Методы и системы обработки больших данных

5. Система хранения и индексные файлы

Система хранения данных

Схема базы данных (метаданные)

- Схема базы данных – совокупность схем отношений
- Схема базы данных хранится на диске в *словаре базы данных*

Схема отношения S (Поставщики)

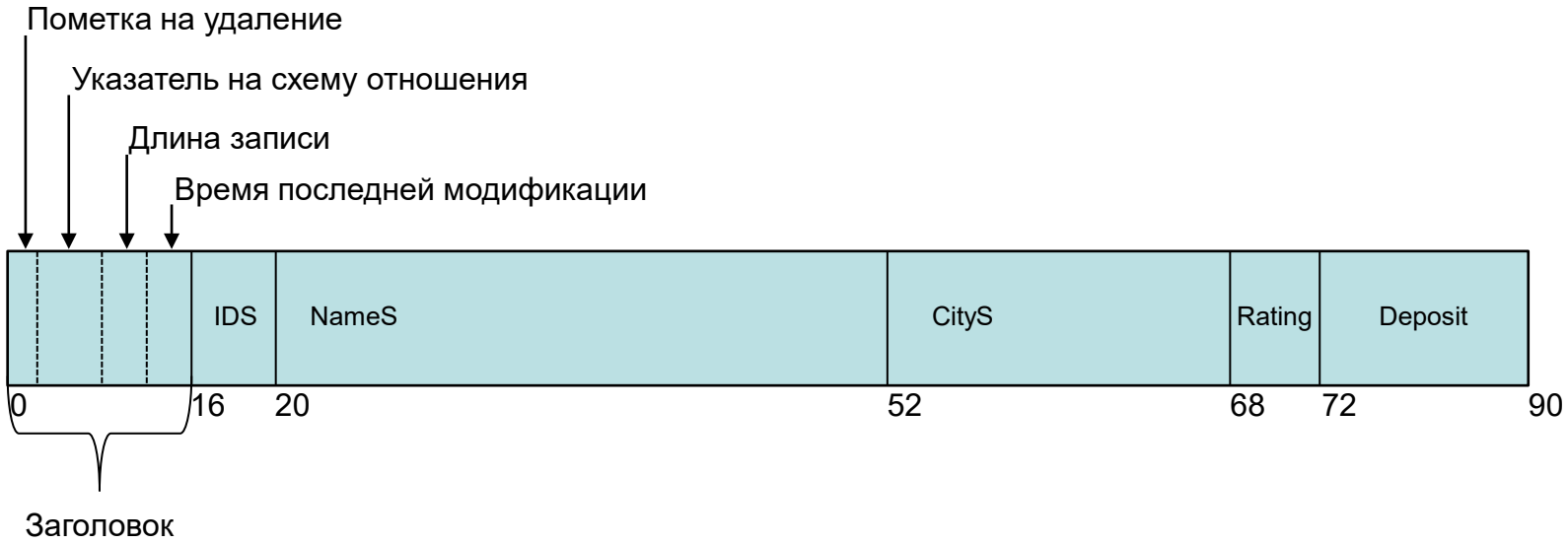
Атрибуты S (Поставщики)

Идентификатор	Тип	Длина (байт)	Ограничения	Порядковый номер
CityS	Char	16		3
Deposit	Float	8		5
IDS	Integer	4	Первич. ключ	1
NameS	Char	32		2
Rating	Integer	4		4

Поля, записи и блоки

- *Поле* – это последовательность байт, представляющих атрибут
- *Запись* – последовательность полей, представляющих кортеж
- *Блок (дисковая страница)* – последовательность байт фиксированной длины, содержащая определенное число записей (кортежей) одного отношения
- Отношение представляется связным списком нескольких блоков

Структура записи, представляющей кортеж из S



Заголовок содержит следующую служебную информацию:

- *Пометка на удаление*: 0 – кортеж не удален; 1 – кортеж удален
- *Указатель на схему отношения*: указатель на место хранения схемы отношения S
- *Длина*: длина записи в байтах
- *Время последней модификации*: время последнего изменения атрибутов кортежа

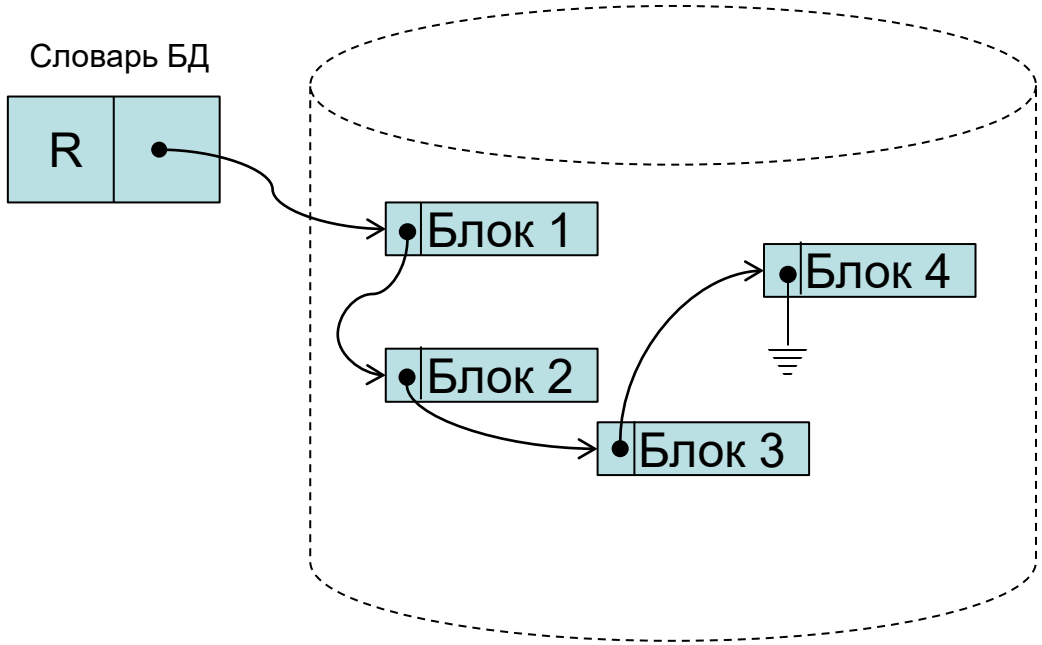
Блок, содержащий записи

- Записи представляют кортежи отношения, хранящиеся в *блоках* (страницах) на диске
- Все блоки имеют фиксированную длину (обычно 4-64 Кбайт)
- Если необходимо считать или обновить одну запись, то в буфер оперативной памяти помещается весь блок, содержащий эту запись; в буфере выполняется операция; после этого в определенный момент блок записывается обратно на диск
- Одно отношение может занимать несколько блоков

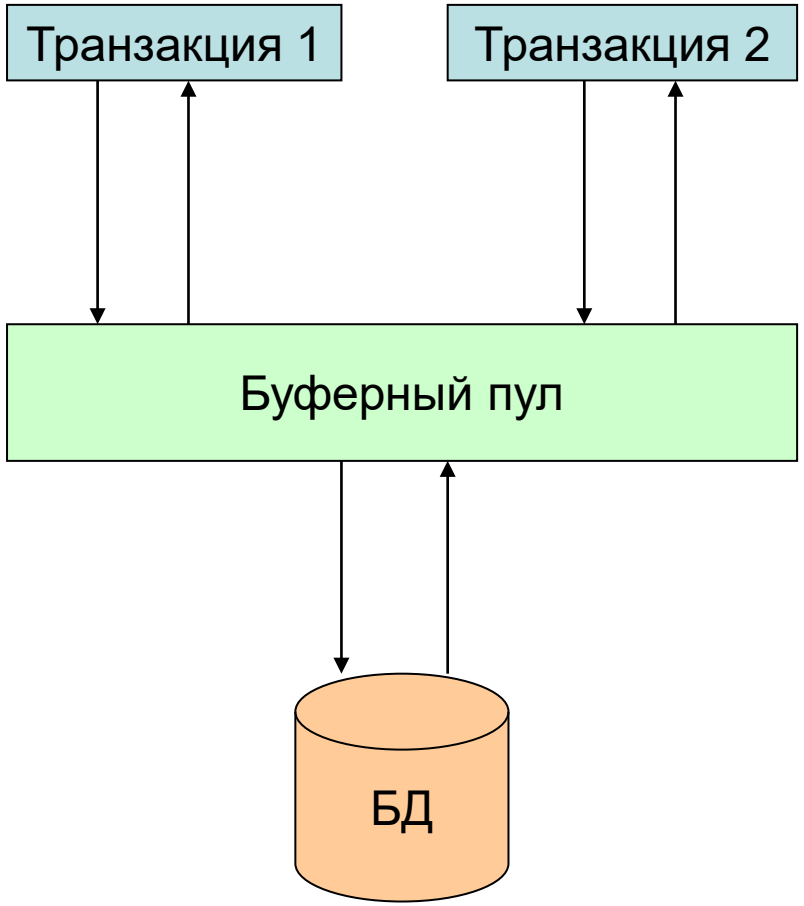
Содержимое заголовка блока

- Указатель на следующий блок (если отношение занимает несколько блоков)
- Информация о том, какому отношению принадлежат кортежи, хранящиеся в данном блоке
- Время последней модификации блока

Связный список блоков, представляющих отношение R



Использование буферного пула



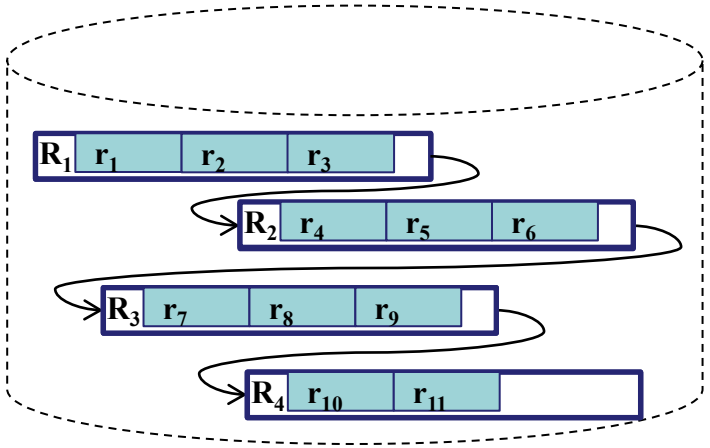
Организация доступа к данным

Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

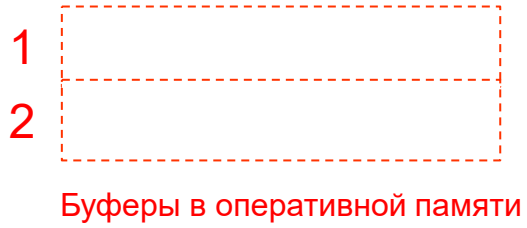
Буферы в оперативной памяти

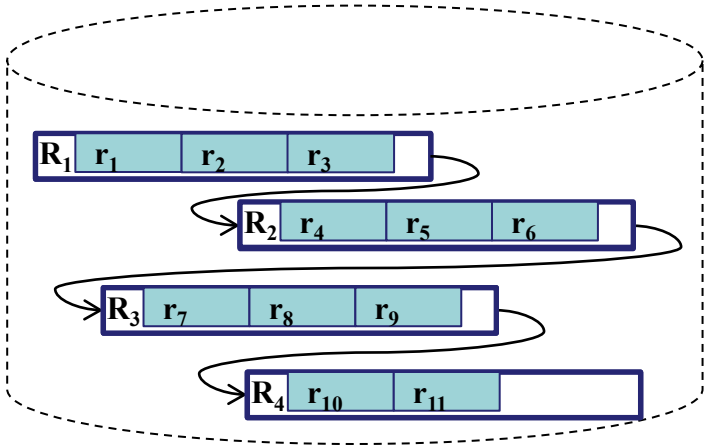


Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```



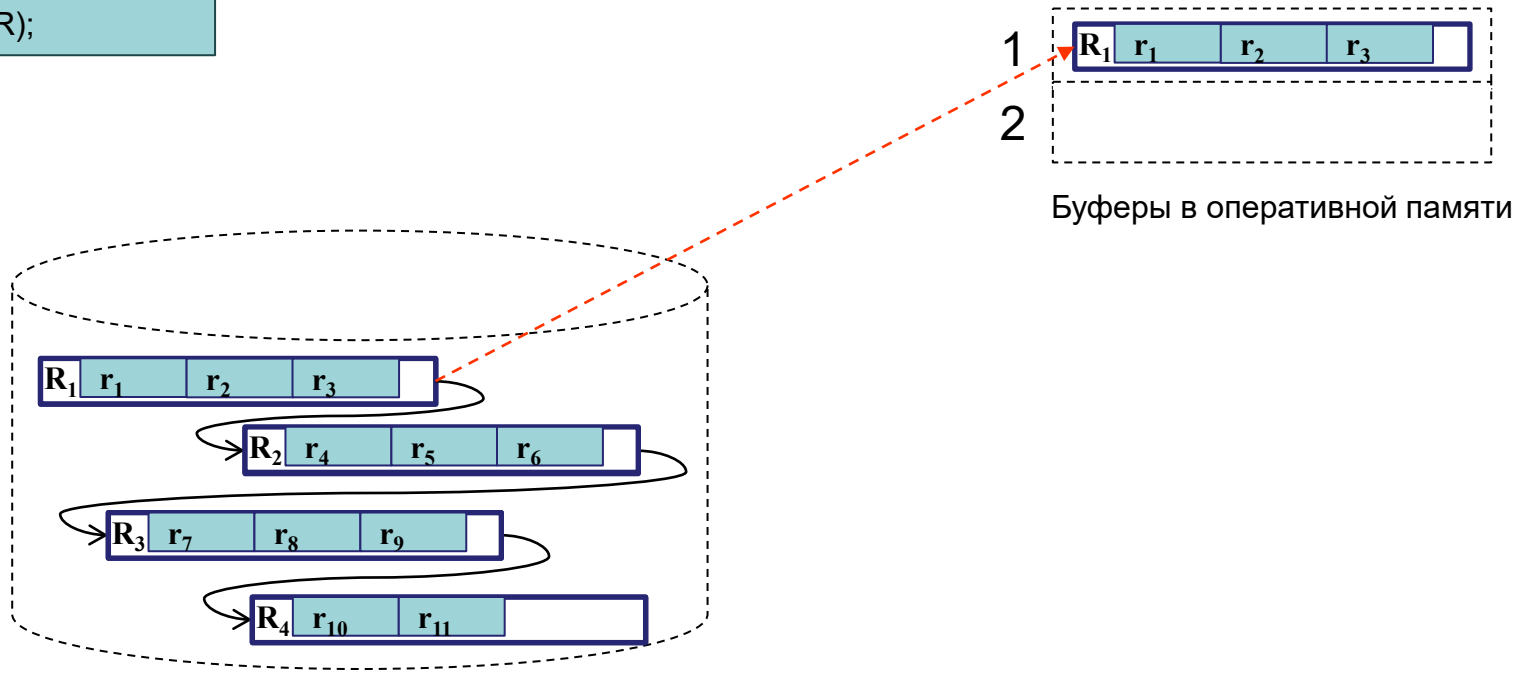


Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) ----► DM.ReadBlock(R_1,1)



