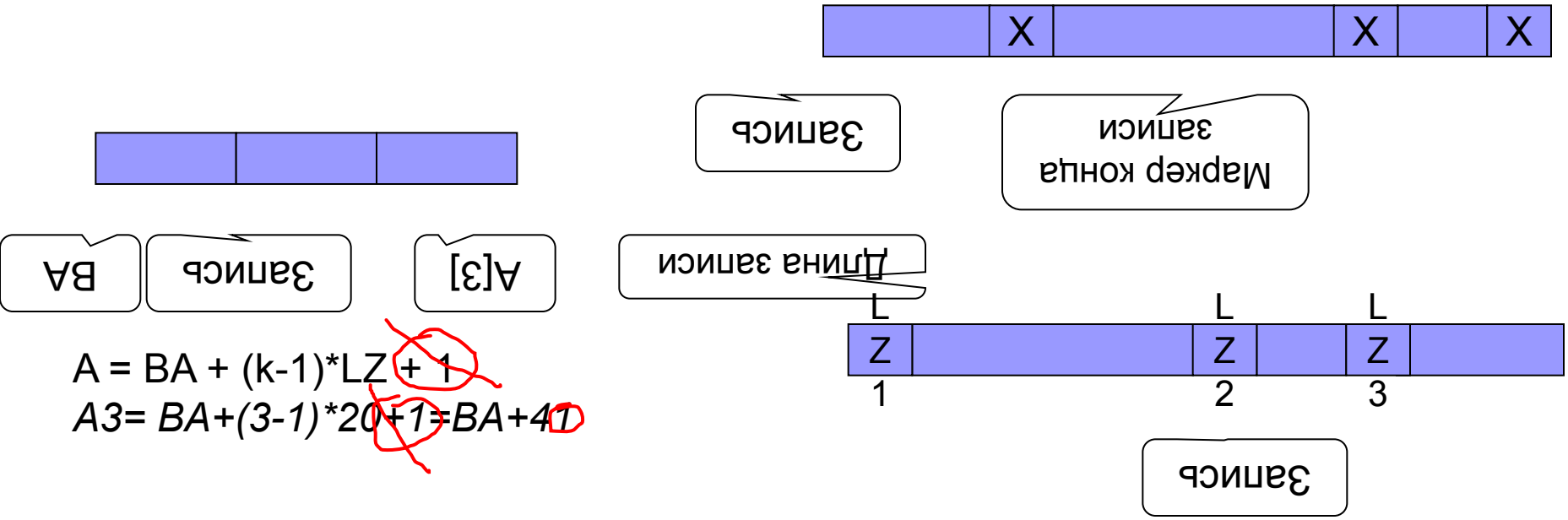
Физические модели баз данных

Управление базами данных

Внешние устройства хранения информации

- Устройства произвольного доступа (например, магнитные диски). Файлы с постоянной длиной записи. Местоположение записи файла определяется **адресом**.
- Устройства последовательного доступа (например, стример). Файлы с переменной длиной записи



A = BA + (k-1)*LZ + 1

*A3= BA+(3-1)*20+1=BA+41*

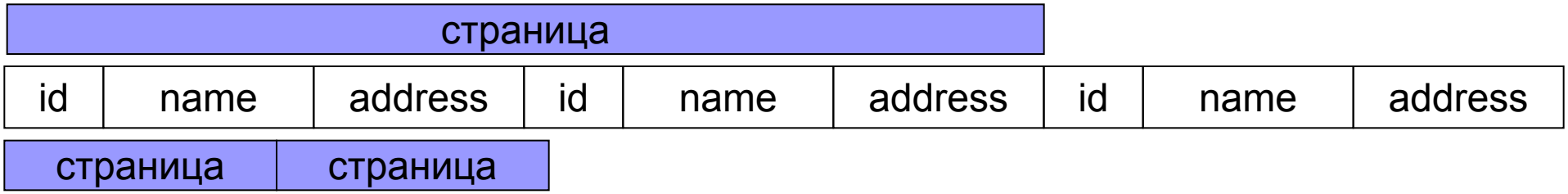
Физическая модель

Далее будем рассматривать только файлы с постоянной длиной записи

Упрощение: для каждой таблицы отдельный файл

Физическая модель

- Логическая запись (запись) – кортеж отношения
- Физическая запись (страница, блок) – единица обмена данными между первичной и внешней памятью



