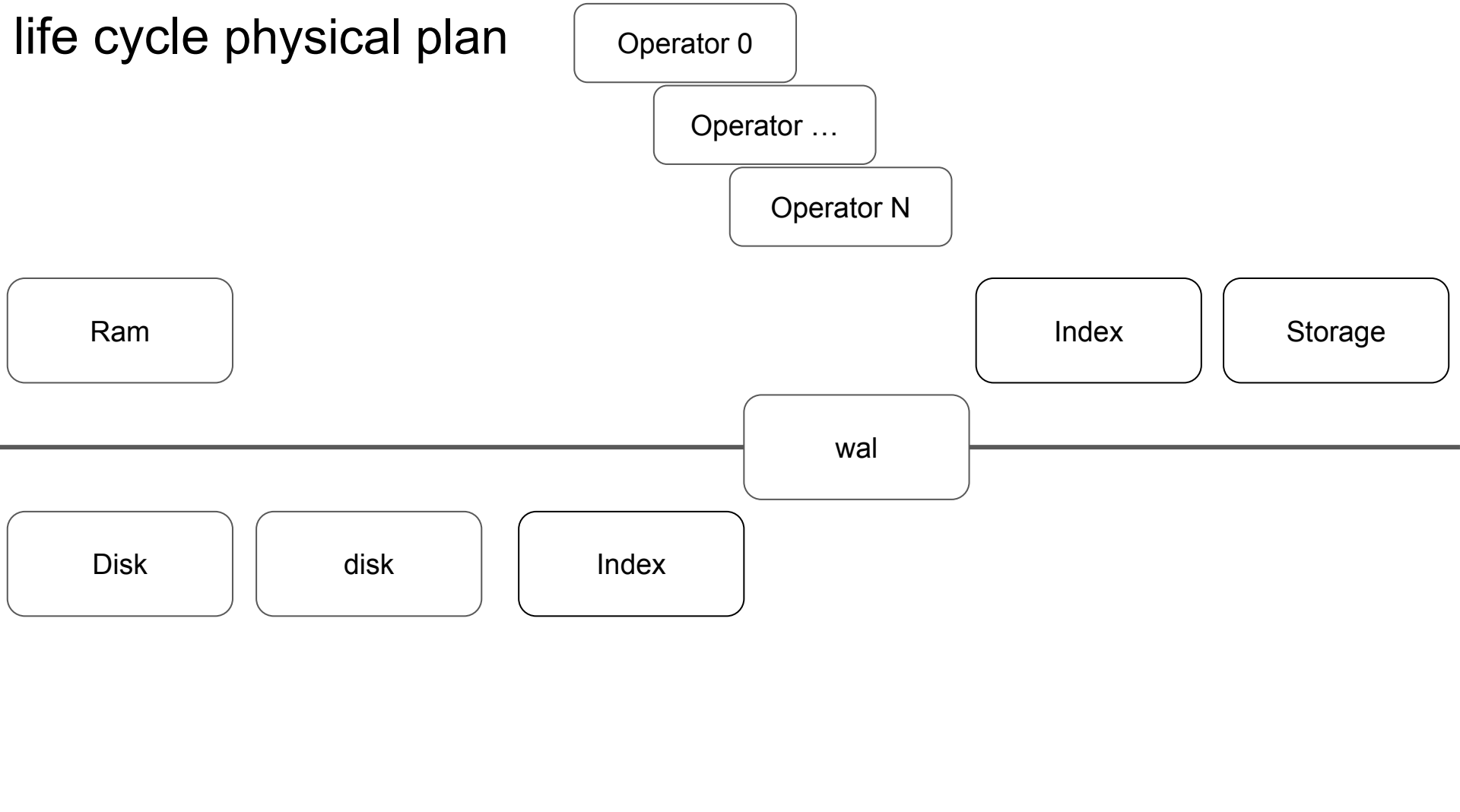
life cycle physical plan



life cycle physical plan



life cycle physical plan



Какую построить систему обработки для Multi-Modal Storage ?
row column document

Pipeline
Обработки

Сиситема титов

Какую построить систему обработки для Multi-Modal Storage ?
row column document

Pipeline
Обработки

Сиситема титов

Row

Logical representation

	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type	Name: metric 3 Type
0				
…				
N				

Logical representation

	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type String	Name: metric 4 Type String
0				
…				
N				

Physical representation

Schema

metric1:int64	metric2:int32	metric3:String	metric4:String

Logical representation

	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type String	Name: metric 4 Type String
0				
…				
N				

Physical representation

Schema

metric1:int64	metric2:int32	metric3:String	metric4:String

Data

0	59	“Foo0”	“Bar0”
1	2	“Foo”	“Bar”
3	4	“Foo1”	“Bar1”

Column

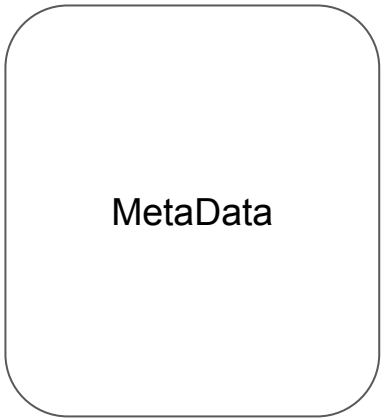
Logical representation

	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type	Name: metric 3 Type
0				
…				
N				

Logical representation

	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type	Name: metric 3 Type
0				
…				
N				

Physical representation



Logical representation

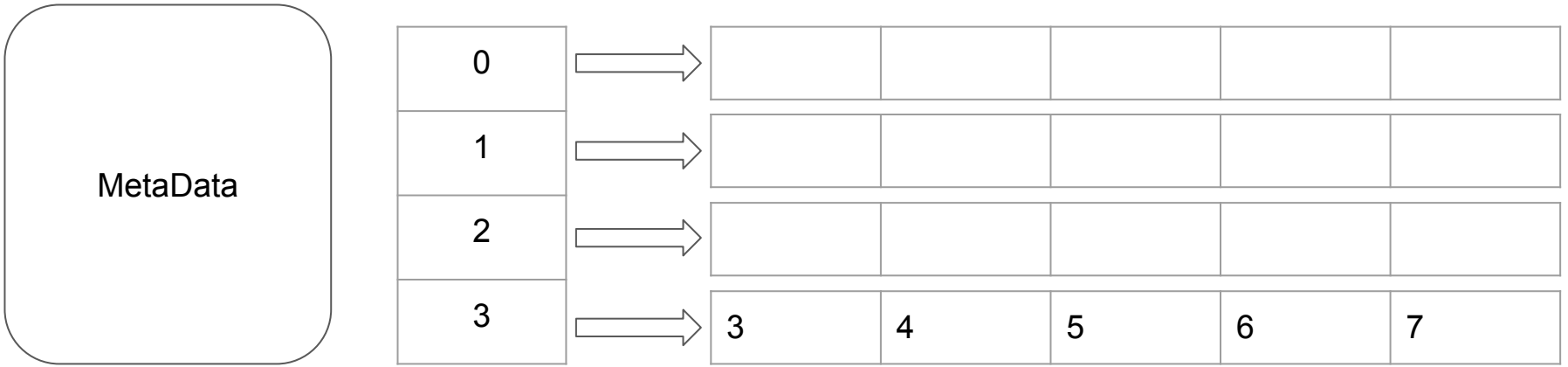
	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type	Name: metric 3 Type
0				
…				
N				

Physical representation

Logical representation

	Name: metric 1 Type int64	Name: metric 2 Type int32	Name: metric 3 Type	Name: metric 3 Type
0				
…				
N				

Physical representation



Типовое использование Ottebrix

```
from otterbrix from Client
client = Client()
client.execute("..................copy csv.....1 gb.........................")
client.execute("...................copy json 5 gb.....................")
client.execute("..............copy....parquet. 10 gb..................")
client.execute("..............join csv and parquet to save new table ..................")
client.execute("..............read and save dump sqlite ..................")
```

Типовое использование Ottebrix part 2

```
from otterbrix from Client
client = Client()
client.execute("..................copy csv.....1 gb.........................")
client.execute("...................copy json 5 gb.....................")
client.execute("..............copy....parquet. 10 gb.................")
client.execute("..............join csv and parquet to save new table .................")
client.execute("..............read and save dump sqlite .................")
//*-----------------------------------------------------------------------------*//
client.execute("..............select join csv and parquet .................")
client.execute("..............select join parquet and json.................")
client.execute("..............select join dump sqlite and json.................")
```

Что делать когда не понятен структура ?

Что делать когда не понятен структура ?
Document !