Буферы в оперативной памяти

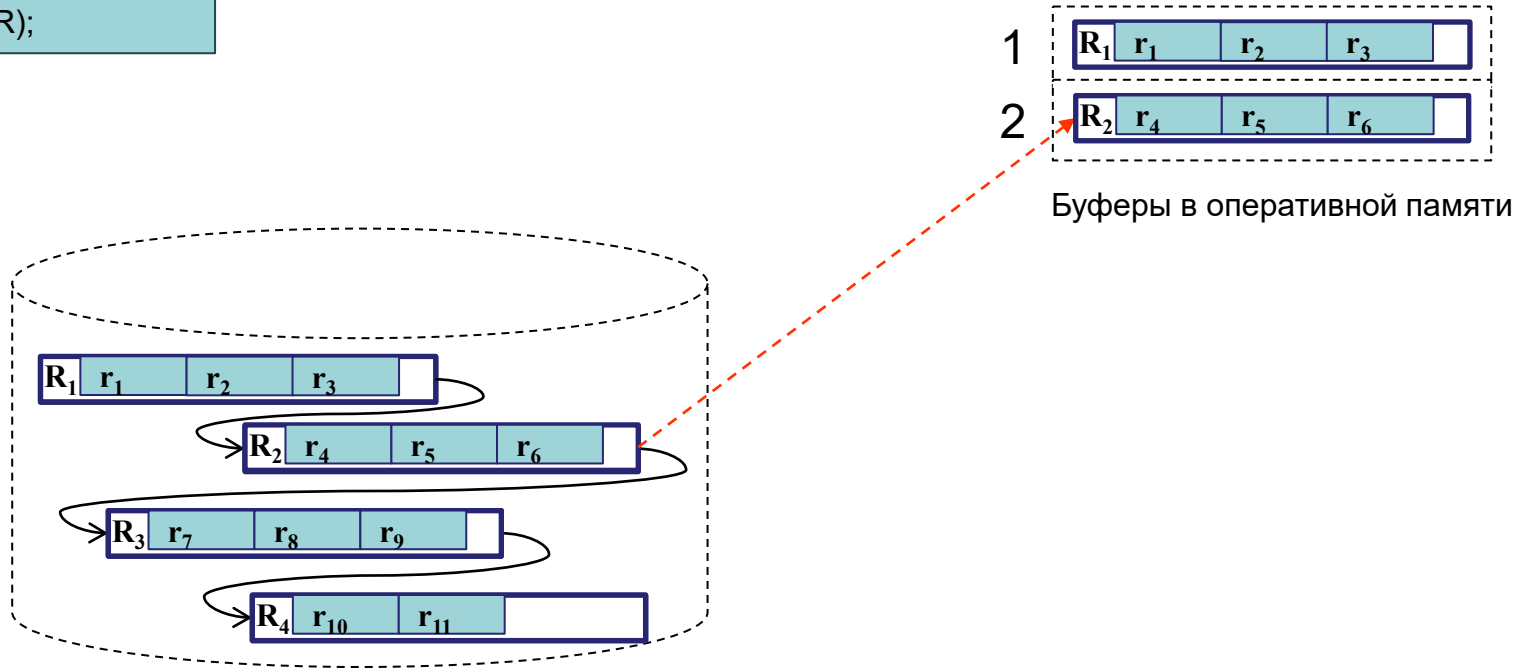
Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)



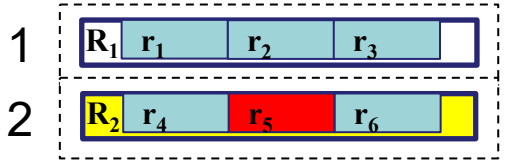
Обработка отношения R

Исполнитель запросов

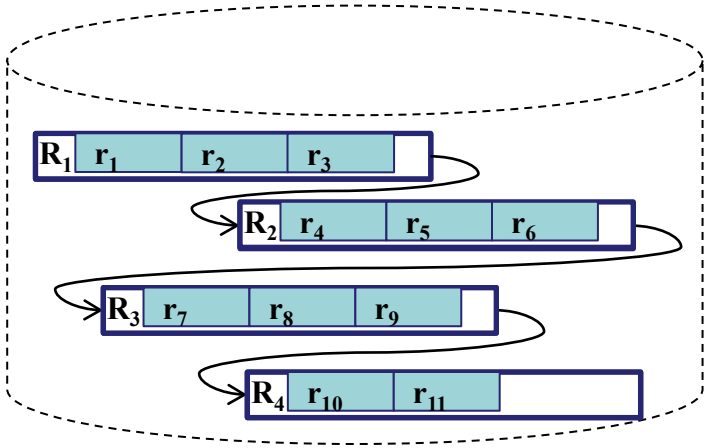
```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) ----► DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----► DM.ReadBlock(R_2,2)



Буферы в оперативной памяти



Обработка отношения R

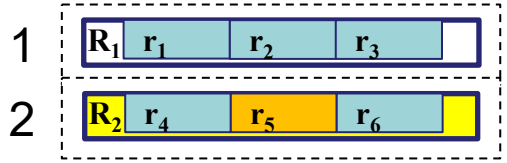
Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

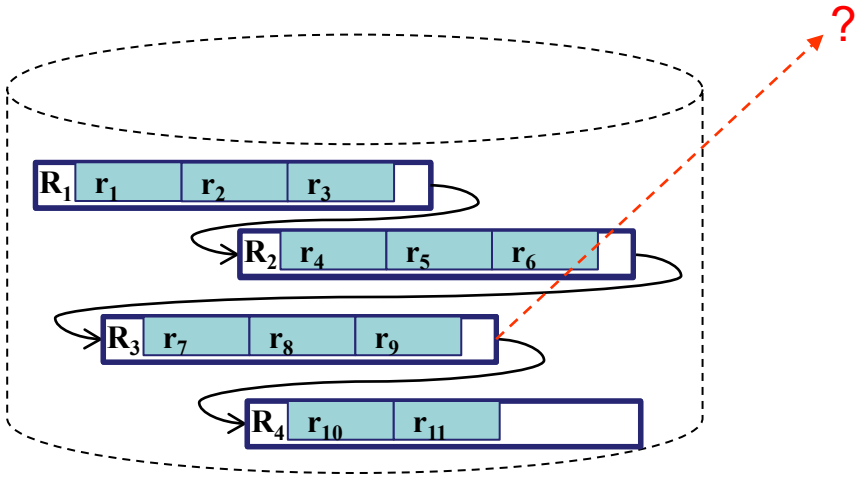
BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----▸ DM.ReadBlock(R_3,1)



Буферы в оперативной памяти



Обработка отношения R

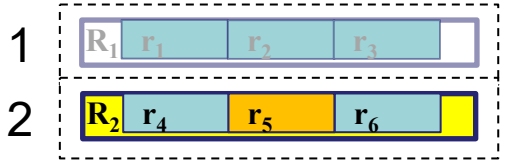
Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

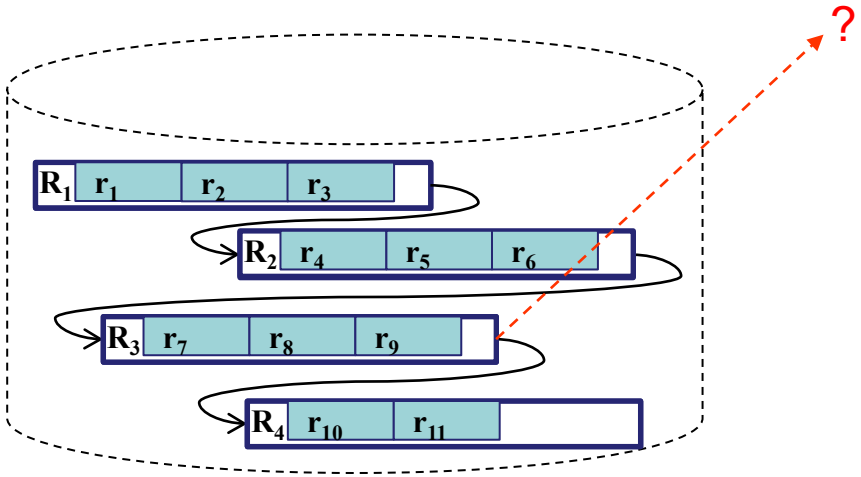
BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----▸ DM.ReadBlock(R_3,1)



Буферы в оперативной памяти



Обработка отношения R

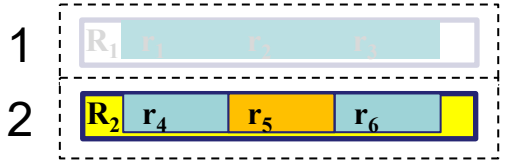
Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

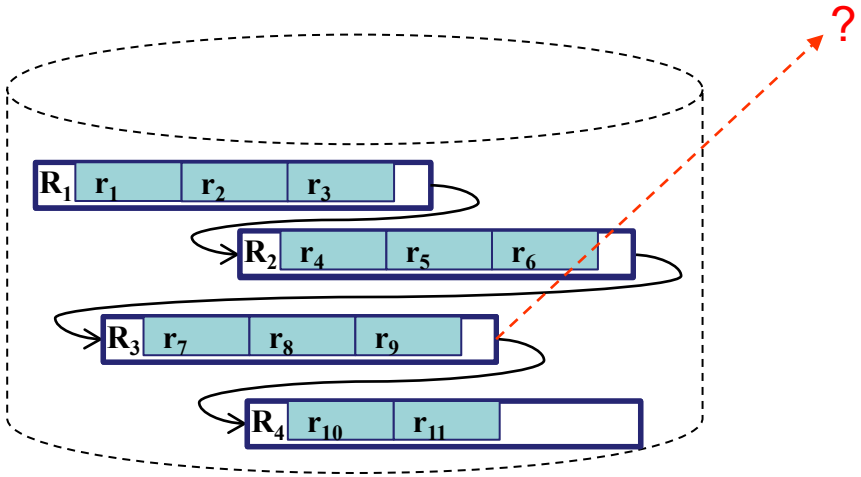
BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----▸ DM.ReadBlock(R_3,1)



Буферы в оперативной памяти



Обработка отношения R

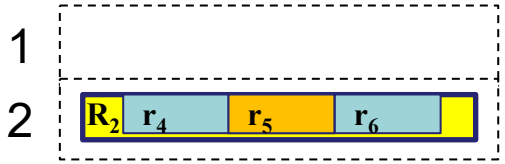
Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

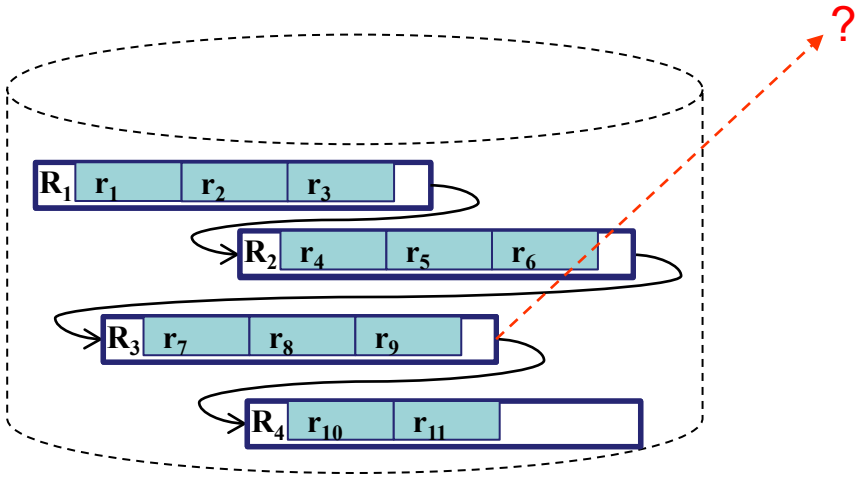
BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----▸ DM.ReadBlock(R_3,1)



Буферы в оперативной памяти



Обработка отношения R

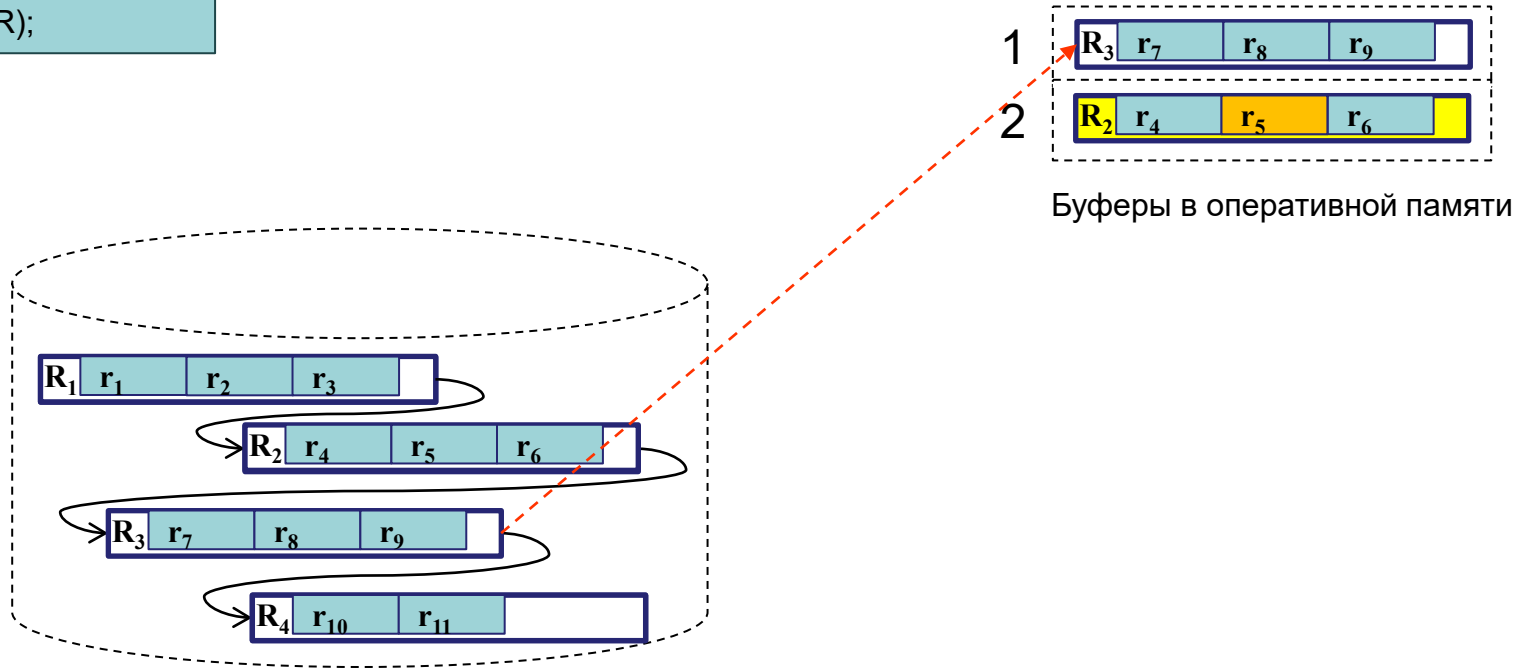
Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----▸ DM.ReadBlock(R_3,1)



Буферы в оперативной памяти

Обработка отношения R

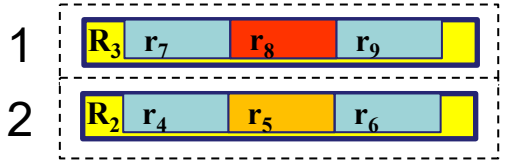
Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

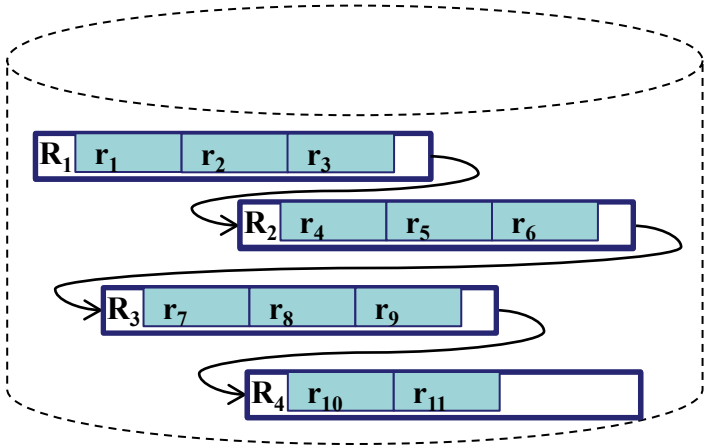
BM.GetBlock(R_1) ----► DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----► DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----► DM.ReadBlock(R_3,1)



Буферы в оперативной памяти



Обработка отношения R

Исполнитель запросов

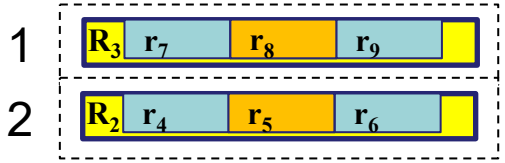
```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) ----▸ DM.ReadBlock(R_1,1)

