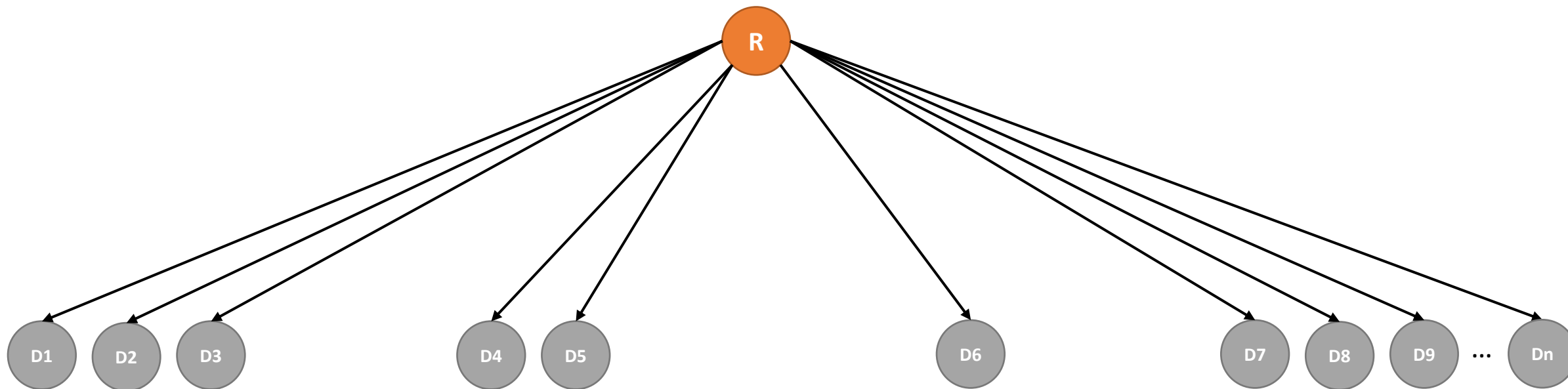


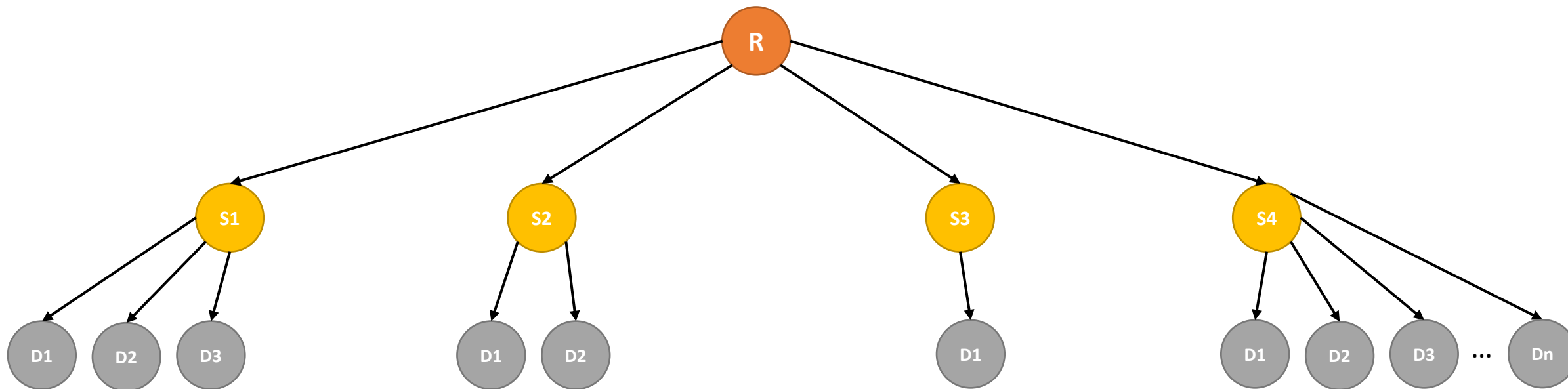


# Логическая организация всех данных. Вариант 1



-  Корень, является точкой входа, содержит список всех документов
-  Документ, содержит данные