Library	Roundtrip Time (s)	Write (MB/s)	Read (MB/s)
simdjson (on demand)			
RapidJSON			
Boost.JSON (direct)			
nlohmann			

nlohmann

```
Class value {
        union json_value{
                object_t* object;
                array_t* array;
                string_t* string;
                binary_t* binary;
                boolean_t boolean;
                number_integer_t number_integer;
                number_unsigned_t number_unsigned;
                number_float_t number_float;
        }
}
```

nlohmann

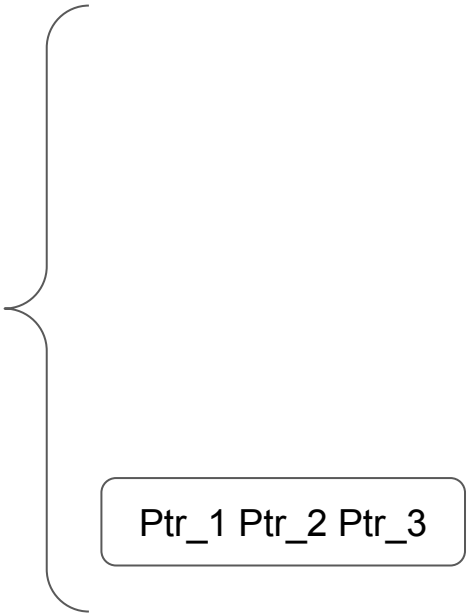
```
Class value {
        union json_value{
                object_t* object; // <-  std::map
                array_t* array;    // <- std::vector,
                string_t* string;   // <- std::string
                binary_t* binary; // <- std::vector<std::uint8_t>
                boolean_t boolean;
                number_integer_t number_integer;
                number_unsigned_t number_unsigned;
                number_float_t number_float;
        }
}
```

RapidJSON

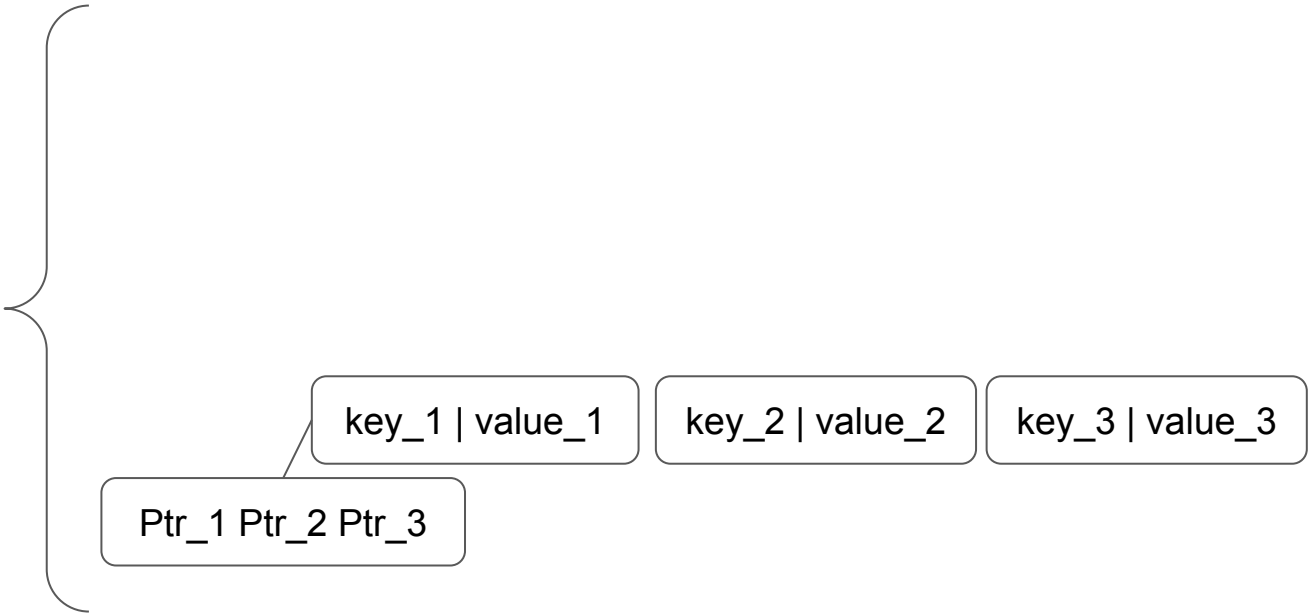
```
struct Flag {};
struct String {};
struct ShortString {};
union Number {};
struct ObjectData {};
struct ArrayData {};
union Data {
        String s;
        ShortString ss;
        Number n;
        ObjectData o;
        ArrayData a;
        Flag f;
};
```

Don't use std
Use custom container and bitpointer optimization
Use stack allocator