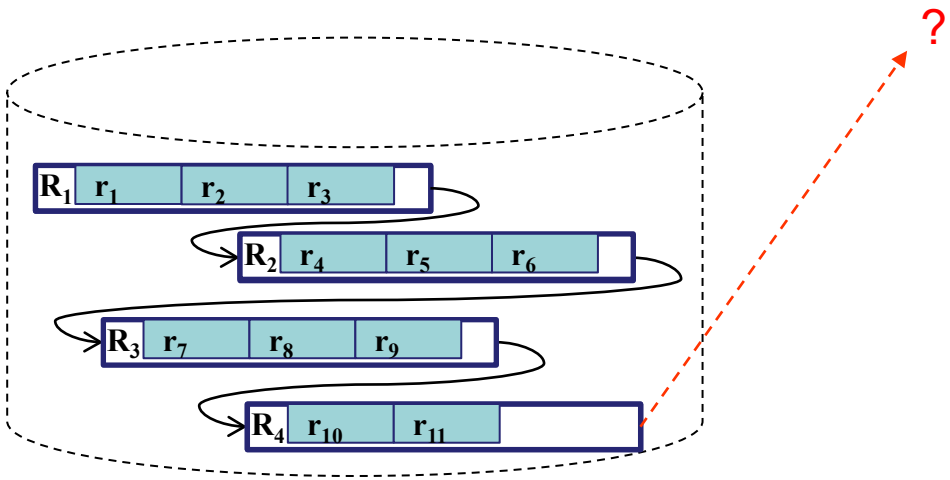
BM.GetBlock(R_2) ----▸ DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----▸ DM.ReadBlock(R_3,1)

BM.GetBlock(R_4)



Буферы в оперативной памяти



Обработка отношения R

Исполнитель запросов

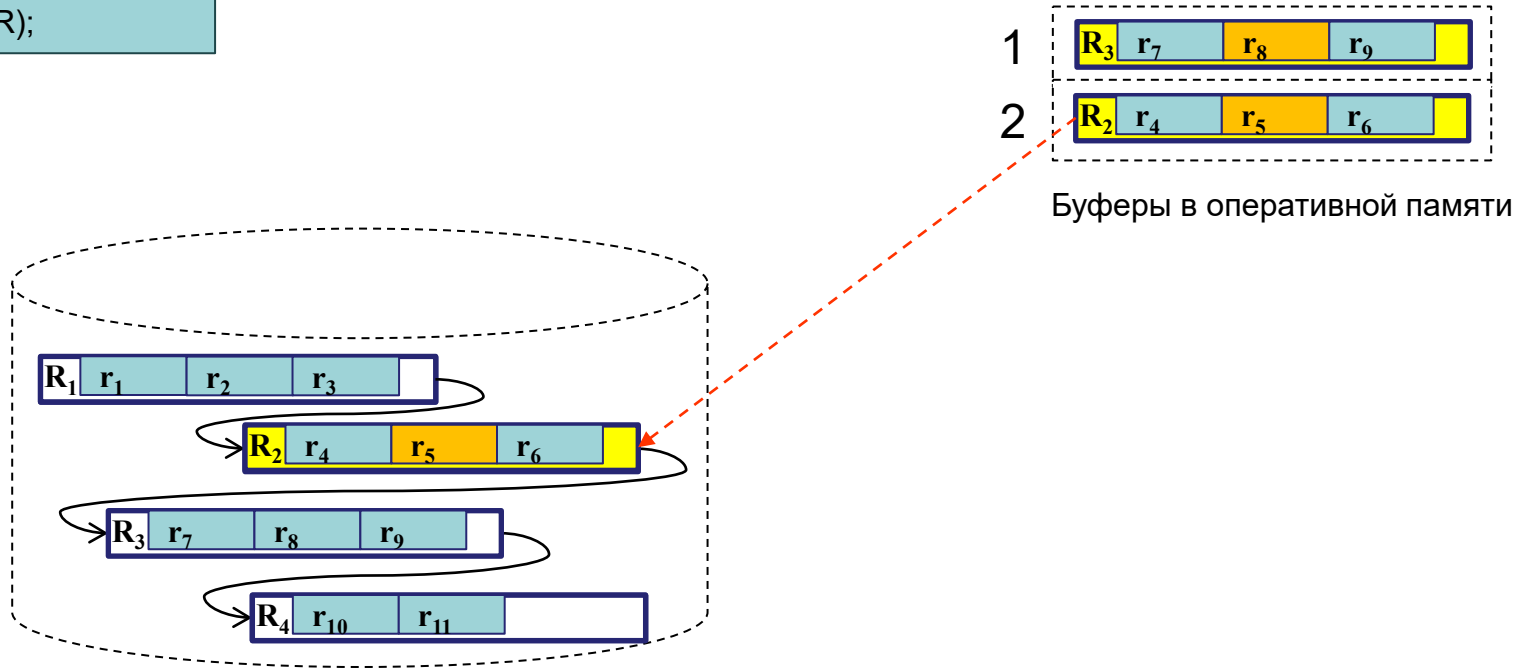
```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) ----► DM.ReadBlock(R_1,1)

BM.GetBlock(R_2) ----► DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) ----► DM.ReadBlock(R_3,1)

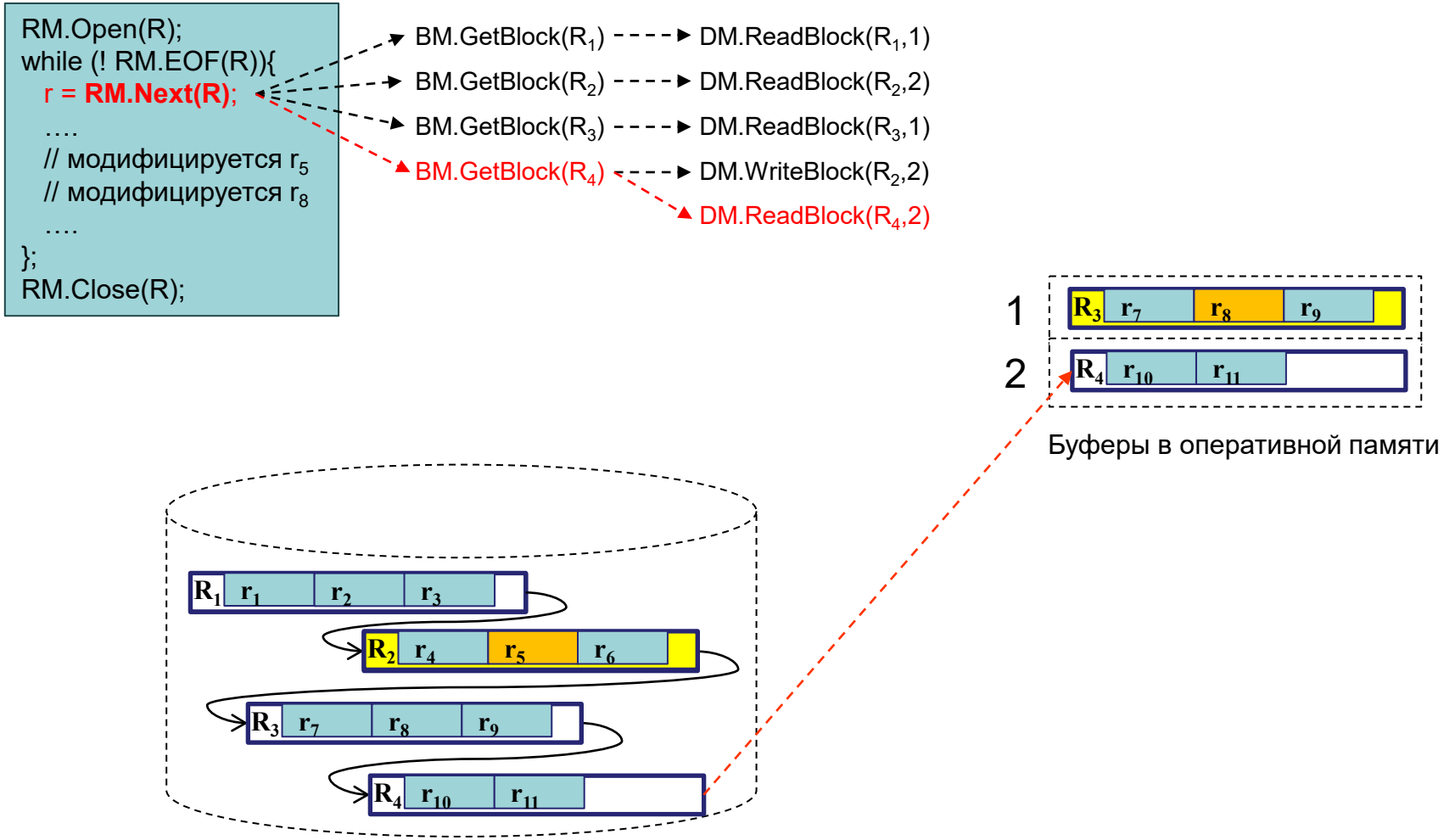
BM.GetBlock(R_4) ----► DM.WriteBlock(R_2,2)



Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```



Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

BM.GetBlock(R_1) → DM.ReadBlock(R_1,1)

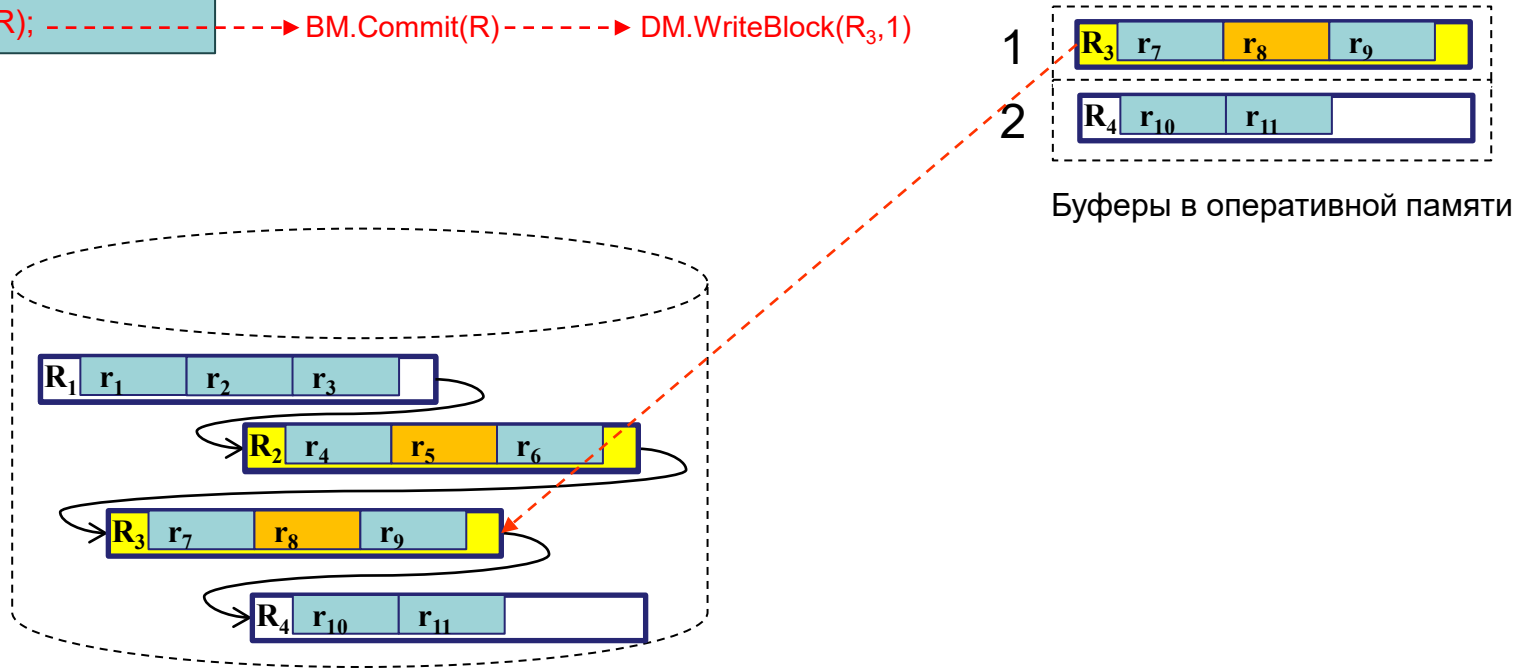
BM.GetBlock(R_2) → DM.ReadBlock(R_2,2)

BM.GetBlock(R_3) → DM.ReadBlock(R_3,1)

BM.GetBlock(R_4) → DM.WriteBlock(R_2,2)

DM.ReadBlock(R_4,2)

RM.Close(R); → BM.Commit(R) → DM.WriteBlock(R_3,1)



Буферы в оперативной памяти

Обработка отношения R

Исполнитель запросов