Порядок выполнения операции над данными (в общем случае):

1) Найти страницу на внешнем носителе, содержащую нужную запись
2) Прочитать страницу в первичную память
3) Найти нужную запись в прочитанной странице
4) Выполнить действие над записью в первичной памяти
5) Сохранить страницу с измененной записью во вторичной памяти

Физическая модель

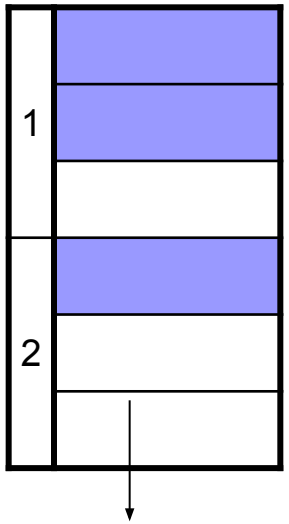
На производительность СУБД влияют:

- Организация файла: распределение данных по записям и страницам на внешнем устройстве
  - последовательная неупорядоченная организация
  - последовательная упорядоченная организация
  - хеширование данных
- Метод доступа: алгоритм сохранения и извлечения записей из файла с определенной организацией хранения данных

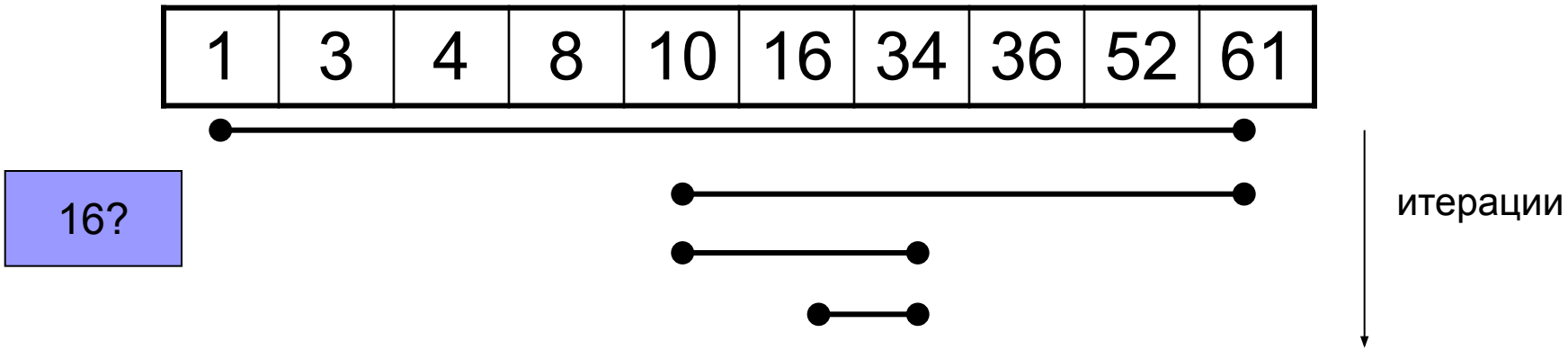
Последовательная неупорядоченная организация файла

Последовательный неупорядоченный файл (куча) – простейший тип структуры файла

- Записи размещаются в файле в том порядке, в котором добавляются – новая запись на последнюю страницу
- При удалении записи место повторно не используется, поэтому потеря эффективности со временем. Требуется периодическая реорганизация
- Требуется периодически перестраивать файл
- Поиск записи – линейный поиск, перебор всех страниц (чтение страницы, поиск внутри страницы)
- Удаление записи – найти нужную страницу, загрузить, изменить страницу в памяти, сохранить на место
- Такая организация эффективна при пакетной загрузке данных (последовательных)