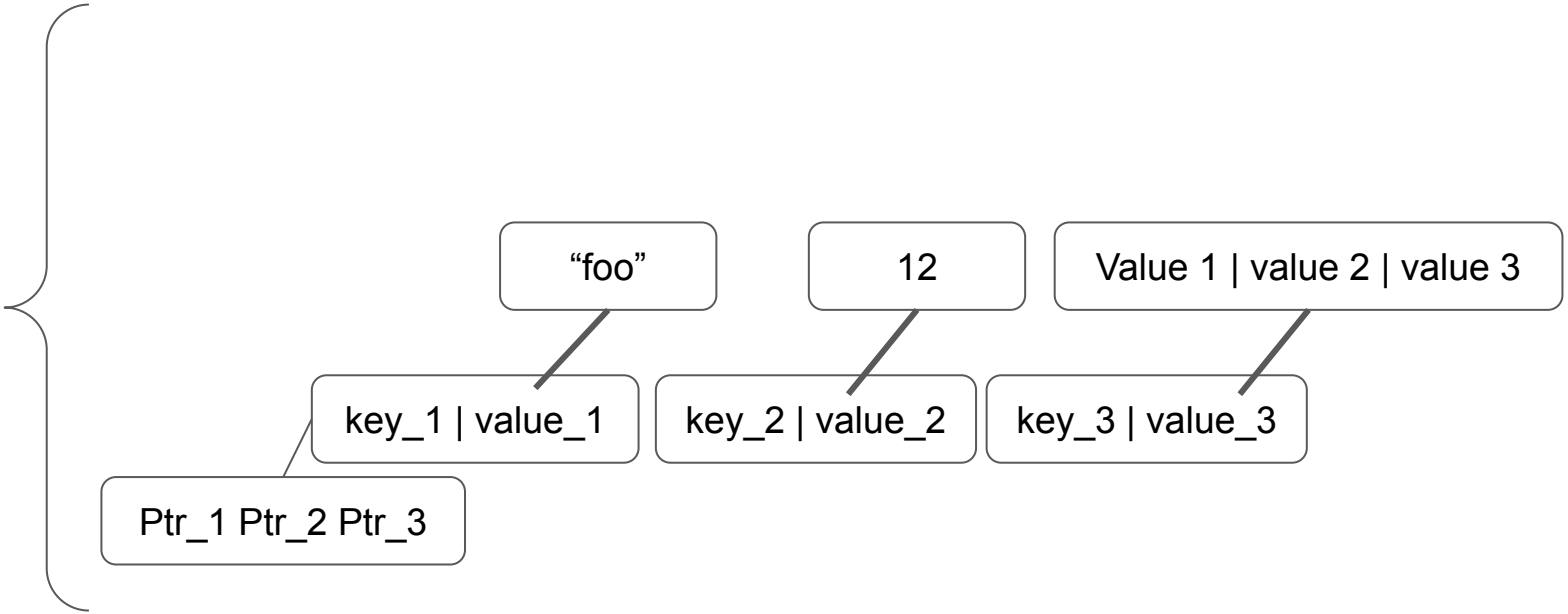
RapidJSON



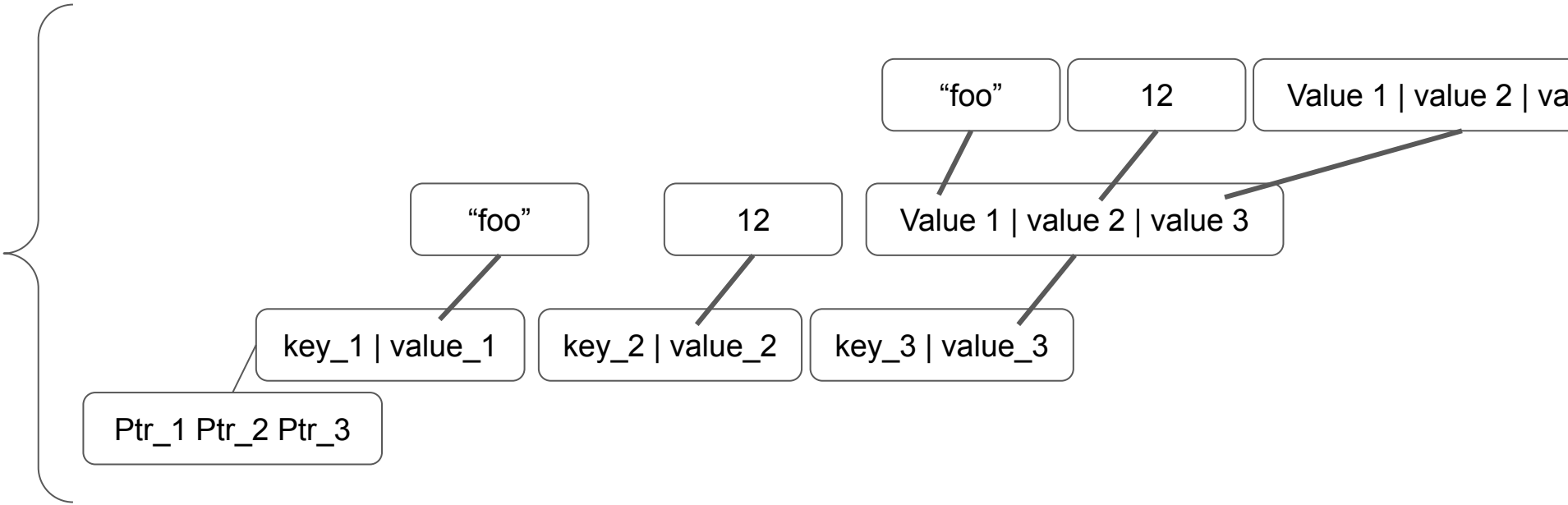
RapidJSON



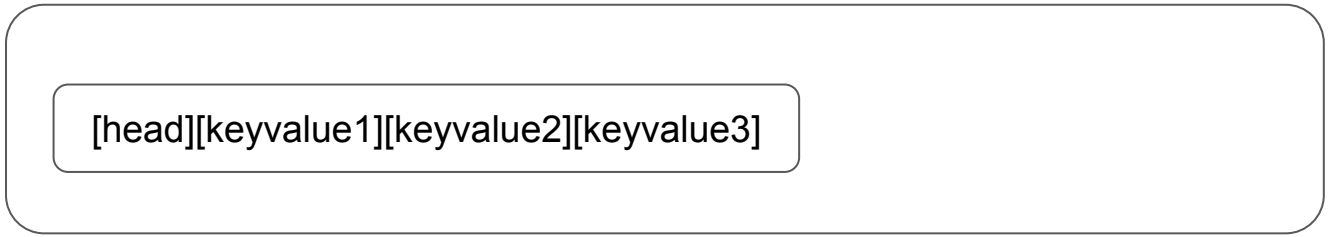
RapidJSON


“foo”
12
Value 1 | value 2 | value 3
key_1 | value_1
key_2 | value_2
key_3 | value_3
Ptr_1 Ptr_2 Ptr_3

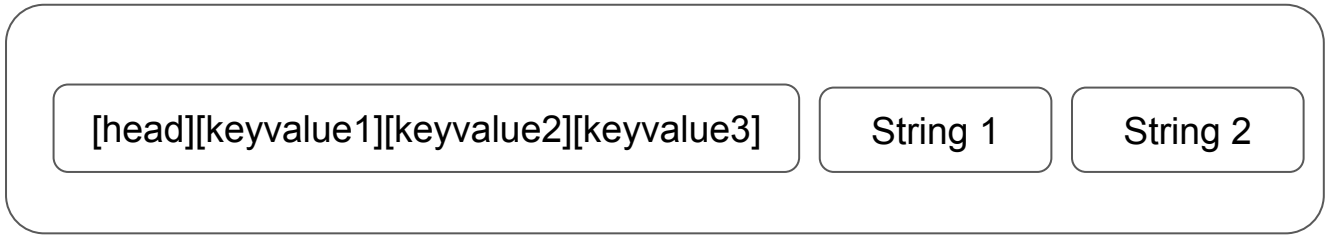
RapidJSON


“foo”
12
Value 1 | value 2 | va
“foo”
12
Value 1 | value 2 | value 3
key_1 | value_1
key_2 | value_2
key_3 | value_3
Ptr_1 Ptr_2 Ptr_3

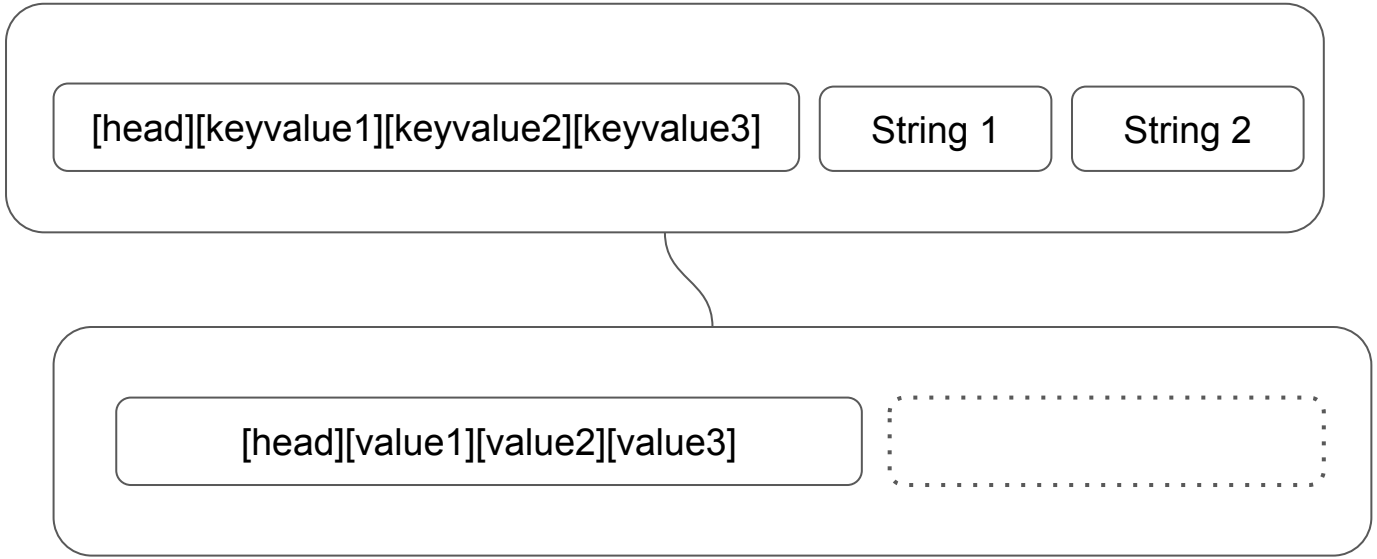
Boost.JSON (direct)



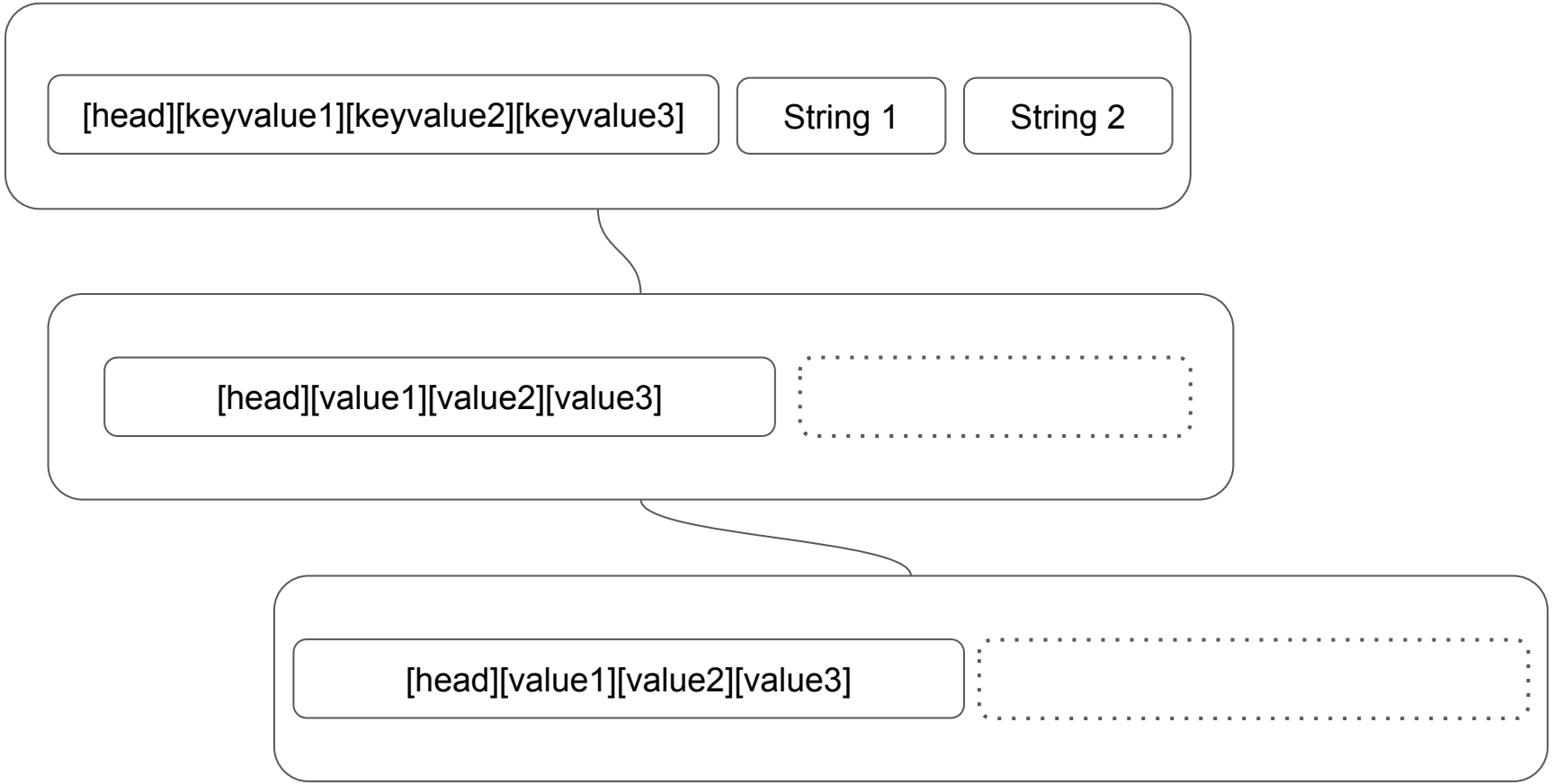
Boost.JSON (direct)