```
RM.Open(R);
while (! RM.EOF(R)){
   r = RM.Next(R);
   ….
   // модифицируется r5
   // модифицируется r8
   ….
};
RM.Close(R);
```

$BM.GetBlock(R_1) \rightarrow DM.ReadBlock(R_1,1)$

$BM.GetBlock(R_2) \rightarrow DM.ReadBlock(R_2,2)$

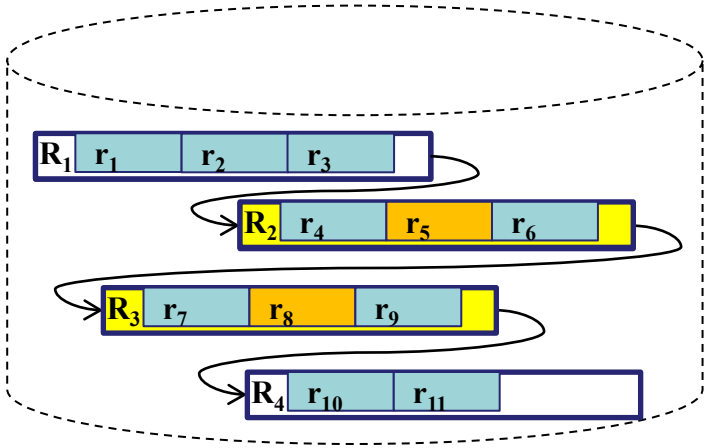
$BM.GetBlock(R_3) \rightarrow DM.ReadBlock(R_3,1)$

$BM.GetBlock(R_4) \rightarrow DM.WriteBlock(R_2,2)$

$DM.ReadBlock(R_4,2)$

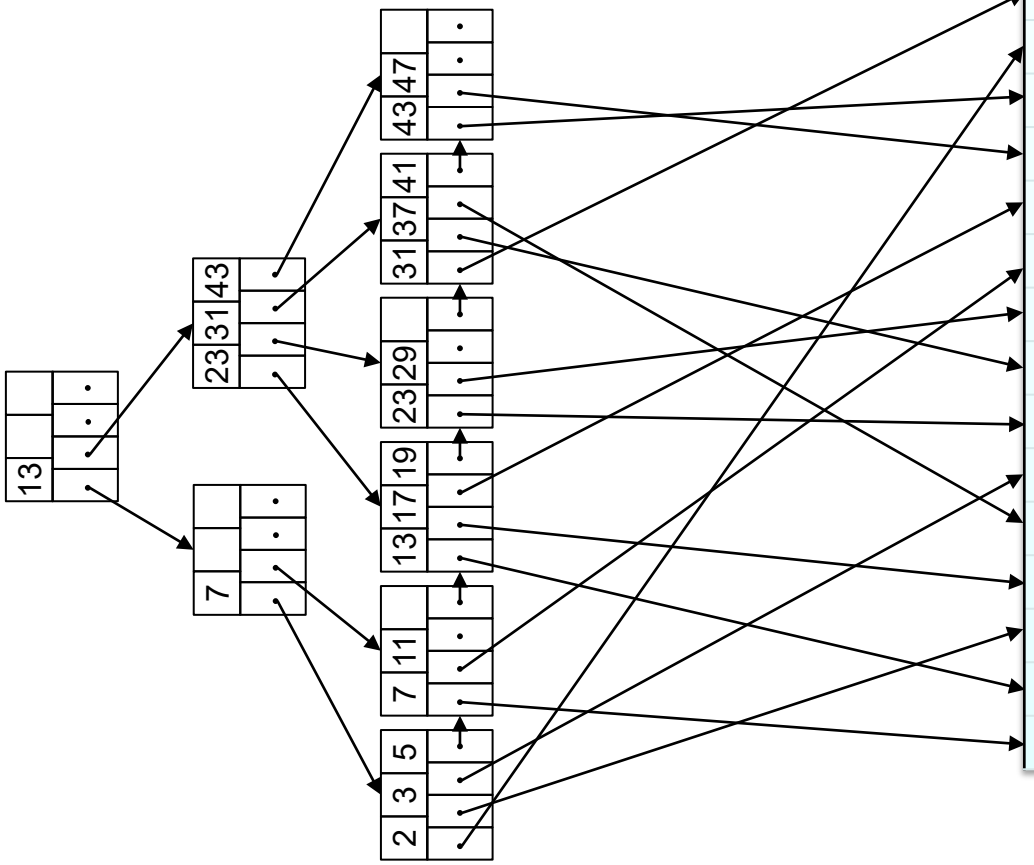
$BM.Commit(R) \rightarrow DM.WriteBlock(R_3,1)$

Буферы в оперативной памяти



Индексные файлы

Пример индекса

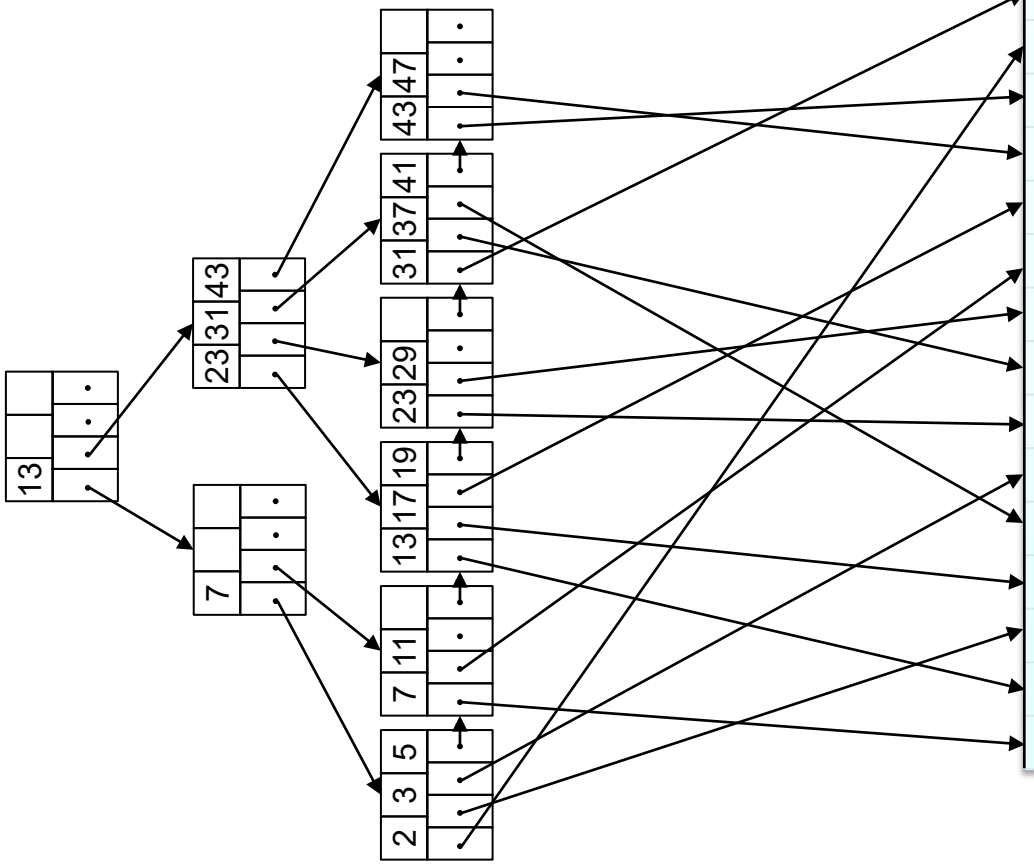


S (поставщики)

IDS	NameS	CityS	Rating	Deposit
31	Петров	Москва	100	1000
2	Иванов Ф	Воронеж	60	2000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000
19	Крюков	Воронеж	100	1000
11	Степанов	Элиста	75	2000
29	Горин	Уфа	85	1000
37	Коваль	Минск	60	2000
23	Федоров	Москва	80	1000
5	Ненашев	Керч	95	1000
41	Глинин	Минск	80	1000
17	Марьин	Тула	60	2000
3	Демин	Москва	50	5000
13	Ильин	Минск	80	1000
7	Иванов К	Москва	100	1000

Пример поиска по индексу

Найти поставщика с IDS 23

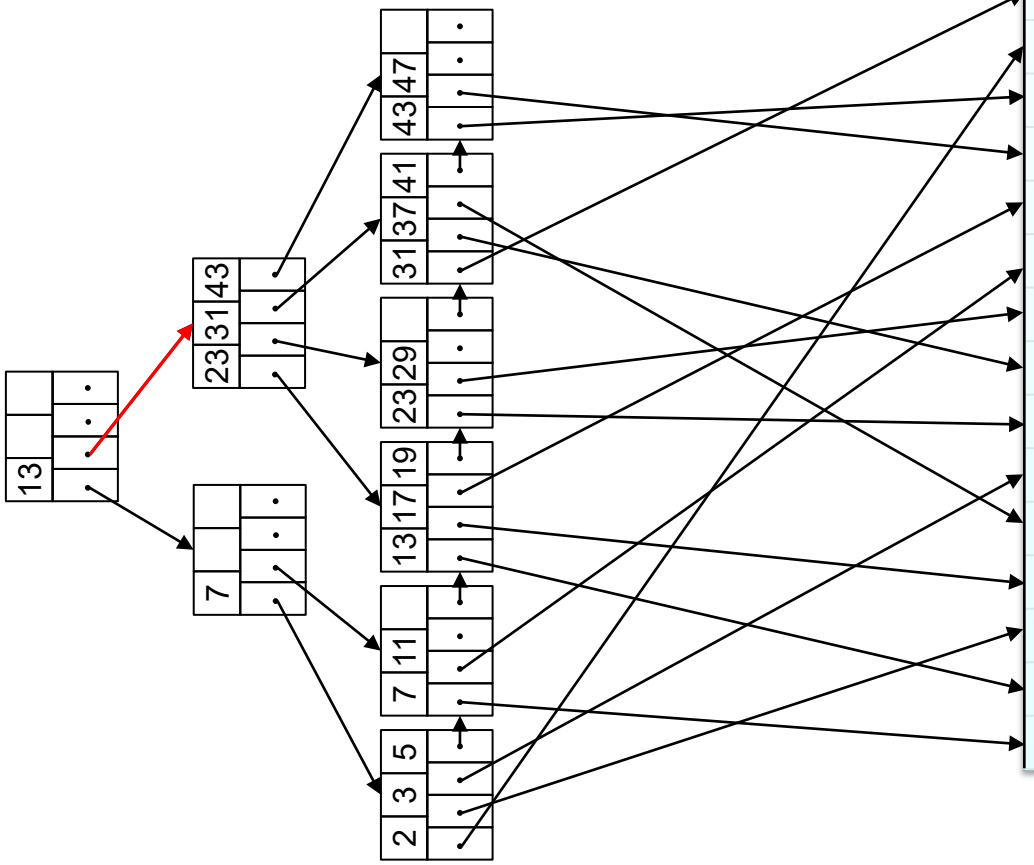


S (поставщики)

IDS	NameS	CityS	Rating	Deposit
31	Петров	Москва	100	1000
2	Иванов Ф	Воронеж	60	2000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000