## Логическая организация всех данных. Вариант 2

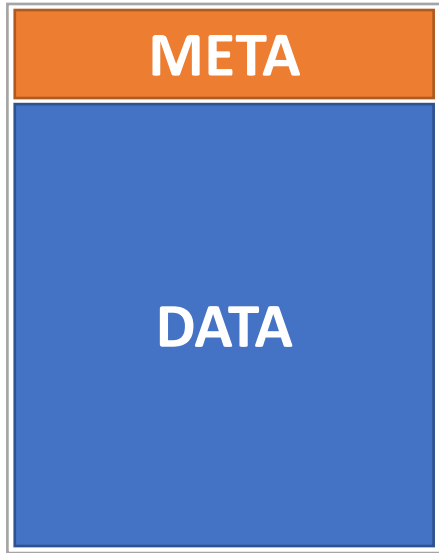


**R** Корень, является точкой входа, содержит список всех коллекций

**S** Коллекция, содержит список всех документов

**D** Документ, содержит данные

# Устройство документа (=блока)



document ::= uint40 uint40 uint72 data "\xFF"

data ::= element data  
| ""

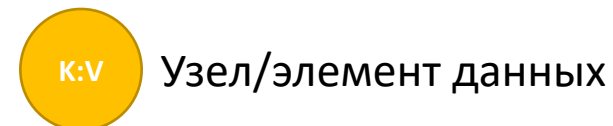
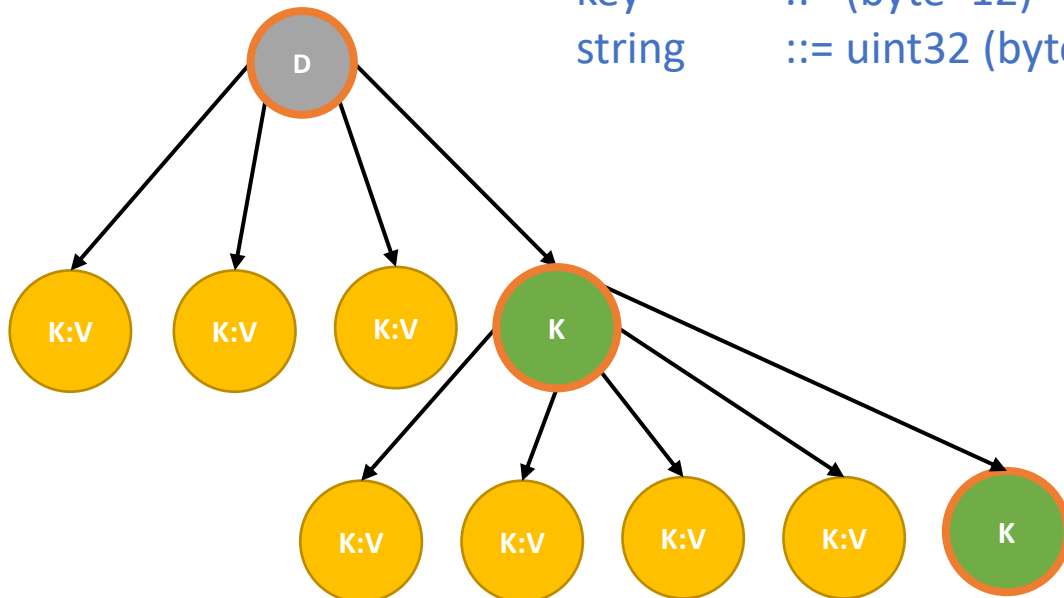
list of data  
empty data

element ::= "\x01" key int  
| "\x02" key double  
| "\x03" key byte  
| "\x04" key string  
| "\x05" key uint40 uint40 uint72 data "\xFE"

32-bit integer  
64-bit float  
Boolean (uint8)  
UTF-8 string  
Embedded document

key ::= (byte\*12) "\x00"  
string ::= uint32 (byte\*) "\x00"