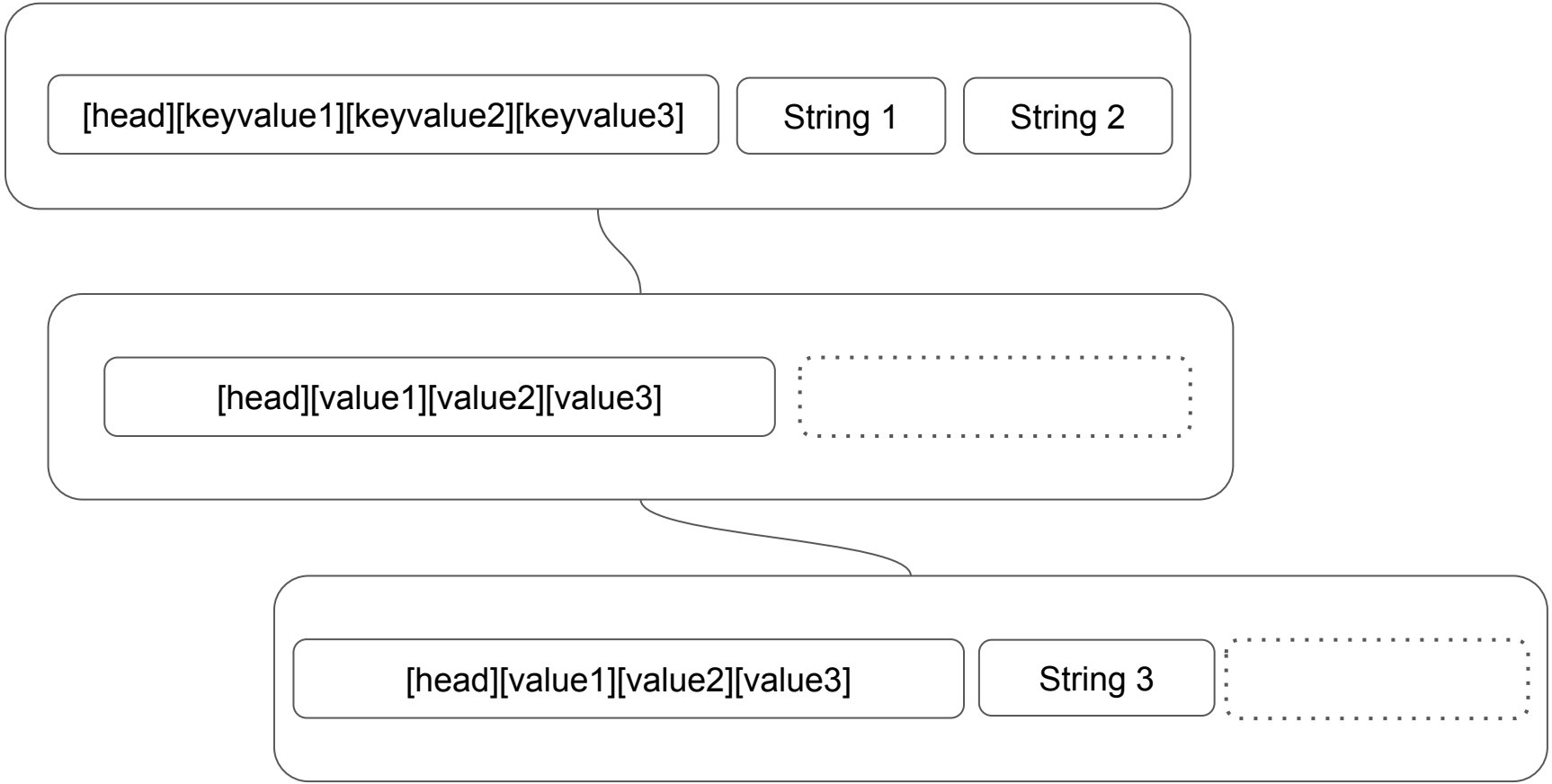
Boost.JSON (direct)



Boost.JSON (direct)



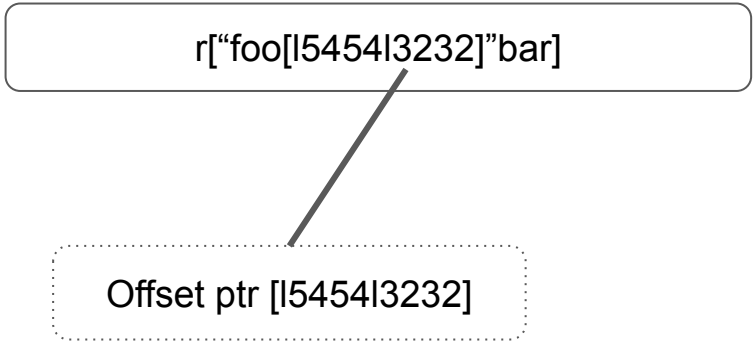
Boost.JSON (direct)



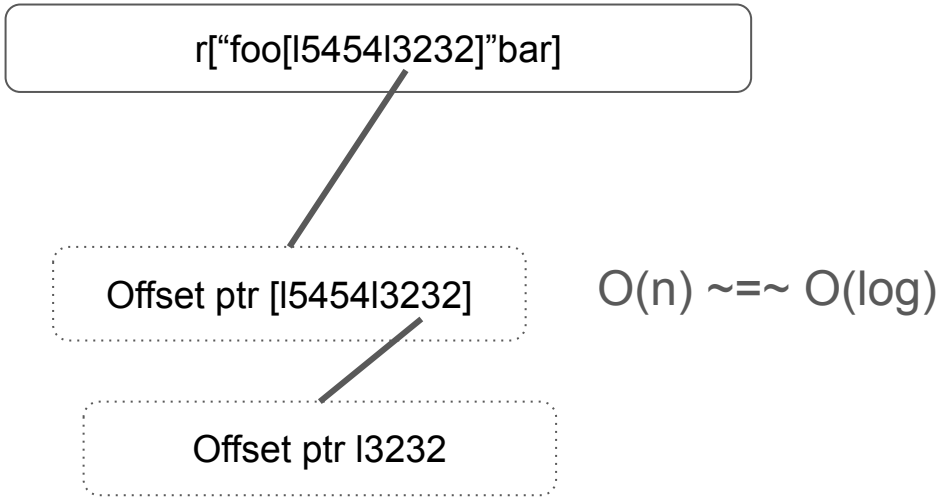
simdjson

r[“foo[l5454l3232]”bar]

simdjson



simdjson



O(n) ~=~ O(log)

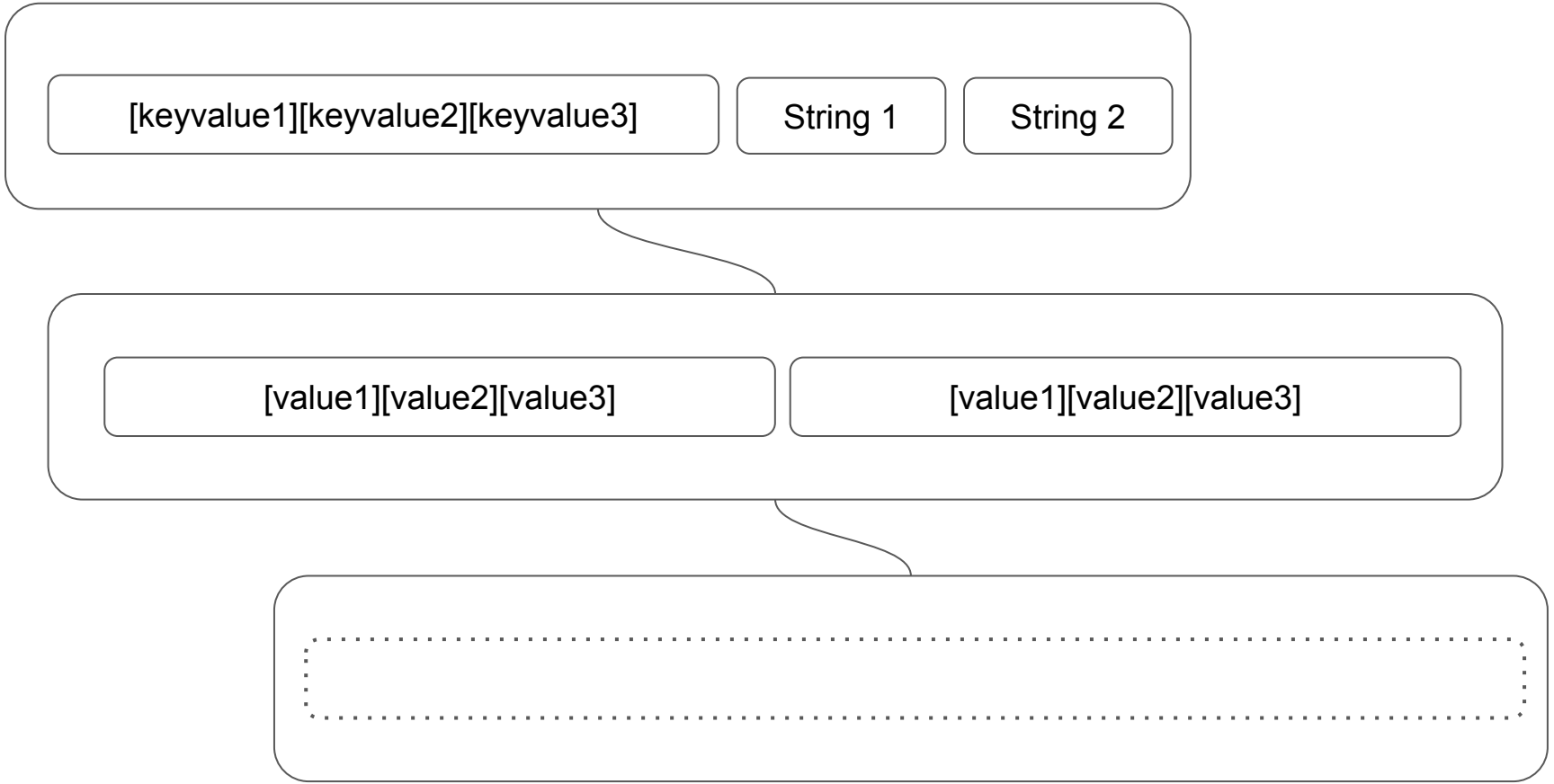
Library	Roundtrip Time (s)	Write (MB/s)	Read (MB/s)
simdjson (on demand)			
RapidJSON			
Boost.JSON (direct)			
nlohmann			

```
{
  "fixed_object": {
     "int_array": [0, 1, 2, 3, 4, 5, 6],
     "float_array": [0.1, 0.2, 0.3, 0.4, 0.5, 0.6],
     "double_array": [3288398.238, 233e22, 289e-1, 0.928759872, 0.22222848, 0.1, 0.2, 0.3, 0.4]
  },
  "fixed_name_object": {
     "name0": "James",
     "name1": "Abraham",
     "name2": "Susan"
  },
  "another_object": {
     "string": "here is some text",
     "boolean": false,
     "nested_object": {
        "v3s": [[0.12345, 0.23456, 0.001345],
                [0.3894675, 97.39827, 297.92387],
                [18.18, 87.289, 2988.298]],
        "id": "298728949872"
     }
  },
  "string_array": ["Cat", "Dog", "Elephant", "Tiger"]
}
```