19	Крюков	Воронеж	100	1000
11	Степанов	Элиста	75	2000
29	Горин	Уфа	85	1000
37	Коваль	Минск	60	2000
23	Федоров	Москва	80	1000
5	Ненашев	Керч	95	1000
41	Глинин	Минск	80	1000
17	Марьин	Тула	60	2000
3	Демин	Москва	50	5000
13	Ильин	Минск	80	1000
7	Иванов К	Москва	100	1000

Пример поиска по индексу

Найти поставщика с IDS 23

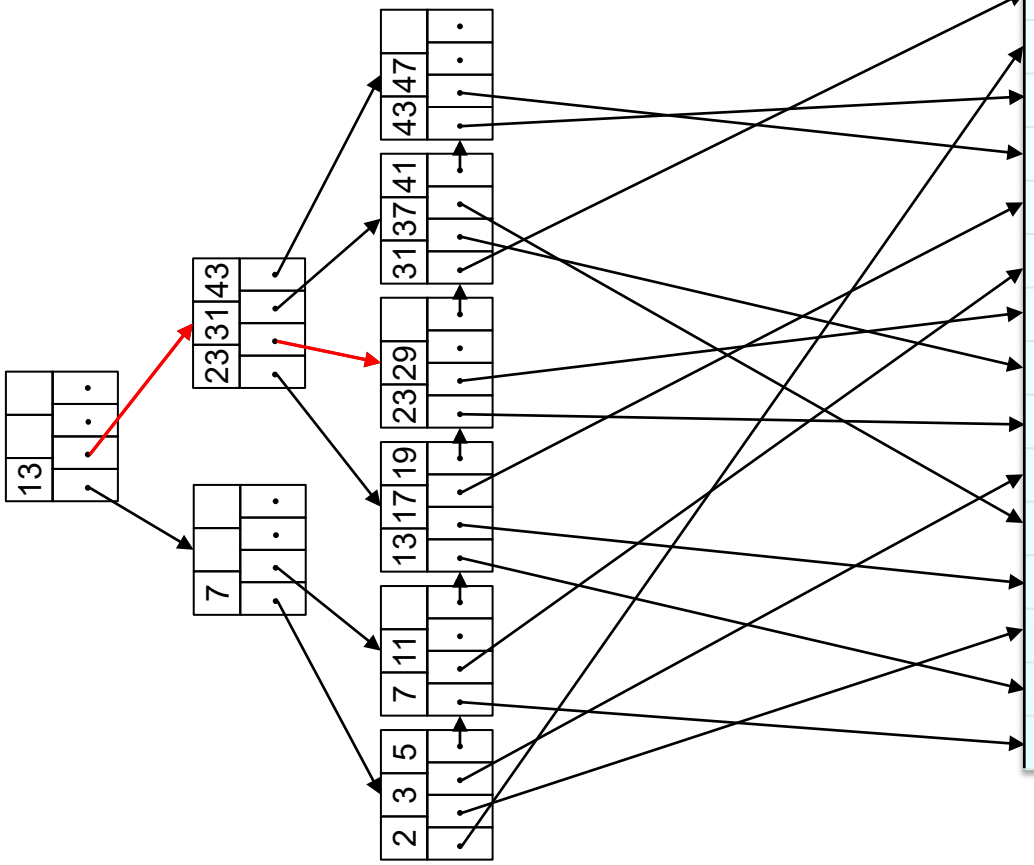


S (поставщики)

IDS	NameS	CityS	Rating	Deposit
31	Петров	Москва	100	1000
2	Иванов Ф	Воронеж	60	2000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000
19	Крюков	Воронеж	100	1000
11	Степанов	Элиста	75	2000
29	Горин	Уфа	85	1000
37	Коваль	Минск	60	2000
23	Федоров	Москва	80	1000
5	Ненашев	Керч	95	1000
41	Глинин	Минск	80	1000
17	Марьин	Тула	60	2000
3	Демин	Москва	50	5000
13	Ильин	Минск	80	1000
7	Иванов К	Москва	100	1000

Пример поиска по индексу

Найти поставщика с IDS 23

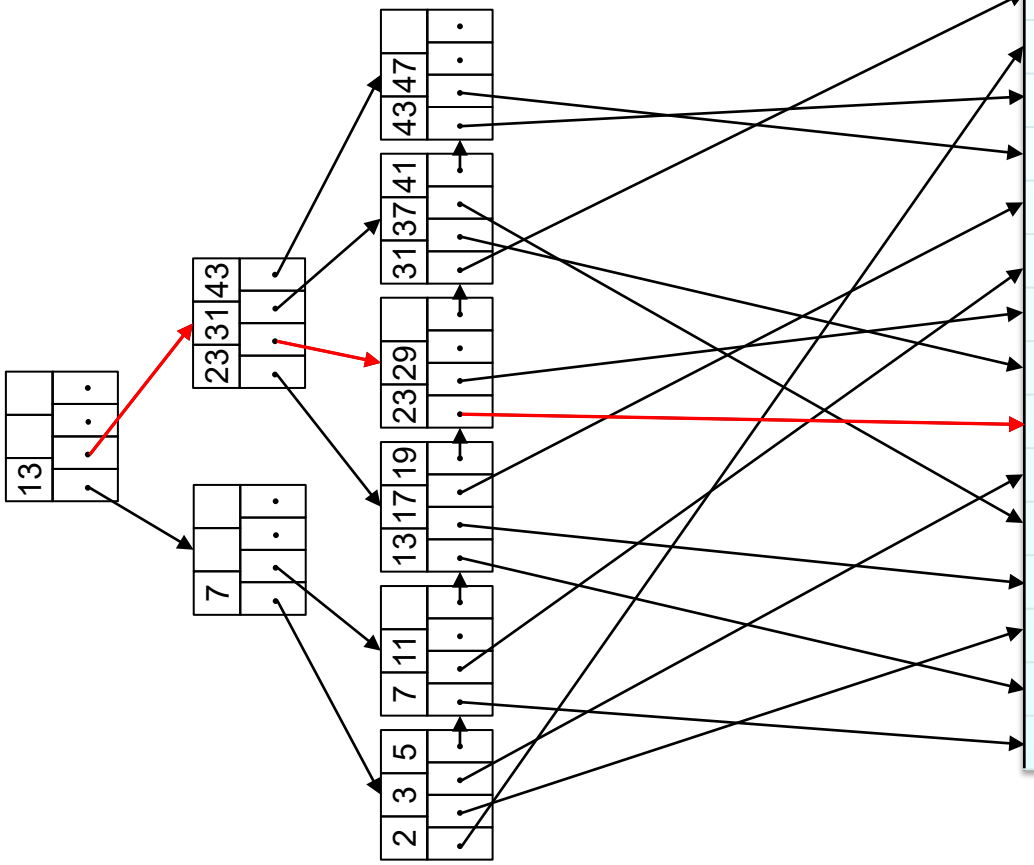


S (поставщики)

IDS	NameS	CityS	Rating	Deposit
31	Петров	Москва	100	1000
2	Иванов Ф	Воронеж	60	2000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000
19	Крюков	Воронеж	100	1000
11	Степанов	Элиста	75	2000
29	Горин	Уфа	85	1000
37	Коваль	Минск	60	2000
23	Федоров	Москва	80	1000
5	Ненашев	Керч	95	1000
41	Глинин	Минск	80	1000
17	Марьин	Тула	60	2000
3	Демин	Москва	50	5000
13	Ильин	Минск	80	1000