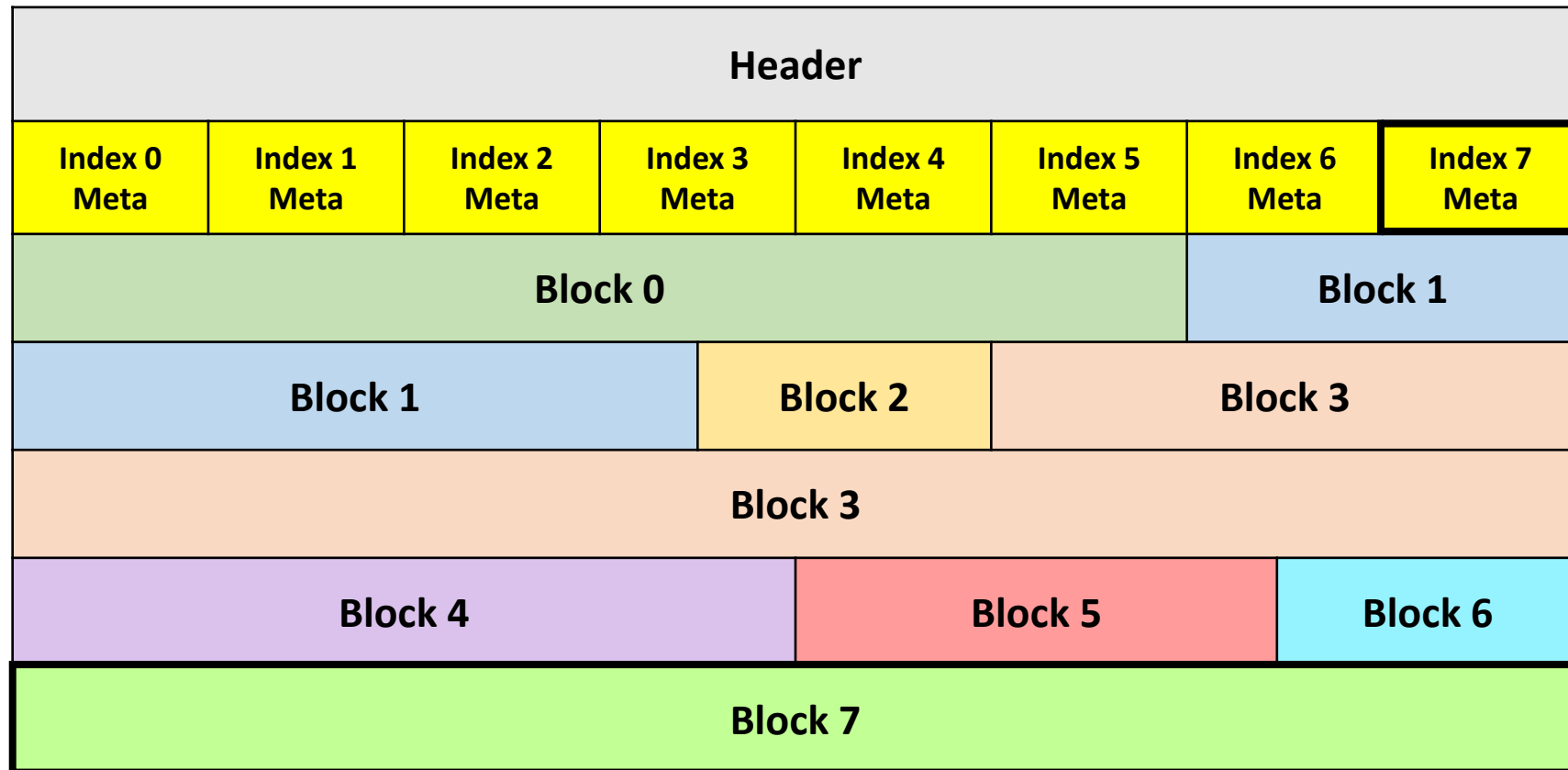
UTF-8 encoded chars with \xFF at the end  
uint32 – length, then UTF-8 chars



**Операция вставки: вариант 1. Места для индексов имеются, пустот нет. Изначальный вид файла.**

Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Block 0						Block 1	
Block 1			Block 2		Block 3		
Block 3							
Block 4				Block 5		Block 6	