Library	Roundtrip Time (s)	Write (MB/s)	Read (MB/s)
simdjson (on demand)	N/A	N/A	1198
RapidJSON	3.68	287	441
Boost.JSON (direct)	4.77	201	442
nlohmann	15.12	87	82

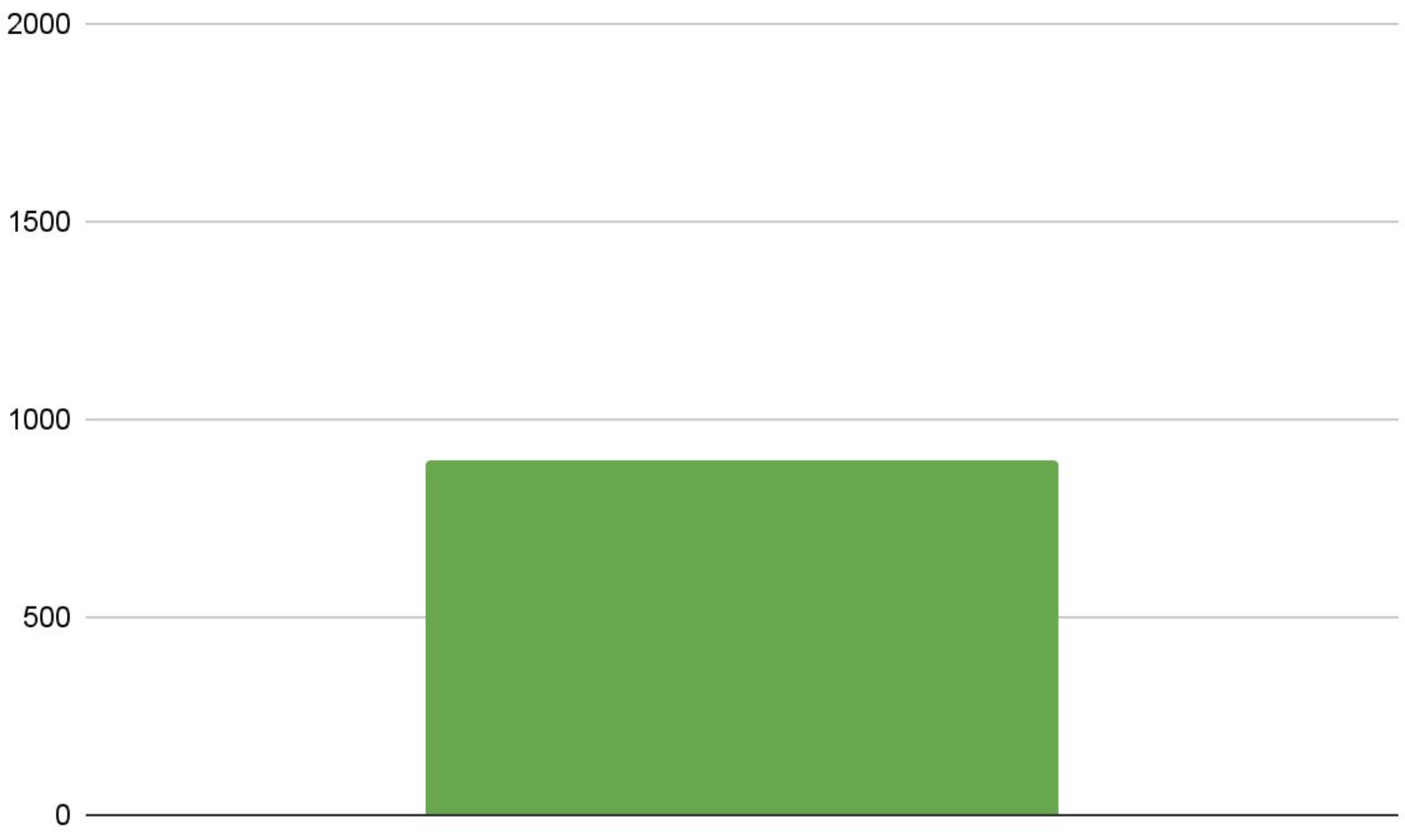
Document

Document ?


[keyvalue1][keyvalue2][keyvalue3]
String 1
String 2
[value1][value2][value3]
[value1][value2][value3]

Benchmark

Lookup(60%) / Update(15%) / Deleted(25%)


2000
1500
1000
500
0

А что делать ?

SoA / AoS

SoA

```
struct coordinate {
    int x;
    int y
    int z;
}

std::vector<coordinate>coordinates
```

SoA

```
struct coordinate {
   int x;
   int y
   int z;
}

std::vector<coordinate>coordinates
```

SoA

```
std::vector<int>xs;
std::vector<int>ys;
std::vector<int>zs;
```

```
class item{
public:
   double const&      get_myDouble() const { return mmy_double; }
   char const&        get_my_char() const { return mmy_char };
   std::string const& get_my_string() const { return mmy_string };

   double&      my_double(){ return mmy_double };
   char&        my_char() { return mmy_char; }
   std::string& my_string() { return mmy_string; }

private:
   double      mmy_double;
   char        mmy_char;
   std::string mmy_string;
};
```

```
template<typename T>
using vector = std::vector<T, std::allocator<T>>;

enum component : int {
   emy_double,
   emy_char,
   emy_string,
   emy_padding
};

struct pad{
       char mPad[1500];
};

template<typename ... T>
struct item : public std::tuple<T ...>{
   using std::tuple<T...>::tuple;
   auto & my_double(){return std::get<emy_double>(*this);}
   auto & my_char()  {return std::get<emy_char>(*this);}
   auto & my_string(){return std::get<emy_string>(*this);}
};

using container_t = BaseContainer<MyVector, DataLayout::SoA, Item<double, char, std::string, pad> >;
```