7	Иванов К	Москва	100	1000

Пример поиска по индексу

Найти поставщика с IDS 23

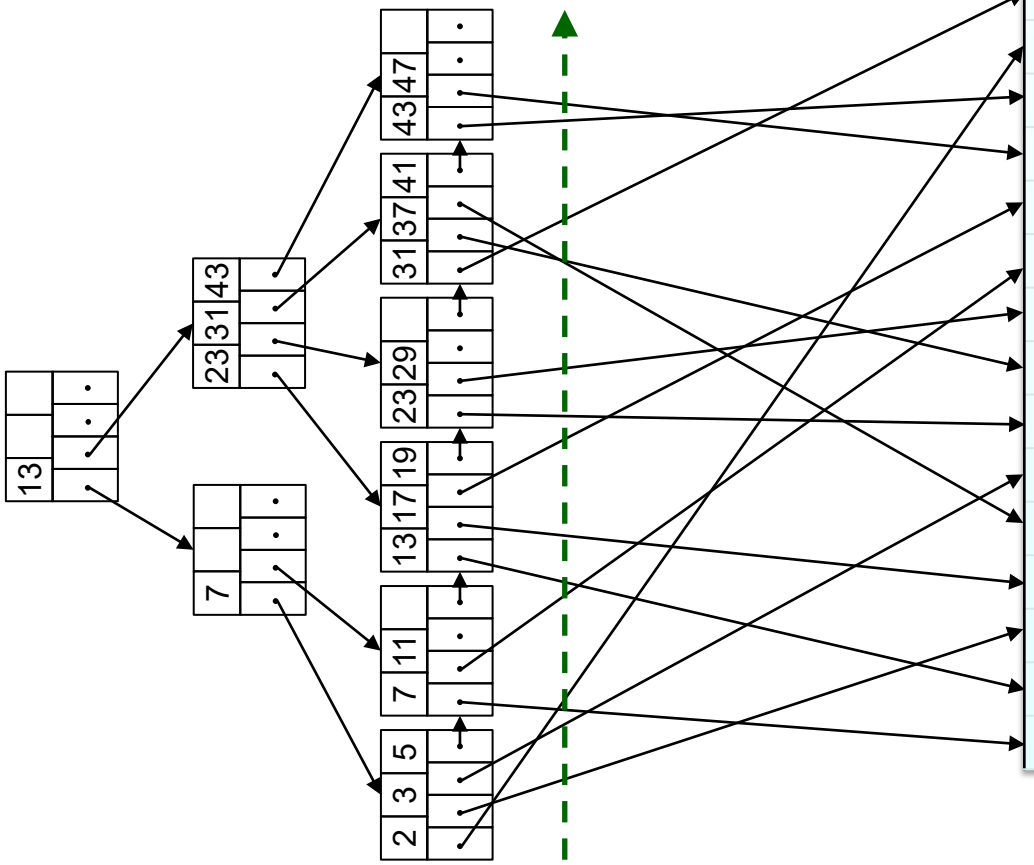


S (поставщики)

IDS	NameS	CityS	Rating	Deposit
31	Петров	Москва	100	1000
2	Иванов Ф	Воронеж	60	2000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000
19	Крюков	Воронеж	100	1000
11	Степанов	Элиста	75	2000
29	Горин	Уфа	85	1000
37	Коваль	Минск	60	2000
23	Федоров	Москва	80	1000
5	Ненашев	Керч	95	1000
41	Глинин	Минск	80	1000
17	Марьин	Тула	60	2000
3	Демин	Москва	50	5000
13	Ильин	Минск	80	1000
7	Иванов К	Москва	100	1000

Пример упорядочивания по индексу

Упорядочить поставщиков по возрастанию IDS

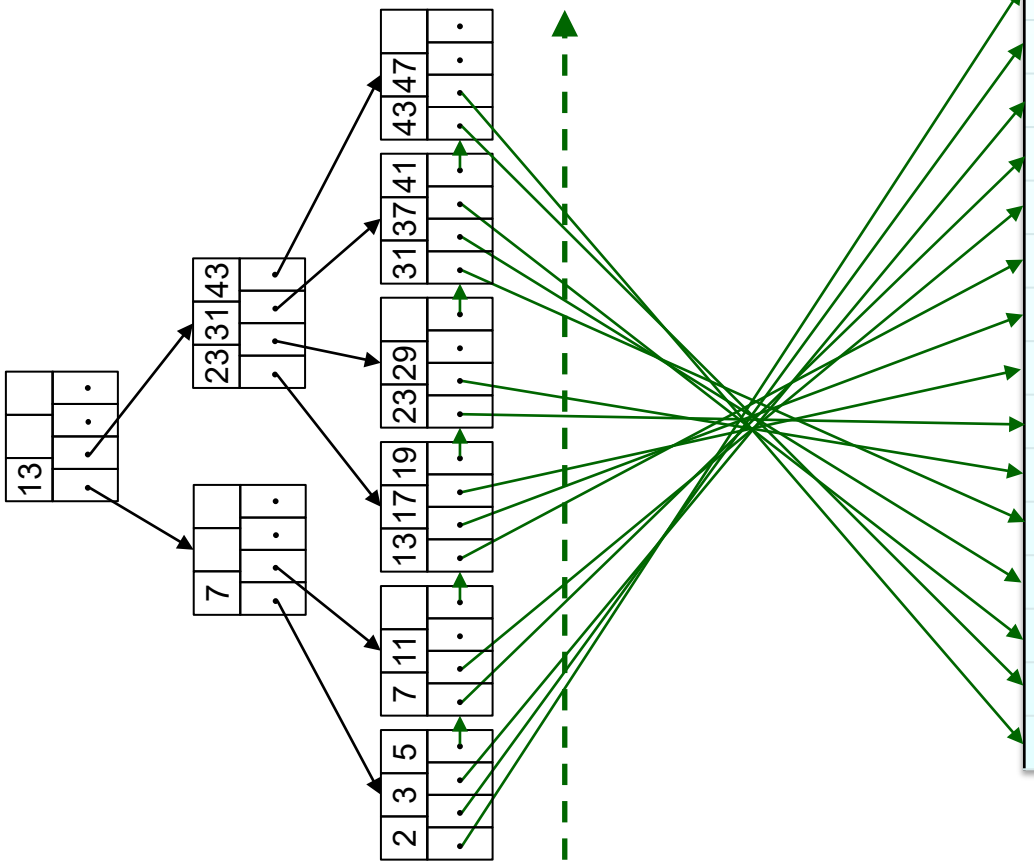


S (поставщики)

IDS	NameS	CityS	Rating	Deposit
31	Петров	Москва	100	1000
2	Иванов Ф	Воронеж	60	2000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000
19	Крюков	Воронеж	100	1000
11	Степанов	Элиста	75	2000
29	Горин	Уфа	85	1000
37	Коваль	Минск	60	2000
23	Федоров	Москва	80	1000
5	Ненашев	Керч	95	1000
41	Глинин	Минск	80	1000
17	Марьин	Тула	60	2000
3	Демин	Москва	50	5000
13	Ильин	Минск	80	1000
7	Иванов К	Москва	100	1000

Пример упорядочивания по индексу

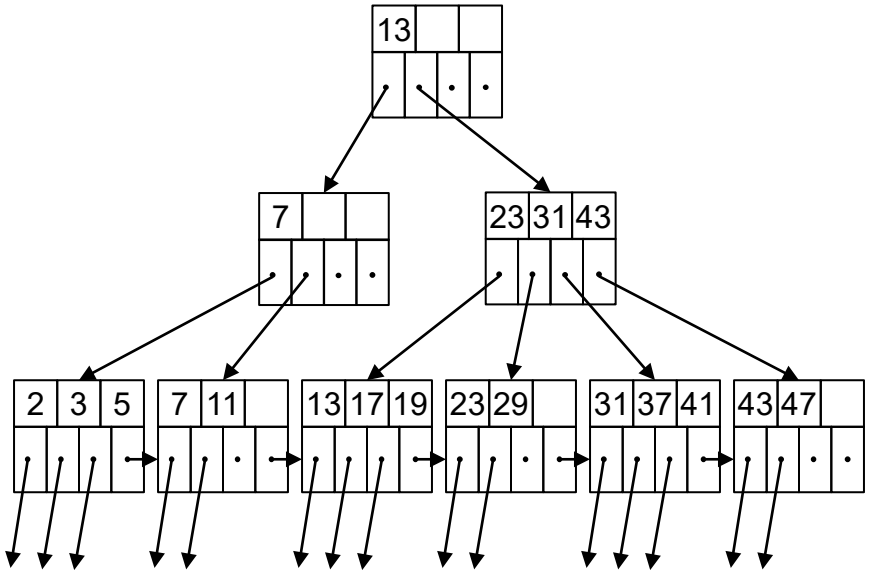
Упорядочить поставщиков по возрастанию IDS



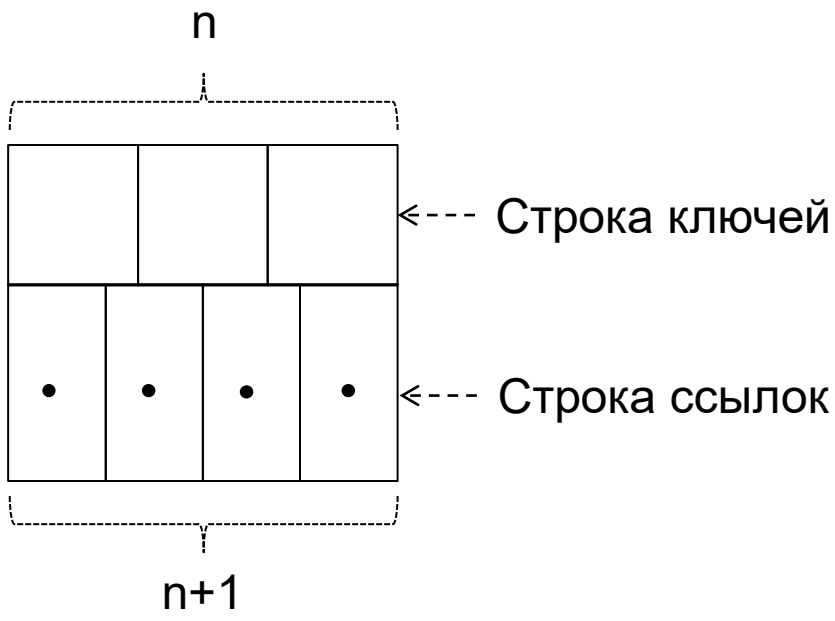