Метод дихотомии



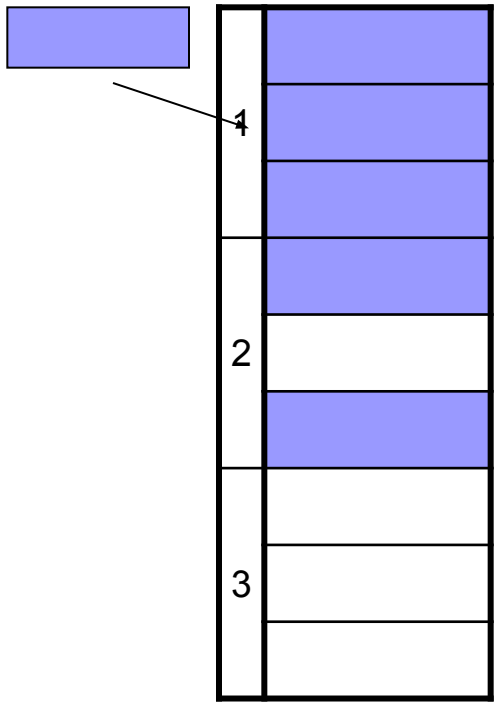
На примере упорядоченного массива:

1) Найти срединный элемент. Выбирается та половина на которая должна содержать заданное значение (по результатам сравнения значения со значениями левой и правой границ половин)
2) Если элемент не является границей, то в качестве исходного элемента рассматривается выбранная половина. Переход к п.1.

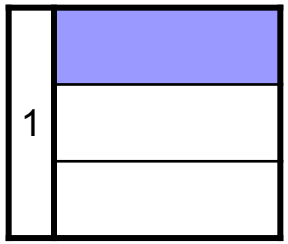
Среднее количество операций ~$\ln_2(n)$,
а последовательный поиск ~n

Последовательная упорядоченная организация файла

- В последовательных упорядоченных файлах записи упорядочены по одному или нескольким полям
- Поиск выполняется быстро методом дихотомии (бинарный поиск)
- Операция вставки – трудоемкая:
  - найти страницу для вставки
  - если на странице есть свободные места – перестроить записи на странице, если нет – то надо сдвинуть все записи к концу. Вставка в начало наиболее трудоемкая
- Удаление – быстрая операция, т.е. после удаления файл не перестраивается



Модификация:
область переполнения – последовательное неупорядоченное хранение



Хеширование данных

- Основная идея – записи в файле ***прямого доступа*** находятся в «перемешанном» порядке (hash = путаница): записи с близкими значениями ключа находятся далеко друг от друга. Поэтому при вставке записи велика вероятность того, что соответствующее место будет свободно и перестраивать файл не потребуется
- Функция перемешивания
  A = h(K), где K – значение ключа записи, A – адрес в файле. K – числовое поле, либо однозначно приводится к числовому виду.
- Например,
  A = K mod N, где N – некоторое заранее заданное число, mod – операция получения остатка от деления по модулю N.

Ключ	Не ключевые поля	**Адрес = K mod 5**
7	Петров	**2**
11	Иванов1	**1**
18	Сидоров1	**3**
20	Петров2	**0**
24	Иванов2	**4**
26	Сидоров3	**1**
27	Пушкин	**2**

близкие значения ключей, дальние адреса

конфликты – одинаковые значения адресов

Хеширование данных

Коллизия: h(Ki) = h(Kj)
Значения таких ключей – «синонимы»

Способы разрешения коллизий:

- Область переполнения:
  - несвязанная
  - связанная
- Свободное замещение

Хеширование данных

- Несвязанная область переполнения

PK

Адрес = PK mod5

Основная область

0	20	Петров2
1	11	Иванов1
2	7	Петров
3	18	Сидоров1
4	24	Иванов2

Область переполнения
(последовательное неупорядоченное хранение)

26	Сидоров3
27	Пушкин
31	Пушкин2

Хеширование данных

- Связанная область переполнения

Основная область

0	20	Петров2	-
1	11	Иванов1	0 2
2	7	Петров	1
3	18	Сидоров1	-
4	24	Иванов2	-

Область переполнения
(связанный список)

0	26	Сидоров3	~~2~~
1	27	Пушкин	-
2	31	Сидоров4	-
3			

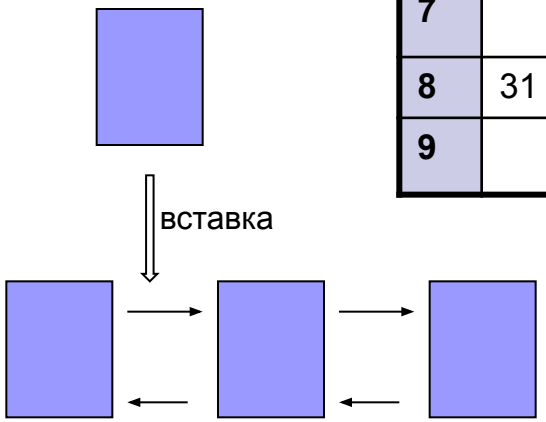
Хеширование данных

- Свободное замещение

предыдущий | следующий

Основная область

0	20	Петров2	-	-
1	11	Иванов1	-	5
2	7	Петров	-	6
3	18	Сидоров1	-	-
4	24	Иванов2	-	-
5	26	Сидоров4	1	8
6	27	Пушкин	2	-
7				
8	31	Сидоров4	6	-
9				