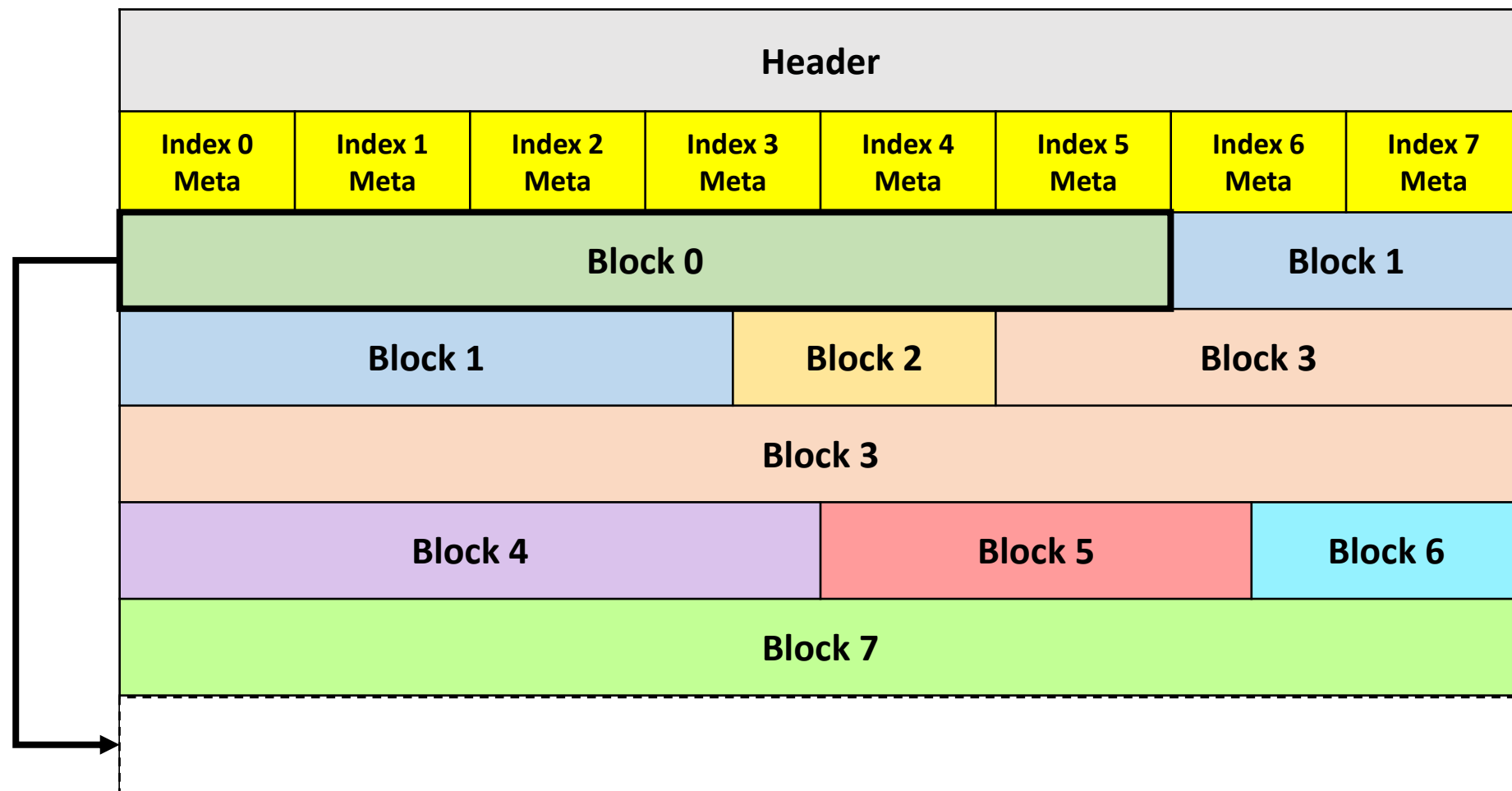
## Операция вставки: вариант 1. Конечный вид файла