Виртуальная таблица S (упорядочена по IDS)

IDS	NameS	CityS	Rating	Deposit
2	Иванов Ф	Воронеж	60	2000
3	Демин	Москва	50	5000
5	Ненашев	Керч	95	1000
7	Иванов К	Москва	100	1000
11	Степанов	Элиста	75	2000
13	Ильин	Минск	80	1000
17	Марьин	Тула	60	2000
19	Крюков	Воронеж	100	1000
23	Федоров	Москва	80	1000
29	Горин	Уфа	85	1000
31	Петров	Москва	100	1000
37	Коваль	Минск	60	2000
41	Глинин	Минск	80	1000
43	Сидоров	Москва	80	1000
47	Абрамов	Сургут	50	5000

B-дерево

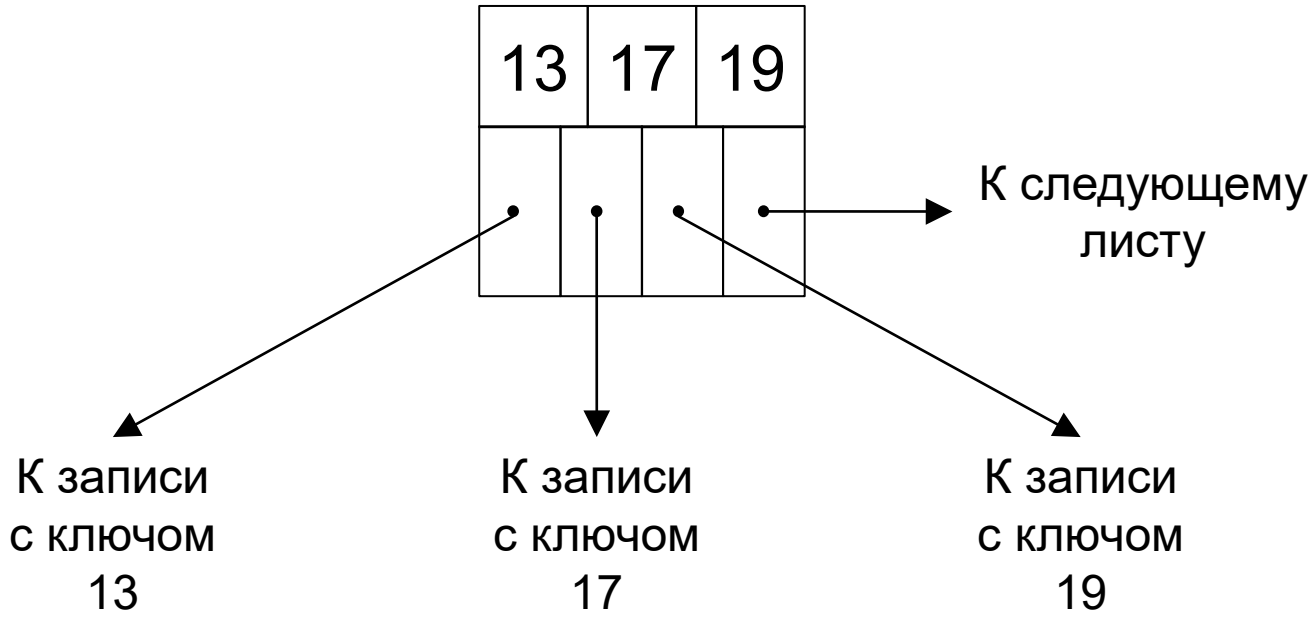


Структура узла

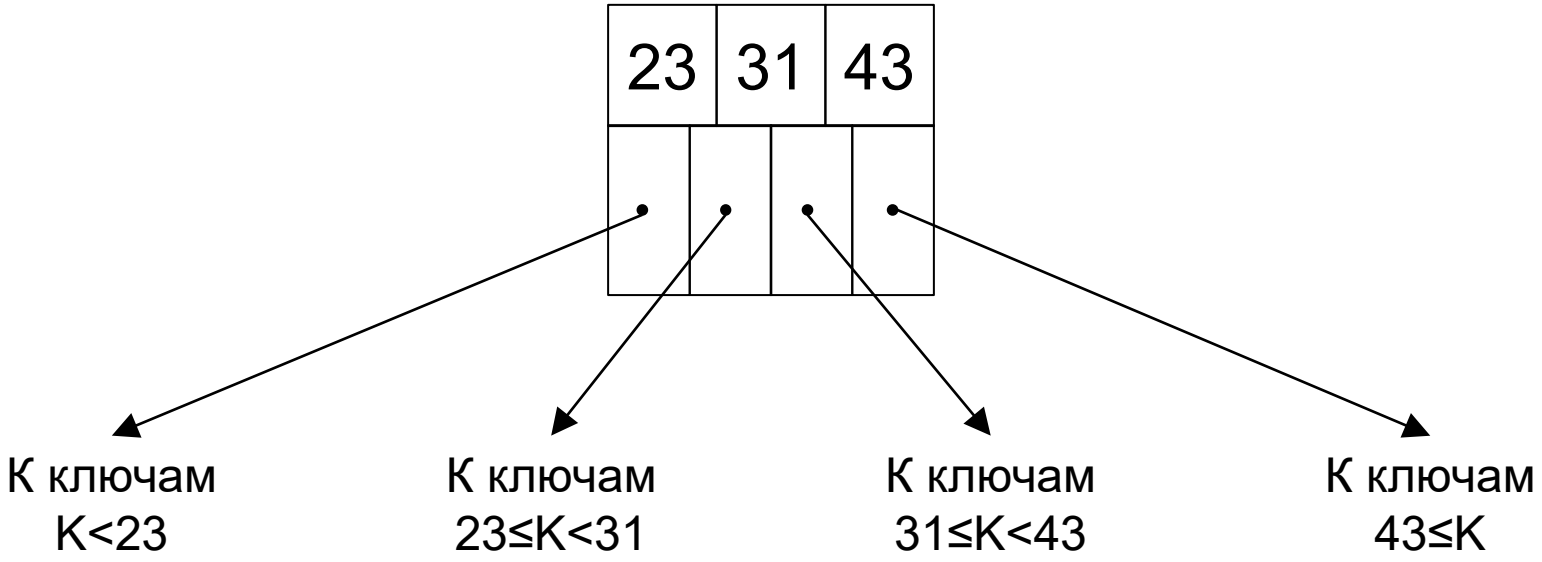


n – параметр индексного файла (в примере n=3)

Семантика полей листа

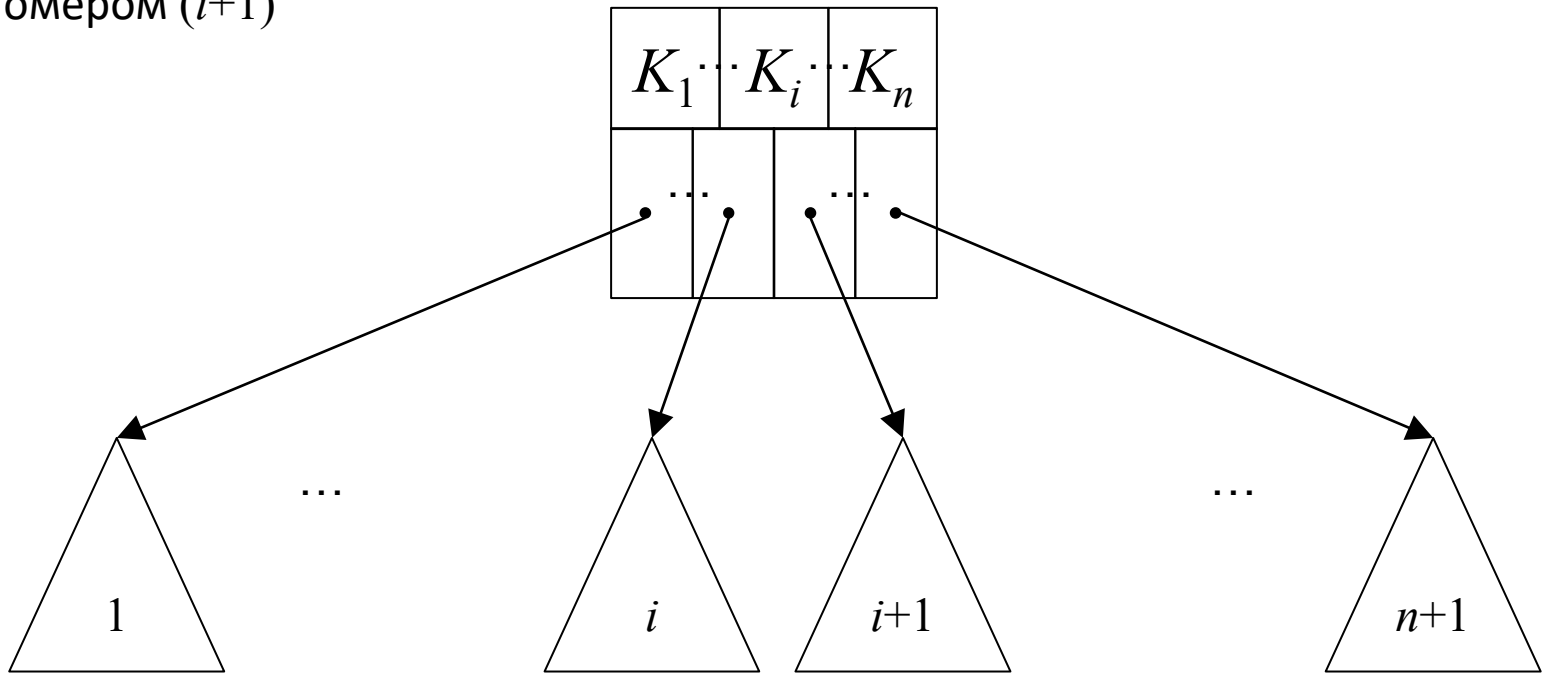


Семантика полей нелистового узла



Семантика полей нелистового узла в общем случае

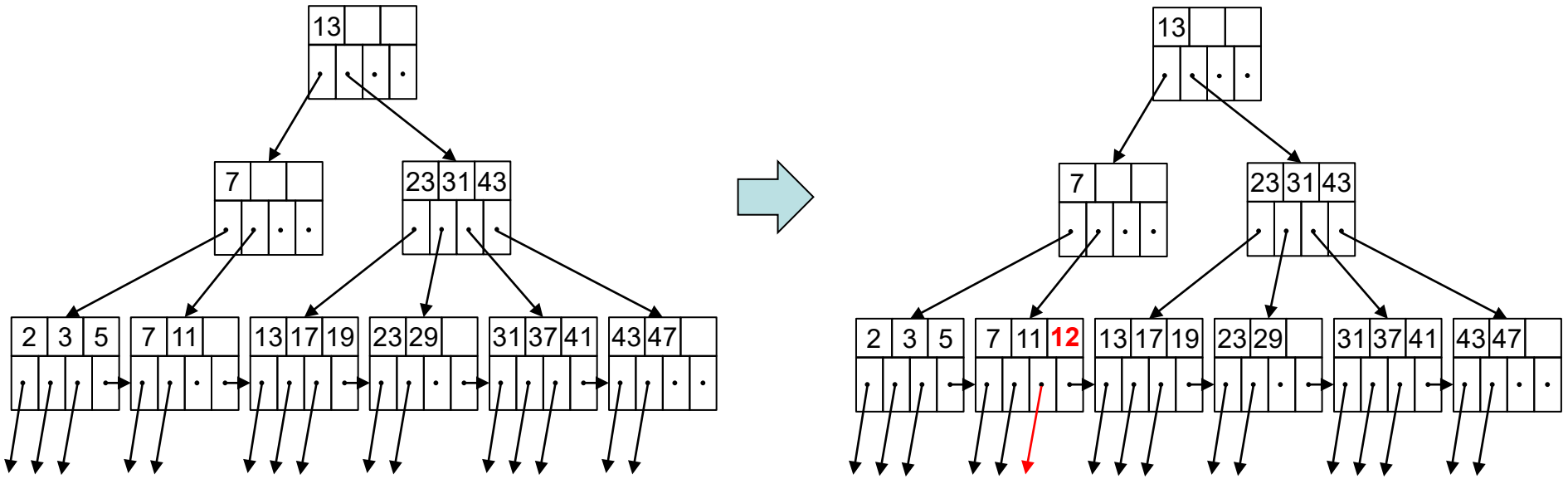
K_i – минимальное значение в поддереве с номером $(i+1)$



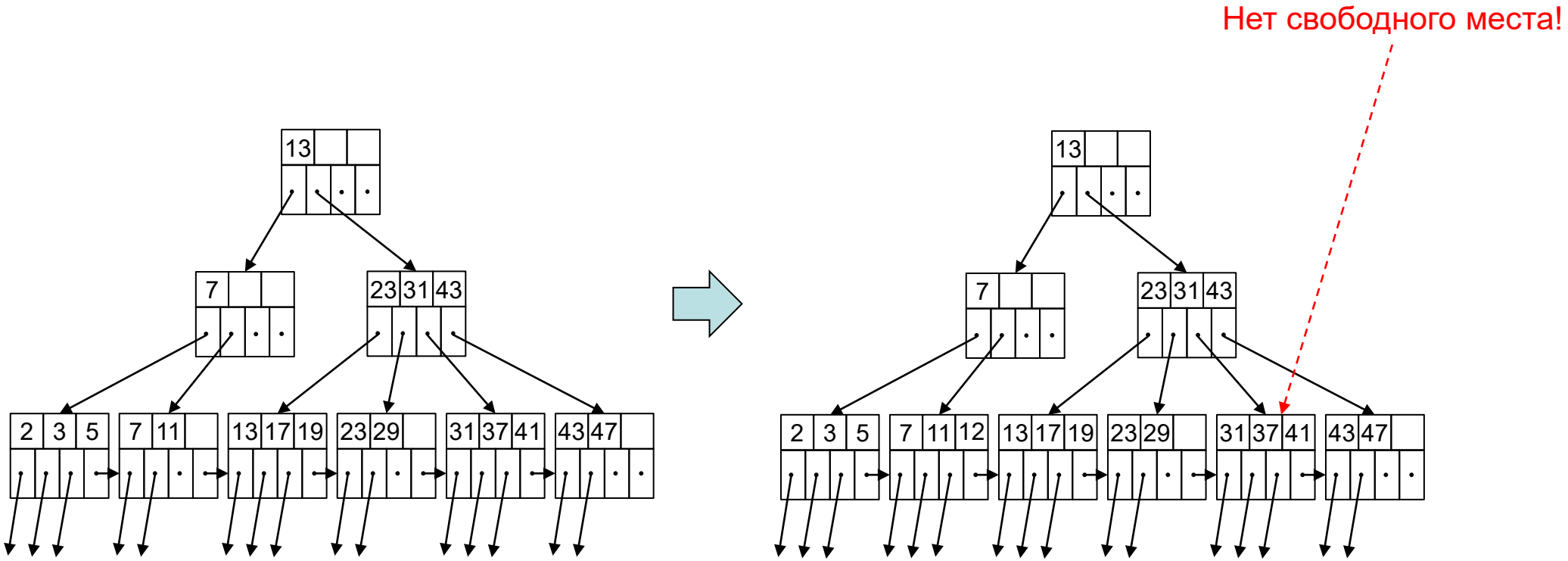
Ограничения

- Корневая вершина содержит не менее двух указателей (за исключением тривиального случая, когда индексируемое отношение имеет один кортеж)
- Ключи в листьях располагаются в порядке возрастания значений слева направо
- В листе должно быть заполнено не менее $\lfloor (n+1)/2 \rfloor$ указателей (? для $n = 3$)
- Во внутренней вершине должно быть заполнено не менее $\lceil (n+1)/2 \rceil$ указателей (? для $n = 3$)