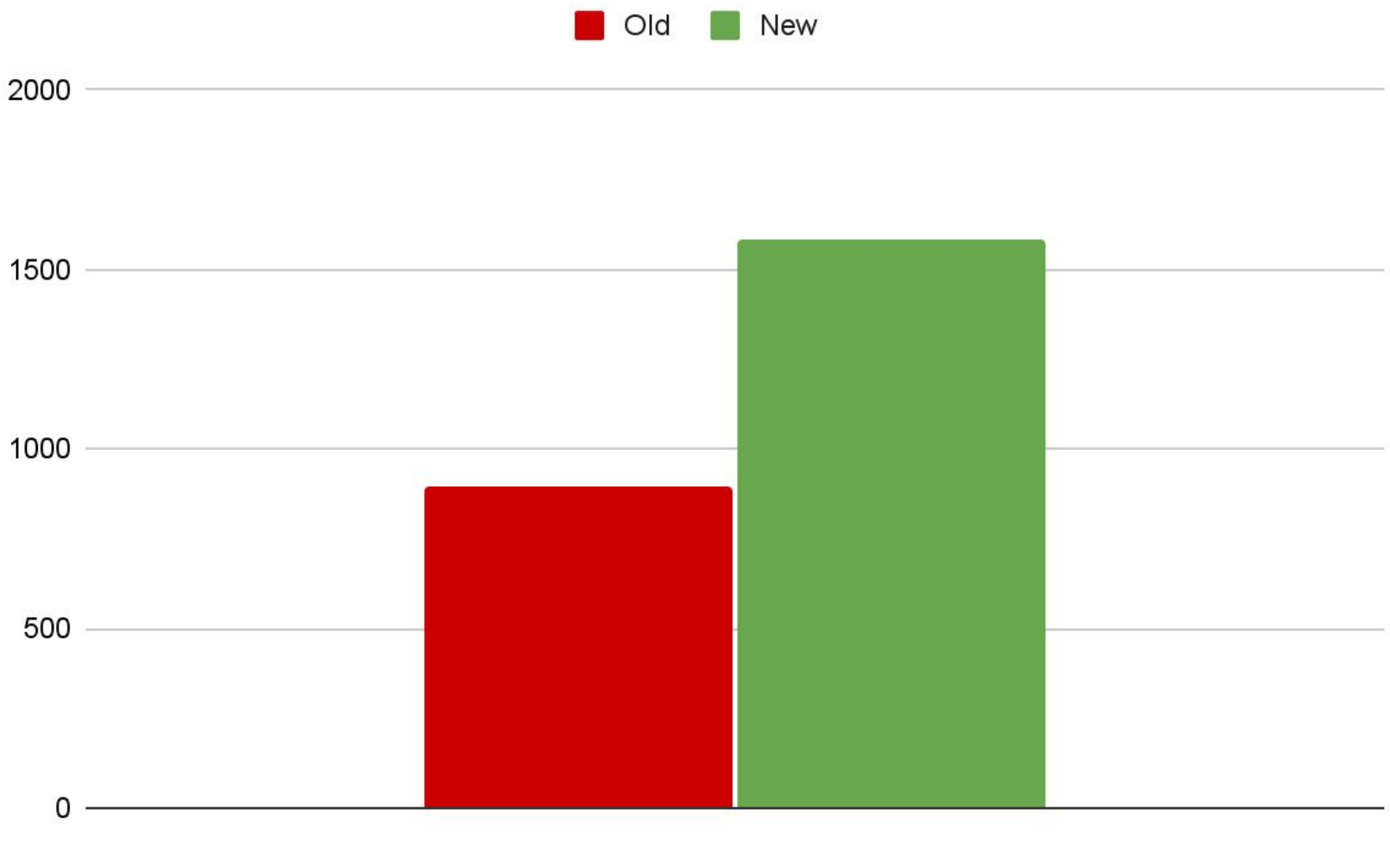
```
template<typename T>
using vector = std::vector<T, std::allocator<T>>;

enum component : int {
   emy_double,
   emy_char,
   emy_string,
   emy_padding
};

struct pad{
       char mPad[1500];
};

template<typename ... T>
struct item : public std::tuple<T ...>{
   using std::tuple<T...>::tuple;
   auto & my_double(){return std::get<emy_double>(*this);}
   auto & my_char()  {return std::get<emy_char>(*this);}
   auto & my_string(){return std::get<emy_string>(*this);}
};

using container_t = BaseContainer<MyVector, DataLayout::SoA, Item<double, char, std::string, pad> >;
```


Old
New
2000
1500
1000
500
0

Что делать когда не понятен структура ?
Document !

Что делать когда не понятен структура ?
Document !
Document != Json :(

А что такое Document если не json?

Document

Single type system

Document

Single type system

MVCC

Document

Single type system

MVCC

Easy transform data and format

- document <-> row
- document <-> column
- document <-> GPU

Document

Single type system

MVCC

Easy transform data and format

    document <-> row

    document <-> column

    document <-> GPU

cheap Lookup / merge / split / update

Document

Single type system

MVCC

Easy transform data and format

    document <-> row

    document <-> column

    document <-> GPU

cheap Lookup / merge / split / update

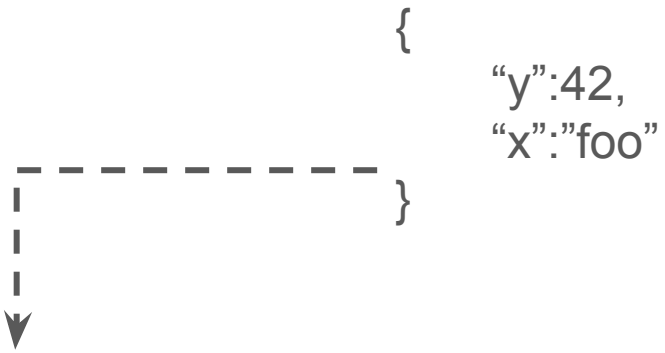
cheap serialization or zero serialization

Library	Roundtrip Time (s)	Write (MB/s)	Read (MB/s)
simdjson (on demand)	N/A	N/A	1198
RapidJSON	3.68	287	441
Boost.JSON (direct)	4.77	201	442
nlohmann	15.12	87	82

Document cread/read

```
{
    “y”:42,
    “x”:”foo”
}
```

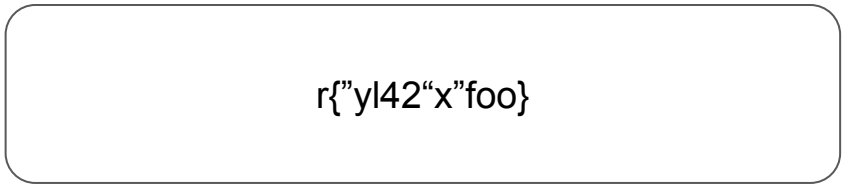
Document cread/read




```
auto* allocator = make_synchronized_pool_resource();
auto doc = document_from_json(json, allocator);
```

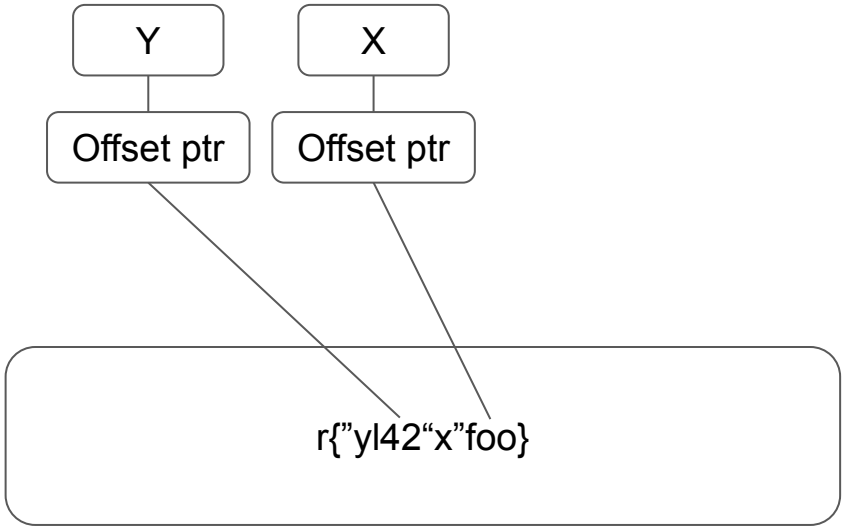
Document cread/read

```
{
    “y”:42,
    “x”:”foo”
}
```