**Операция вставки: вариант 2. Места для индексов нет, пустот нет. Изначальный вид файла.**

Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Block 0						Block 1	
Block 1			Block 2		Block 3		
Block 3							
Block 4				Block 5		Block 6	
Block 7							

## Операция вставки: вариант 2. Выделение места под индексы.



# Операция вставки: вариант 2. Конечный вид файла

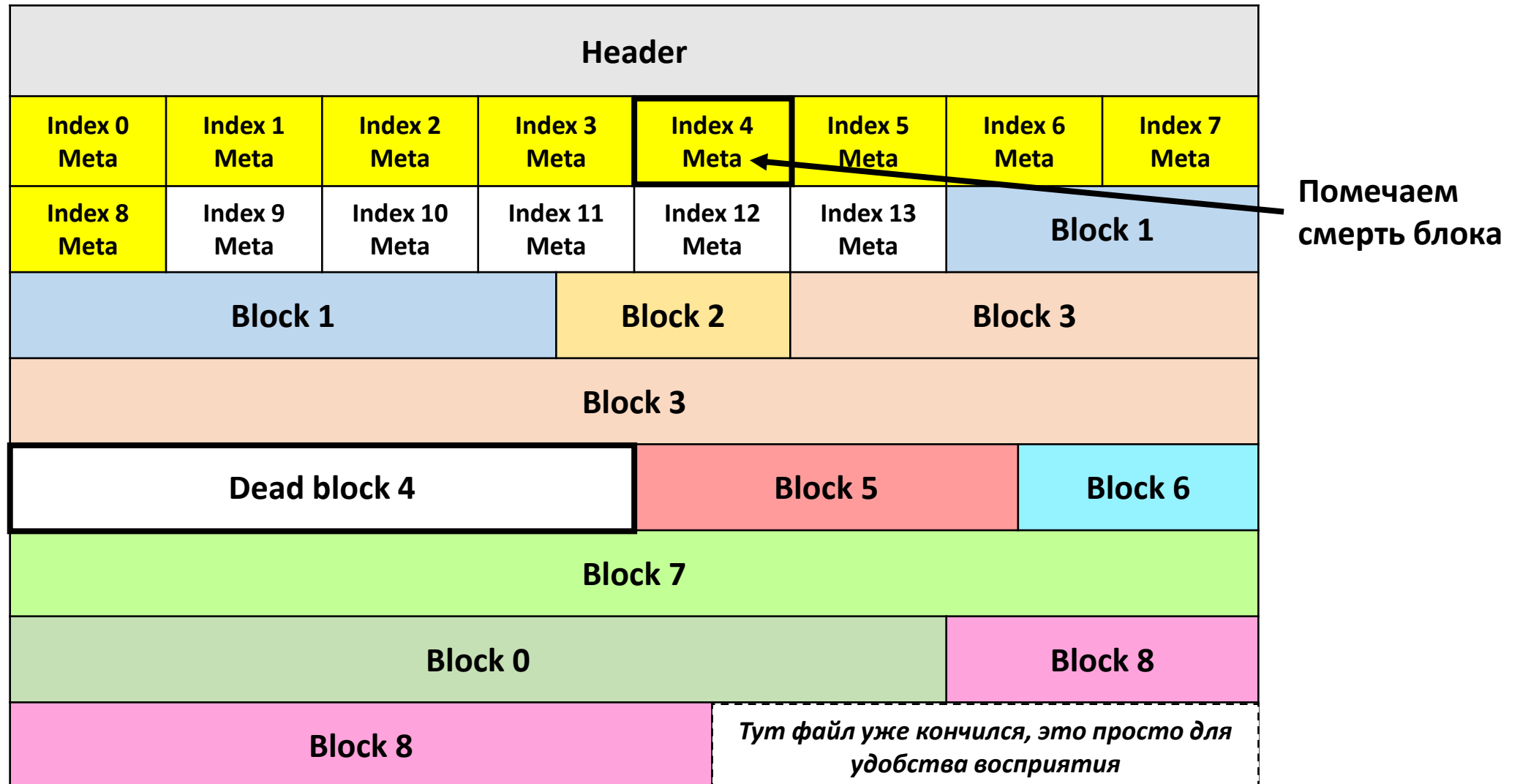
Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Index 8 Meta	Index 9 Meta	Index 10 Meta	Index 11 Meta	Index 12 Meta	Index 13 Meta	Block 1	
Block 1			Block 2		Block 3		
Block 3							
Block 4				Block 5		Block 6	
Block 7							
Block 0						Block 8	
Block 8				Тут файл уже кончился, это просто для удобства восприятия			