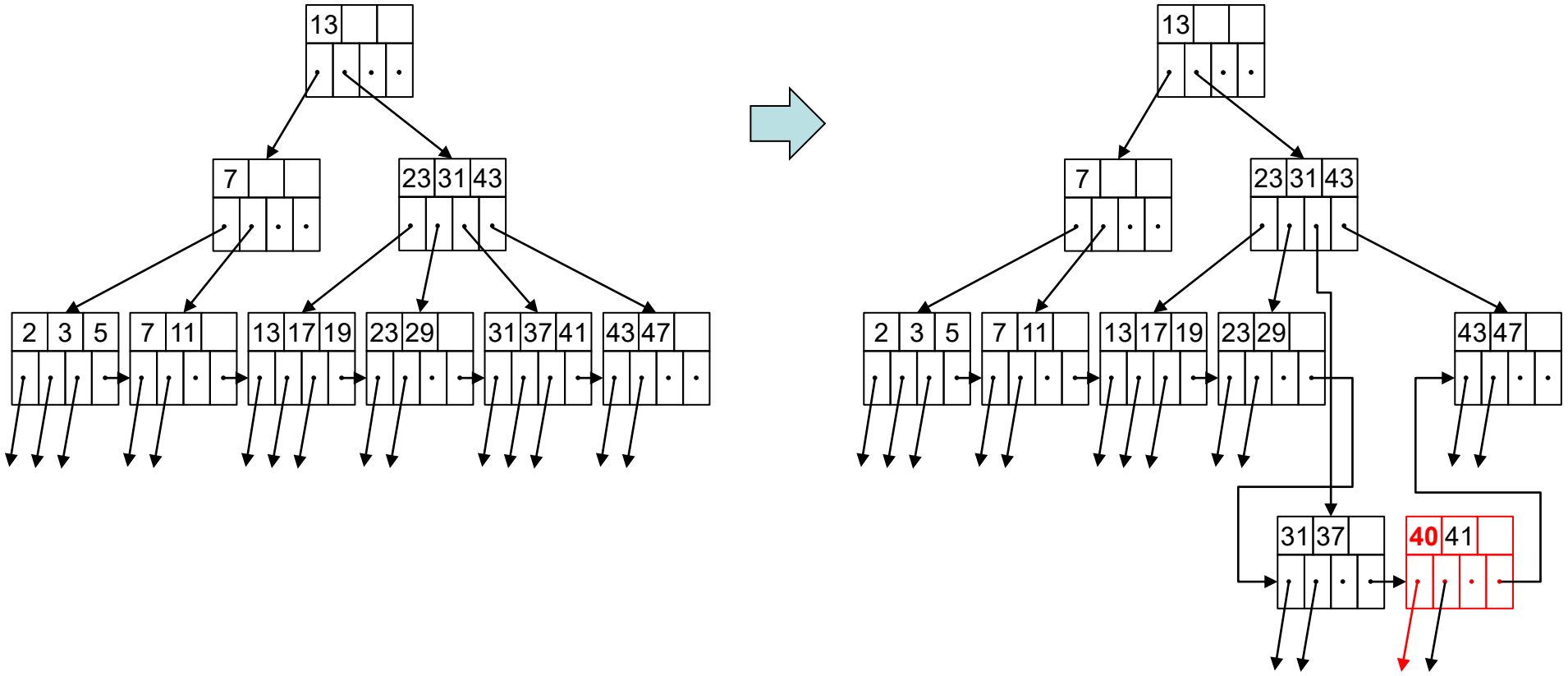
Вставка ключа 12



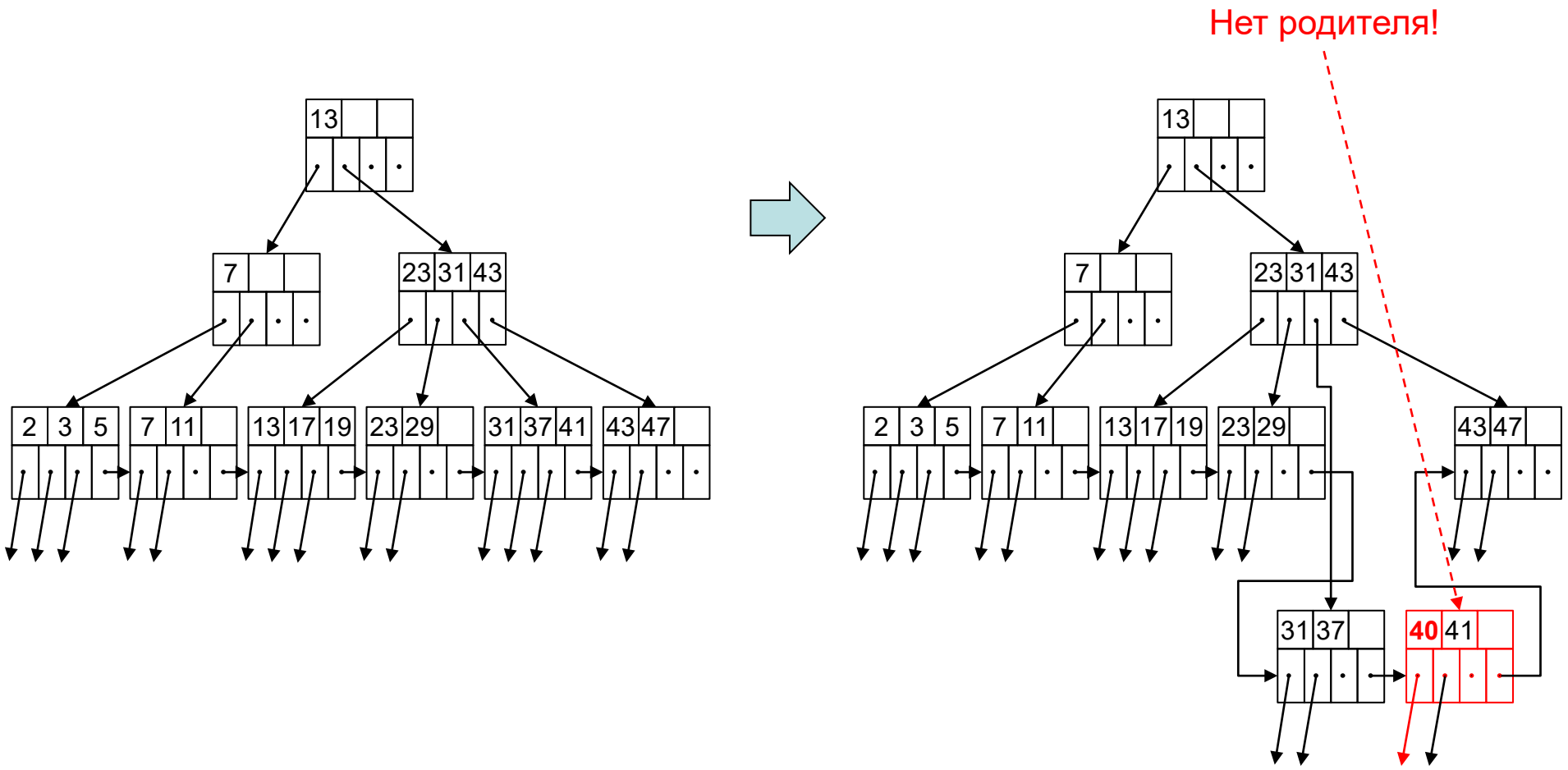
Вставка ключа 40



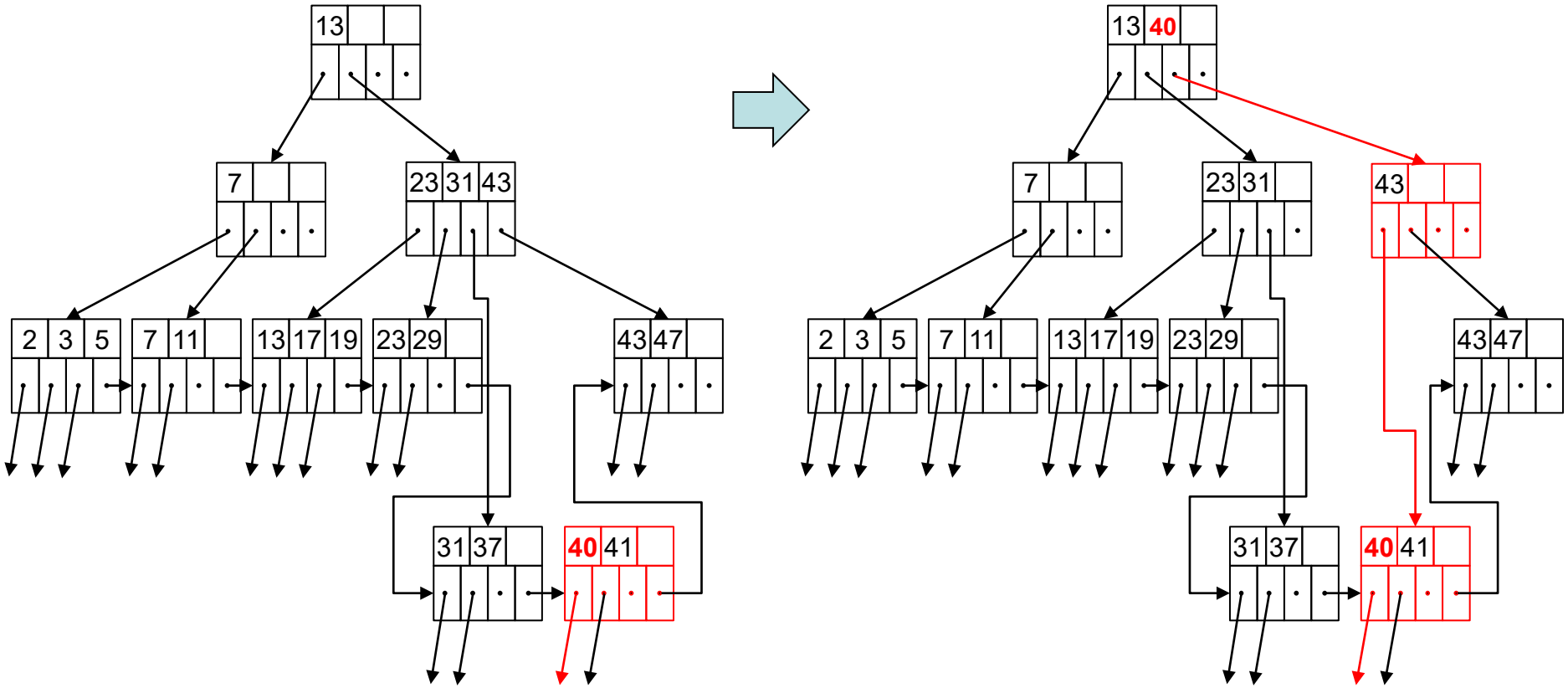
Вставка ключа 40: фаза I



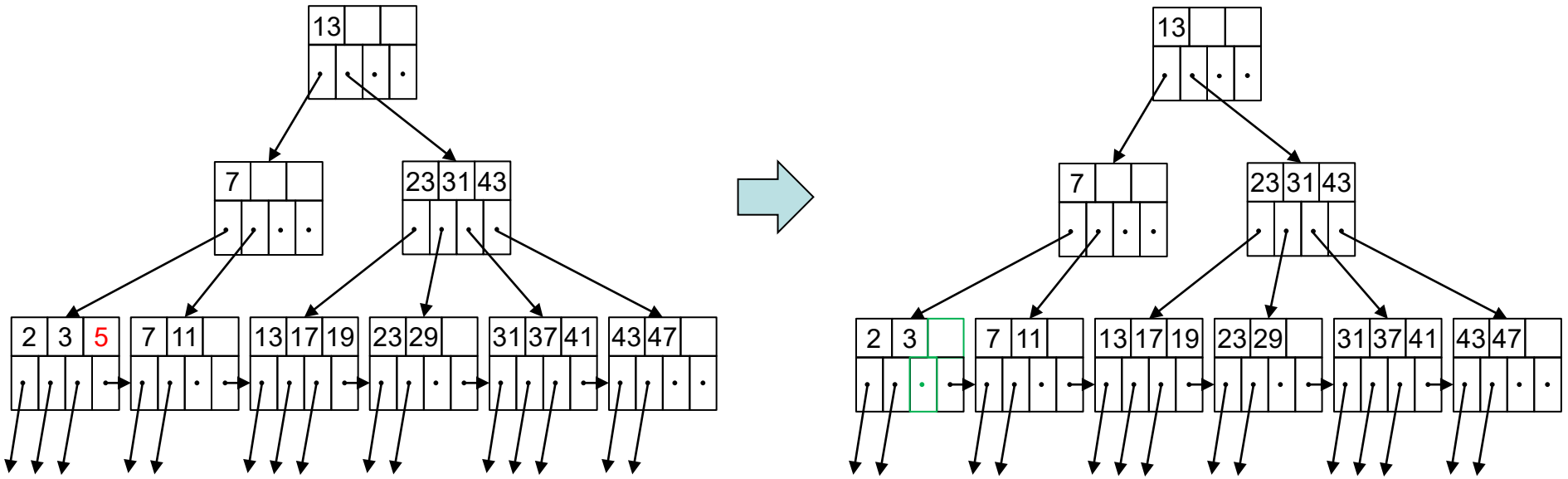
Вставка ключа 40: фаза I



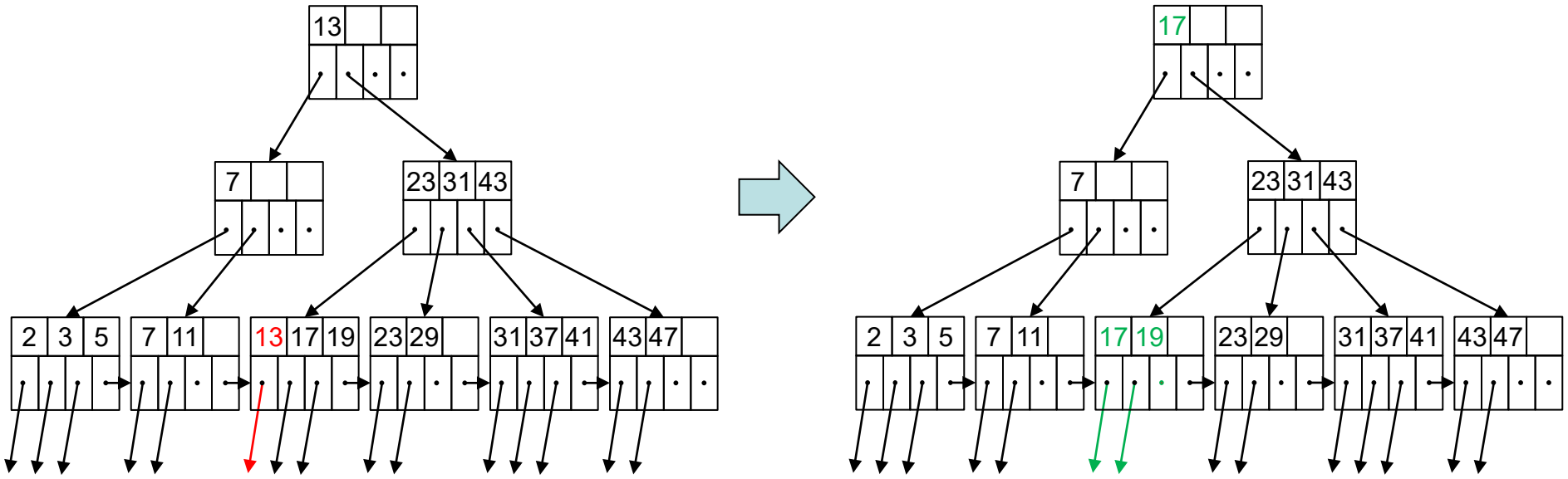
Вставка ключа 40: фаза II



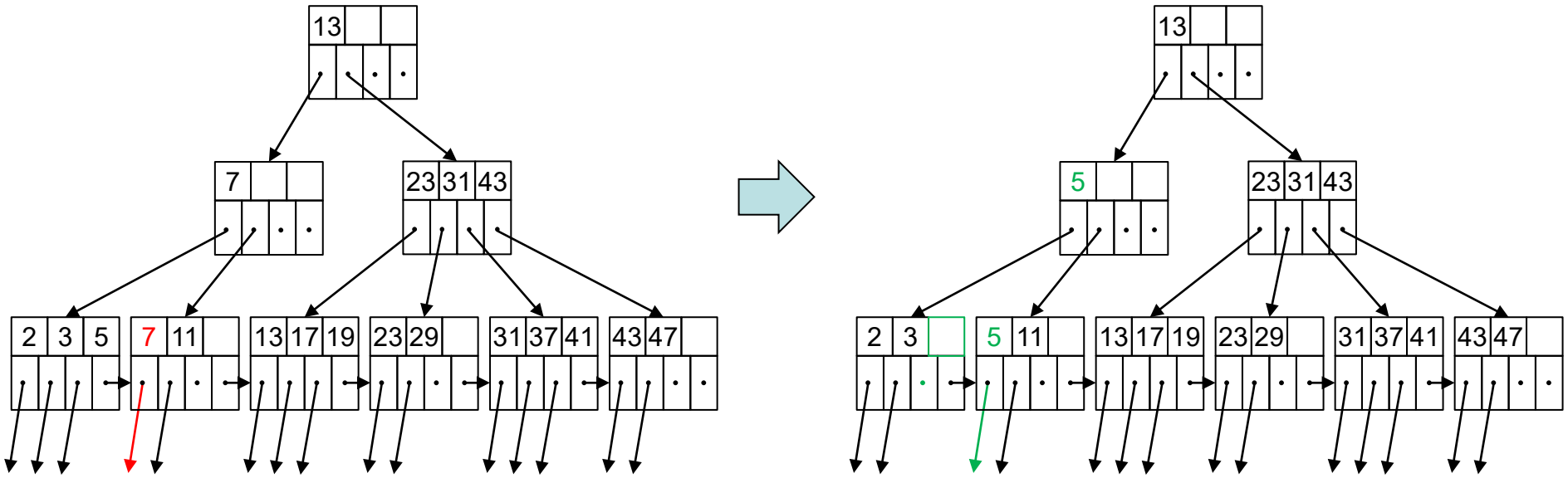
Удаление ключа 5



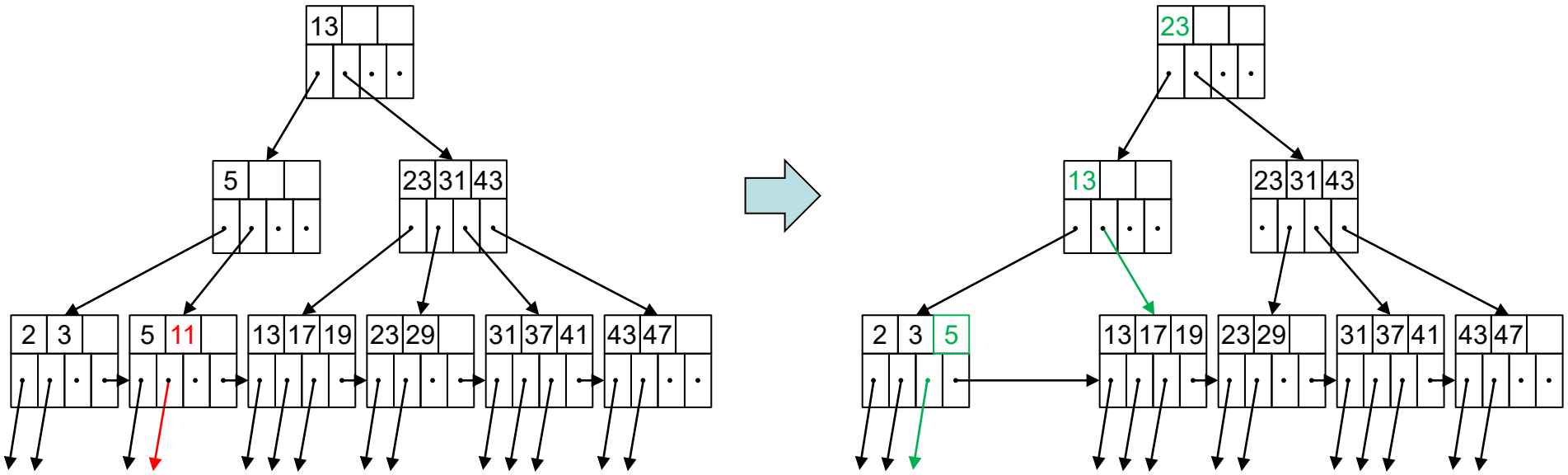
Удаление ключа 13



Удаление ключа 7

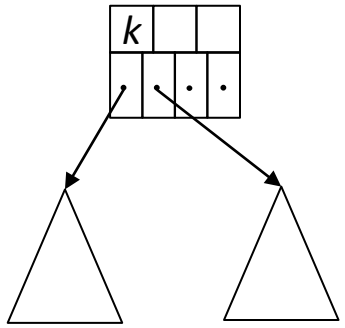


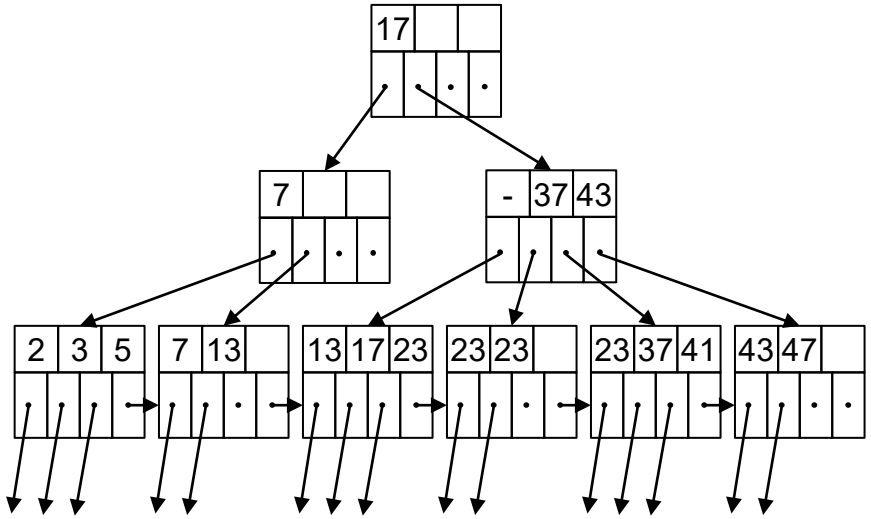
Удаление ключа 11



B-дерево с дубликатами

- *k* – минимальное *новое* значение ключа в правом поддереве
- Все ключи в левом поддереве строго меньше *k*





Оценка эффективности 3-х уровневых B-деревьев

- Пусть блок имеет размер 4096 байт (за вычетом заголовка), ключ занимает 4 байта, указатель занимает 8 байт
- $4n+8(n+1) \leq 4096$ => максимальное $n=340$,
  то есть один блок может содержать 340 пар «ключ-указатель»
- $\frac{340+340/2}{2} = 255$ – среднее между максимальным и минимальным количеством ссылок в блоке
- Корневая вершина адресует 255 поддеревьев
- Вершины второго уровня адресуют
  $255^2 = 65\ 025$ листьев
- Листья адресуют $255^3 = 16\ 581\ 375$ записей

Конец лекции 5