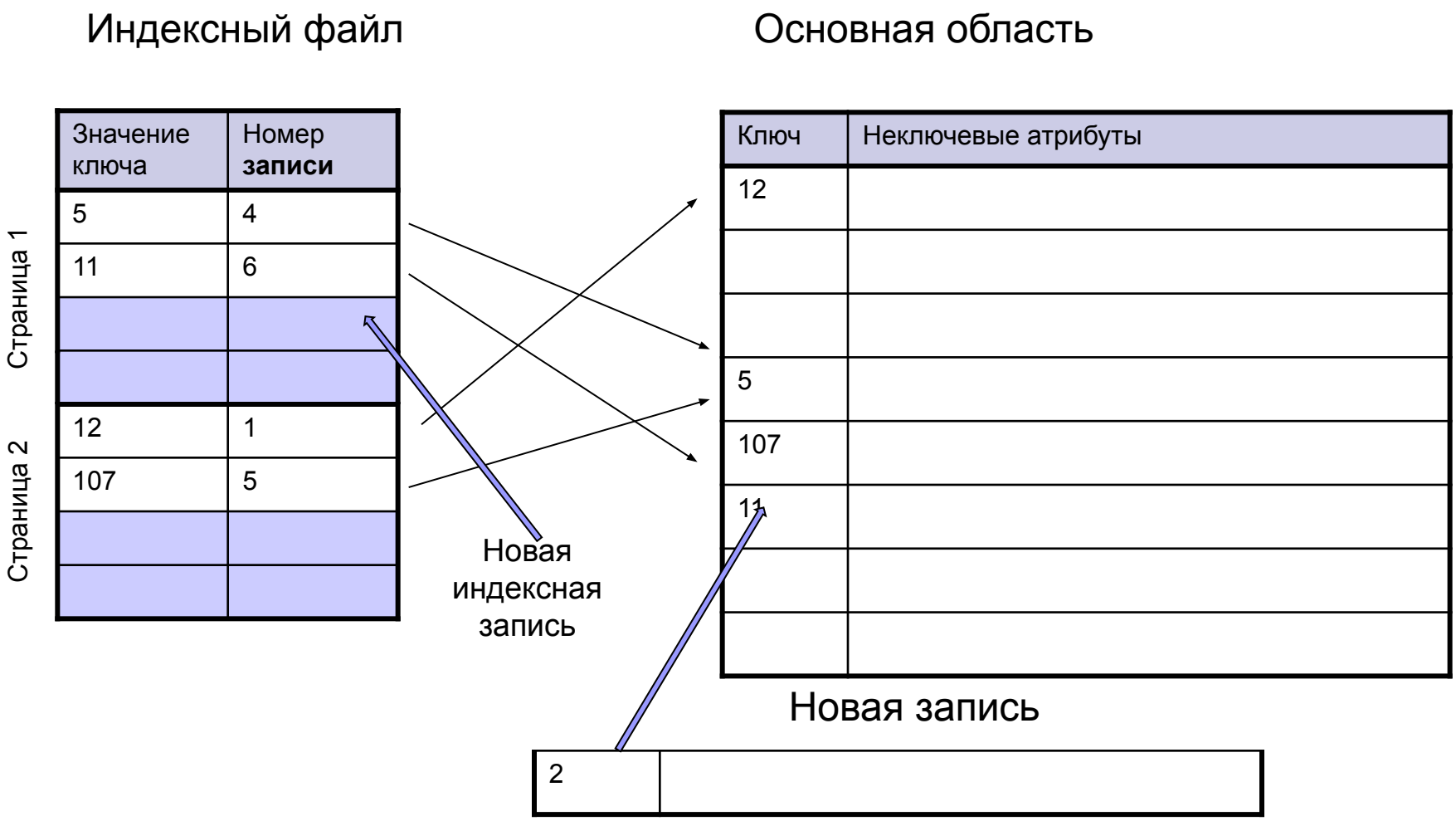
?	41	Сидоров5	?	?