## Операция удаления: Изначальный вид файла

Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Index 8 Meta	Index 9 Meta	Index 10 Meta	Index 11 Meta	Index 12 Meta	Index 13 Meta	Block 1	
Block 1			Block 2		Block 3		
Block 3							
Block 4				Block 5		Block 6	
Block 7							
Block 0						Block 8	
Block 8				Тут файл уже кончился, это просто для удобства восприятия			

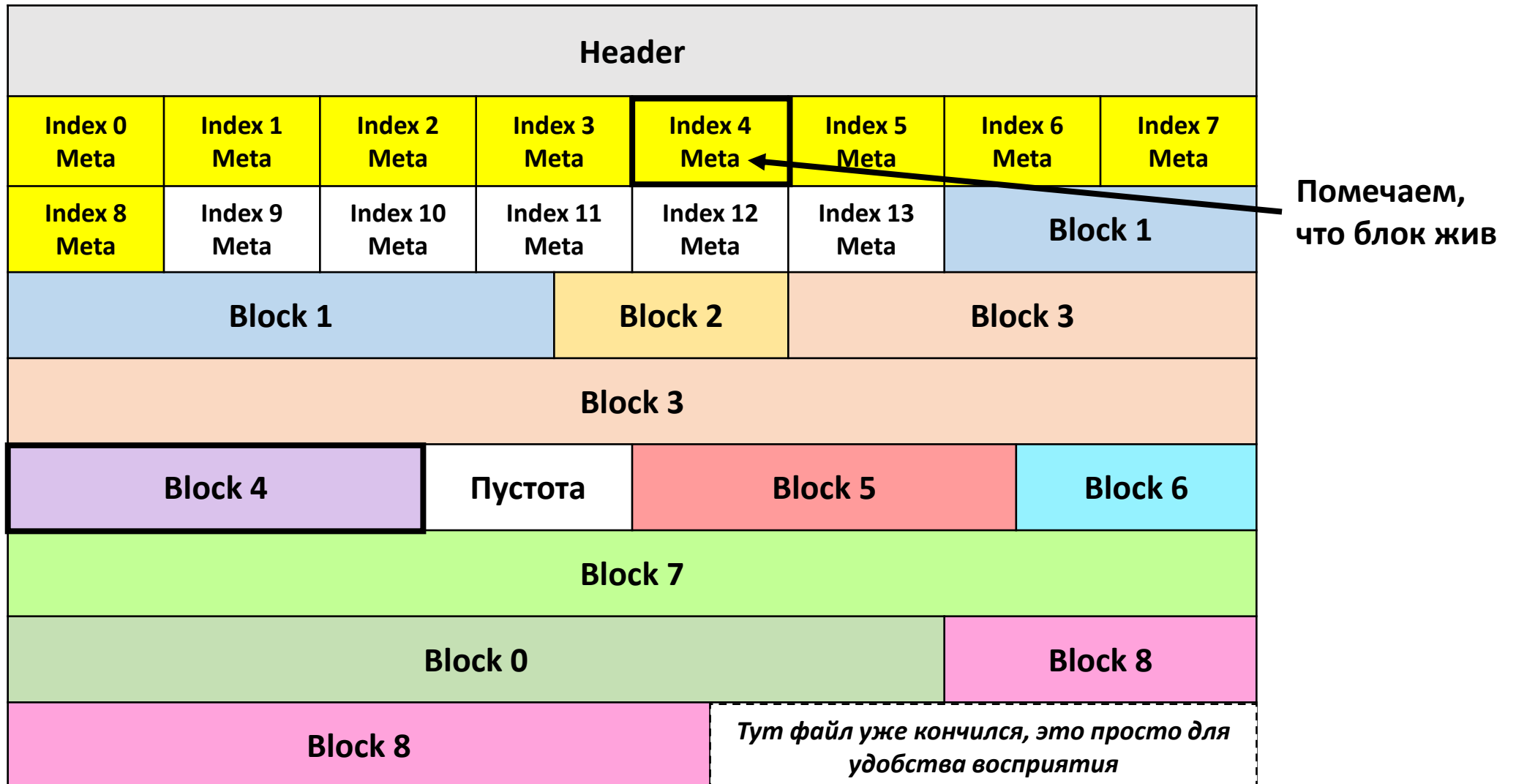
# Операция удаления: Конечный вид файла



## Операция вставки: вариант 3. Место под индексы есть, заполняем пустоту. Изначальный вид файла.

Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Index 8 Meta	Index 9 Meta	Index 10 Meta	Index 11 Meta	Index 12 Meta	Index 13 Meta	Block 1	
Block 1			Block 2		Block 3		
Block 3							
Dead block 4				Block 5		Block 6	
Block 7							
Block 0						Block 8	
Block 8				Тут файл уже кончился, это просто для удобства восприятия			

## Операция вставки: вариант 3. Конечный вид файла.



## Операция вставки: вариант 4. Место под индексы есть, не заполняем пустоту. Изначальный вид файла.

Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Index 8 Meta	Index 9 Meta	Index 10 Meta	Index 11 Meta	Index 12 Meta	Index 13 Meta	Block 1	
Block 1				Block 2		Block 3	
Block 3							
Dead block 4				Block 5		Block 6	
Block 7							