imMutable

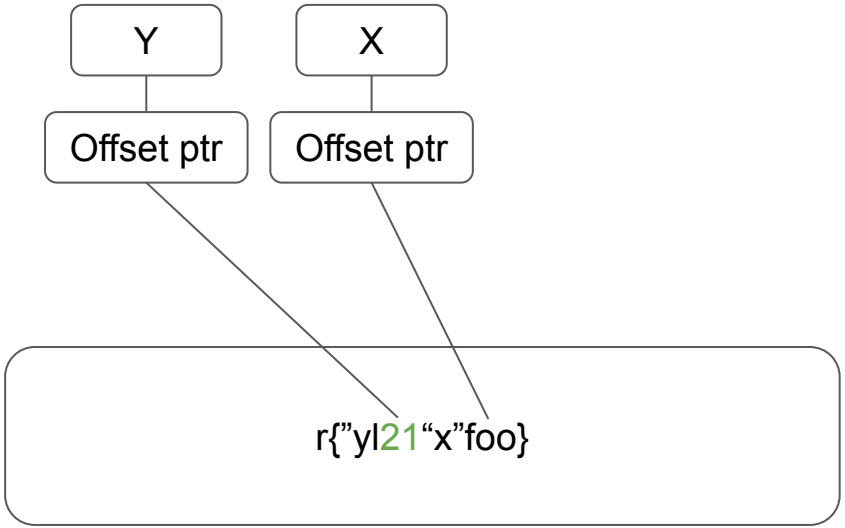
Document cread/read



imMutable

```
{
    “y”:42,
    “x”:”foo”
}
```

Document: update

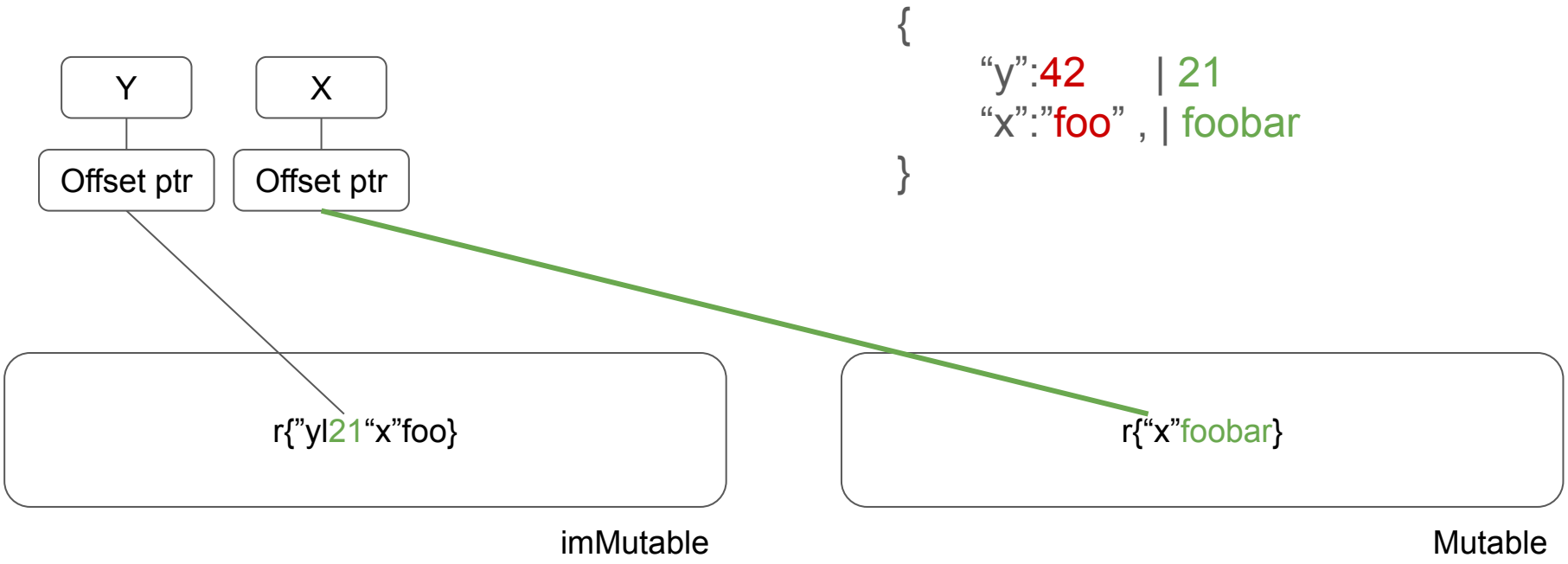


imMutable

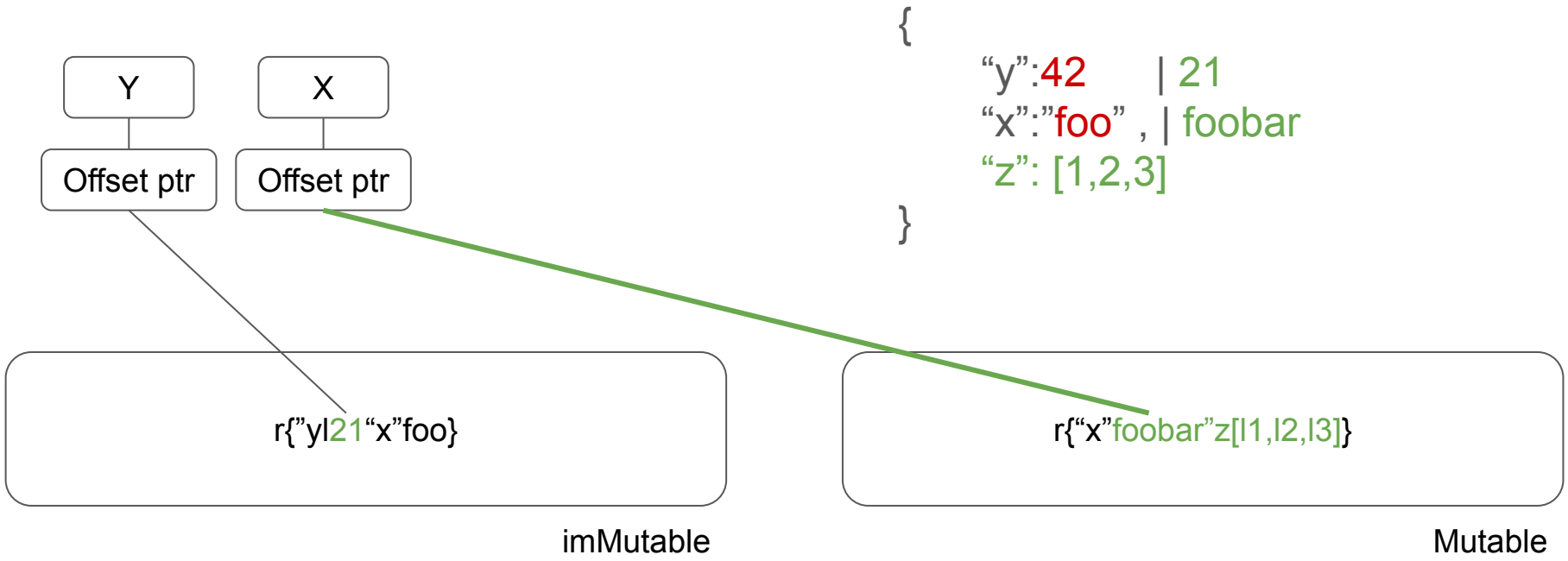
```
{
    “y”:42      | 21
    “x”:”foo”
}
```

Mutable

Document: update



Document: update



Document: merge / split

```
const std::string json = R"({
     "obj": {
          "double": 2.3,
          "arr": [
               {
                 "hello": "world"
               }
             ]
     }
})";
```

```
const std::string json1 =
R"({
     "obj": {
          "long": 5,
          "bool": true
     }
})";
```

Document: merge / split

```
const std::string json = R"({
    "obj": {
        "double": 2.3,
        "arr": [
            {
              "hello": "world"
            }
          ]
    }
})";
```

```
const std::string json1 =
R"({
    "obj": {
        "long": 5,
        "bool": true
    }
})";
```

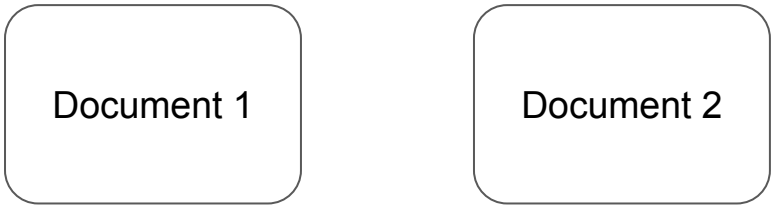
Document: merge / split / delete

```
const std::string json = R"({
    "obj": {
        "double": 2.3,
        "arr": [
            {
              "hello": "world"
            }
          ]
    }
})";
```

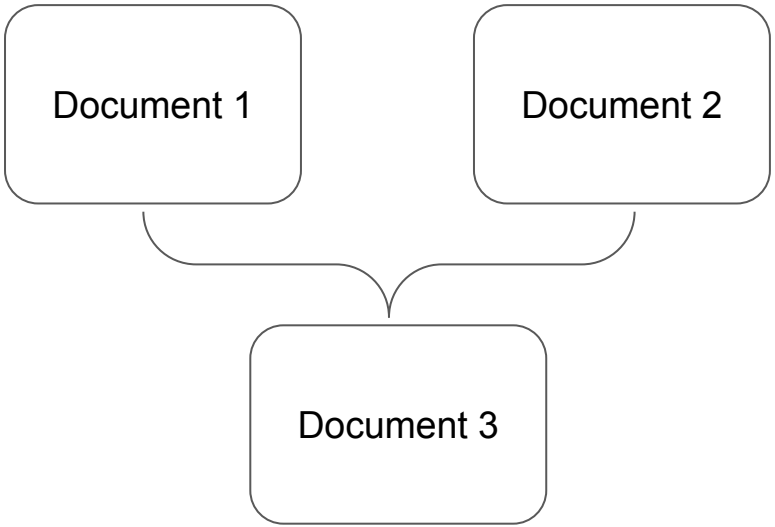
```
const std::string json1 =
R"({
    "obj": {
        "long": 5,
        "bool": true
     }
})";
```

```
auto* gc = make_garbage_collector();
auto* allocator = make_tracker_allocator();
auto doc = document_from_json(gc, allocator, json);
auto doc1 = document_from_json(gc, allocator, json1);
auto merged_doc = merge(gc, doc, doc1);
```