Block 0						Block 8	
Block 8				Тут файл уже кончился, это просто для удобства восприятия			

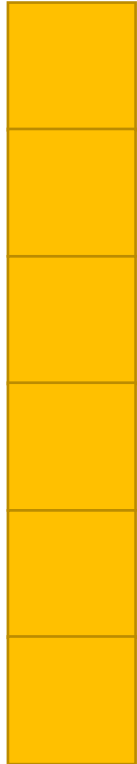
## Операция вставки: вариант 4. Конечный вид файла.

Header							
Index 0 Meta	Index 1 Meta	Index 2 Meta	Index 3 Meta	Index 4 Meta	Index 5 Meta	Index 6 Meta	Index 7 Meta
Index 8 Meta	Index 9 Meta	Index 10 Meta	Index 11 Meta	Index 12 Meta	Index 13 Meta	Block 1	
Block 1				Block 2		Block 3	
Block 3							
Dead block 4				Block 5		Block 6	
Block 7							
Block 0						Block 8	
Block 8				Block 9			

Max Heap (sorted linked list)



Номер индекса и размер блока



← Максимальный свободный блок

← Минимальный свободный блок (в том числе с размером 0)

Вставка 0-ых блоков всегда идёт в конец, вставка обычного блока в худшем случае за  $O(n)$ .  
Куча строиться при загрузке индексов, изменяется при вставке/удалении/обновлении.  
Извлечение макс блока или 0-го за  $O(1)$ .

## ID Generation

4 byte – UNIX timestep  
8 byte – index offset  
  
12-byte unique ID

## ID in hex

24 symbols in HEX

Max block size –  $2^{40}$  байт = 1 Тбайт  
Min block size – 17 байт (пустой)  
Max offset –  $2^{40}$  (можно адресовать 1 Тбайт)  
Index size (in file) – 6 байт  
File header size – in dev (cur: 12 байт)  
Min block size – 21 байт (с одним bool и ключом в байт)