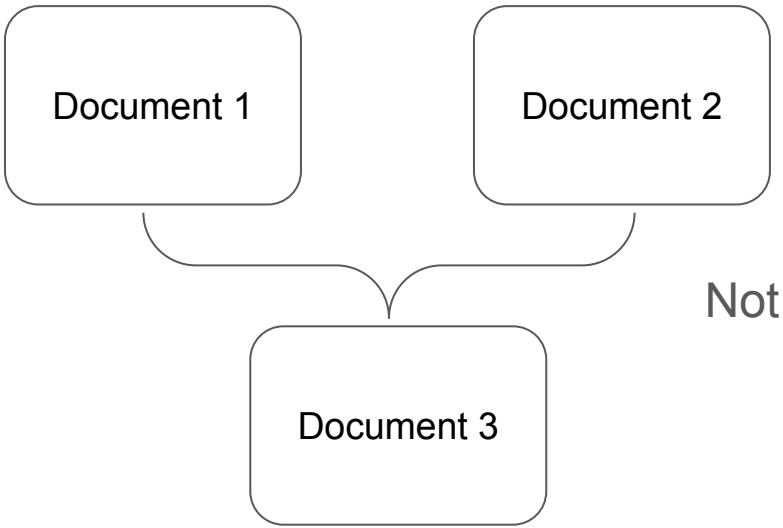
Document: merge / split / delete



Document: merge / split / delete

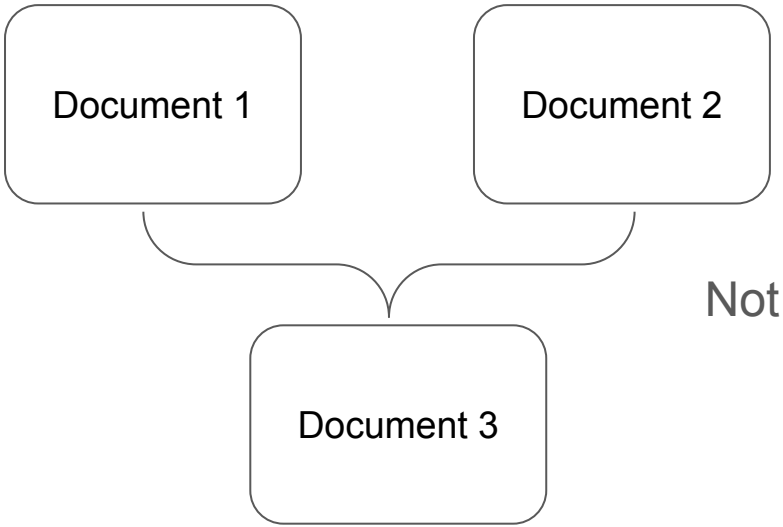


Document: merge / split / delete



Not allocator

Document: merge / split / delete

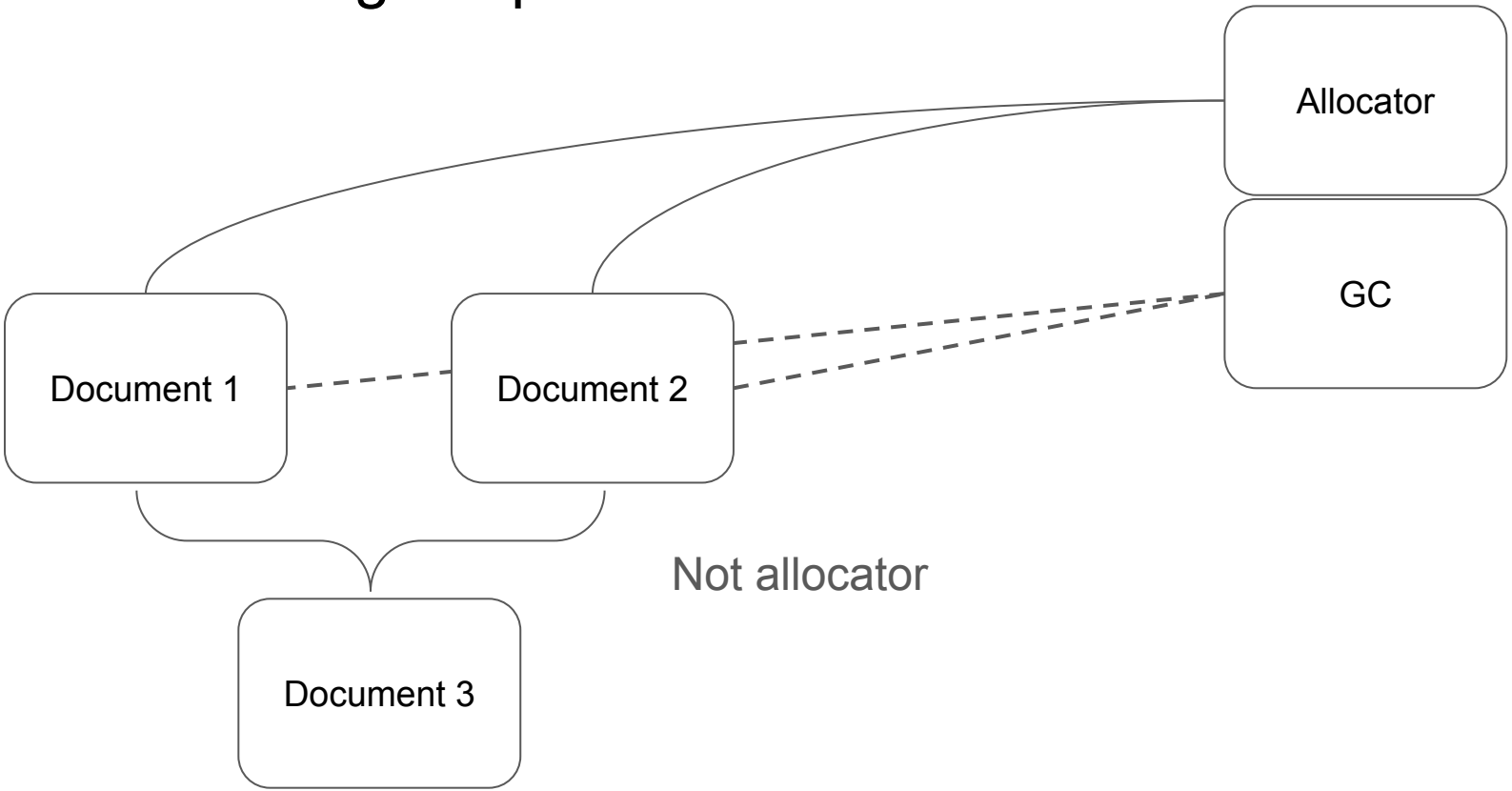


Not allocator

Allocator

GC

Document: merge / split / delete

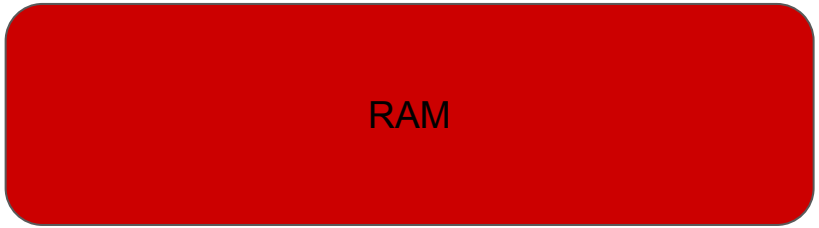


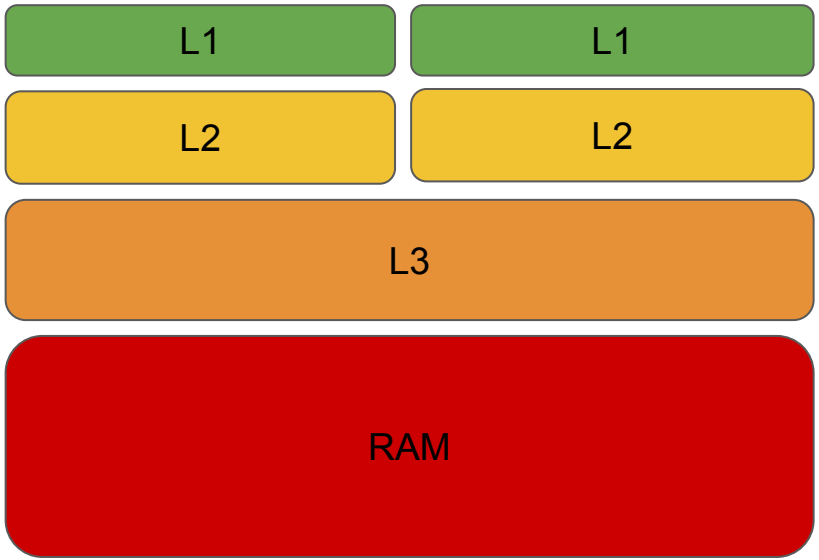
Benchmark
Lookup / Merge & Split / Update / Deleted

Benchmark

Lookup(40%) / Merge & Split(30%) / Update(15%) / Deleted(15%)


Classic
New
5000
4000
3000
2000
1000
0


RAM


L1
L1
L2
L2
L3
RAM


Core 0
Core 1
registers
registers
L1
L1
L2
L2
L3
RAM


Core 0
Core 1
registers
registers
L1
L1
L2
L2
L3
RAM

Выводы

Выводы

- Cache

Выводы

- Cache
- Data locality
- Hot or Cold Data

Выводы

- Cache
- Data locality
- Hot or Cold Data
- SIMD friendly

Выводы

- Cache
- Data locality
- Hot or Cold Data

Ottergon Dev

Спасибо

duckstax.com

github.com/duckstax

Borgardt Alexander

aa.borgardt@yandex.ru

kotbegemot

TG: @k0tb9g9m0t