## Index structure

8 byte – offset  
1 byte - Flags

## Index in file

8 byte – offset  
1 byte - Flags

### Flags:

0000 0001 – empty new  
0000 0010 – alive  
0000 0011 – empty dead

## File header

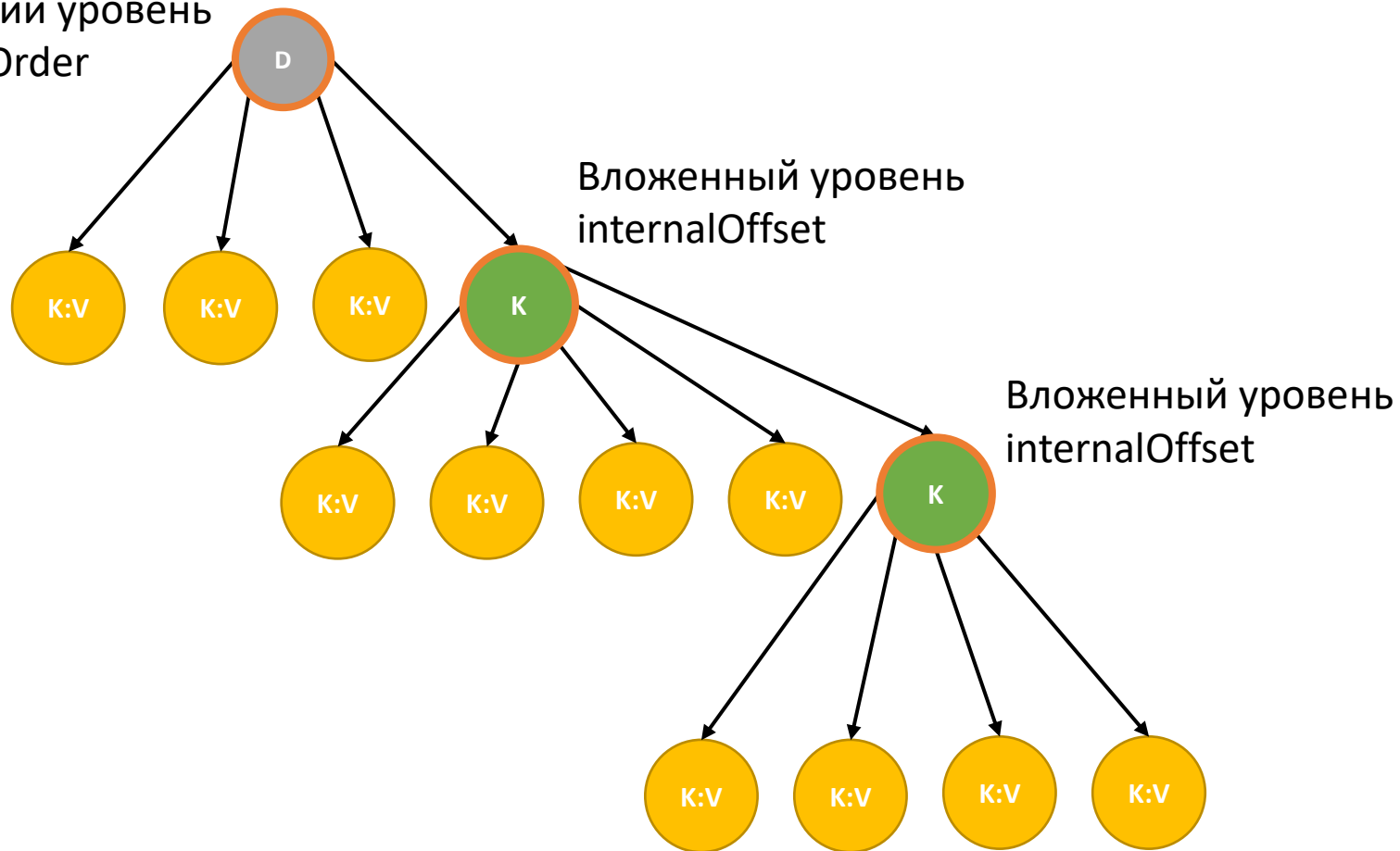
4 bytes - zgdbType  
5 bytes – indexCount  
8 bytes – freeListOffset

## Doc header

5 bytes – size;  
5 bytes – indexOrder or internalOffset;  
12 bytes – id



Верхний уровень  
indexOrder



При увеличении размера строки в документе, необходимо обновить offset дочерних документов того же уровня, что и строка. Для этого необходимо пройти по всем ключам и сохранить позиции вложенных документов