Индексные файлы

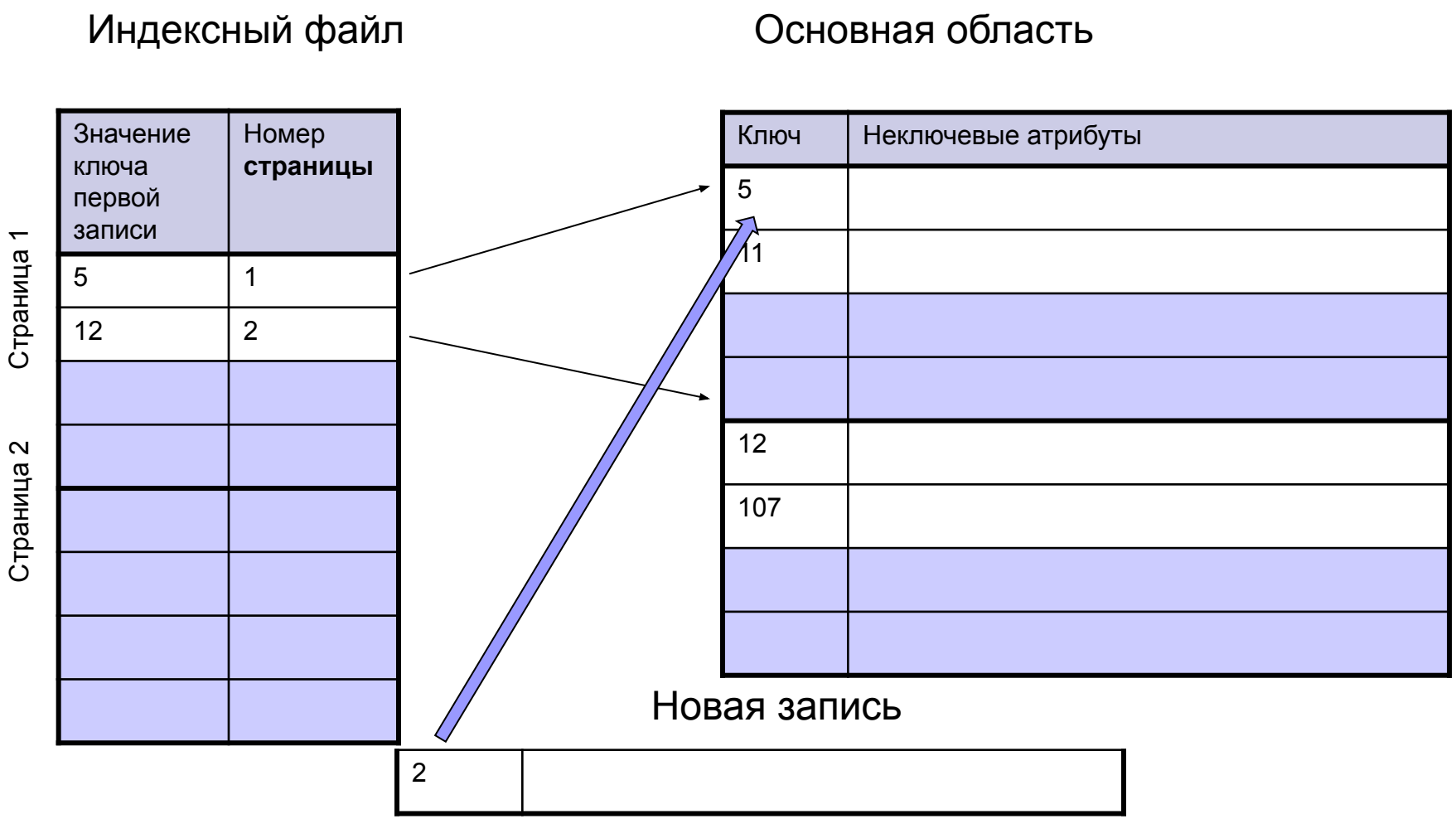
Снижение времени поиска:

- Бинарный поиск
- Часть индексного файла – в оперативной памяти

Файлы с плотным индексом



Файлы с неплотным индексом



Многоуровневые индексы

Индексный файл 1

Значение ключа первой записи	Номер **страницы**
2	
23	

Индексный файл 2

Значение ключа первой записи	Номер **страницы**
2	
9	
15	
23	

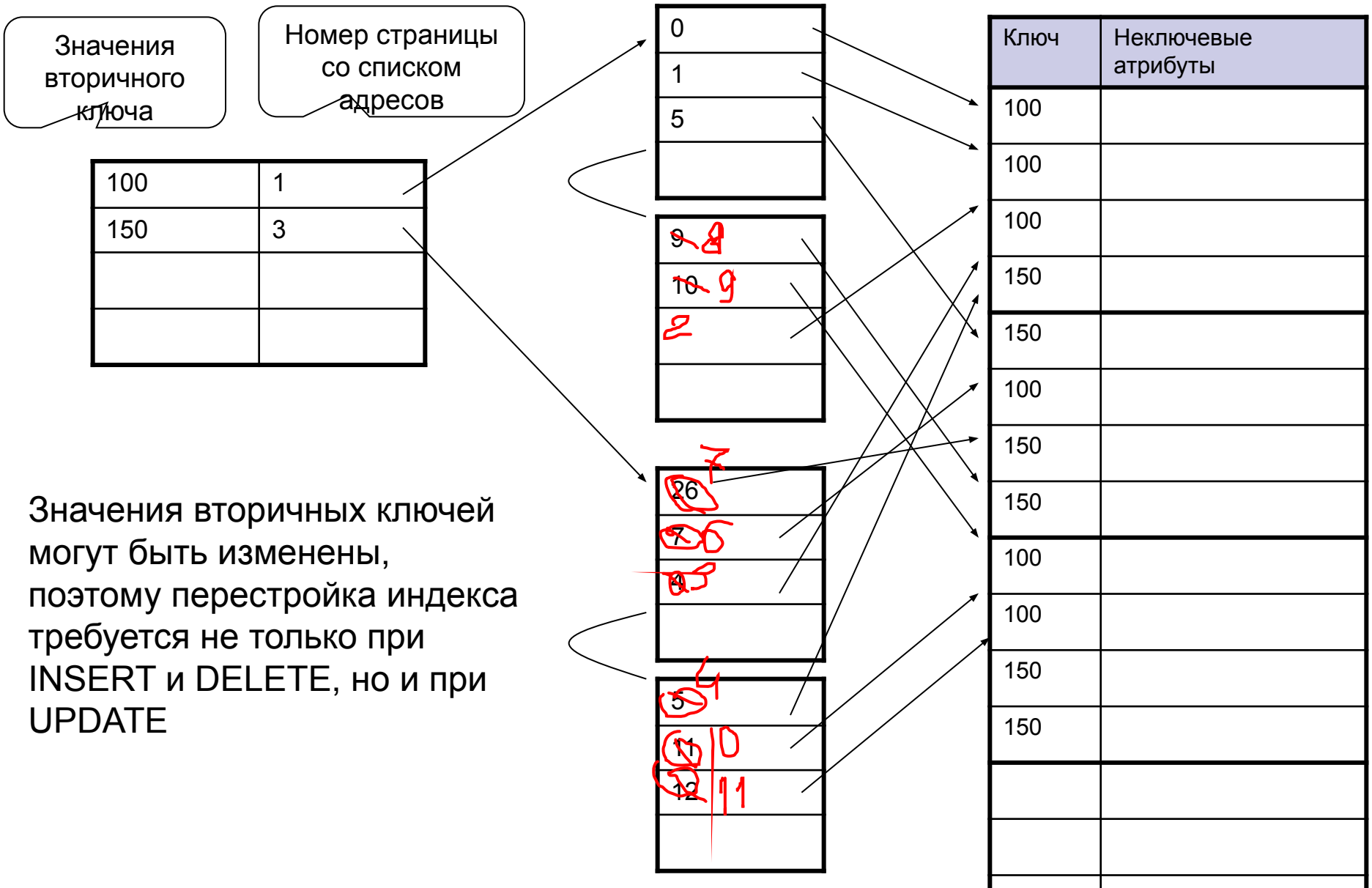
Основная область

Ключ	Неключевые атрибуты
2	
3	
6	
9	
10	
13	
15	
18	
21	
23	
33	

Вторичные ключи

Вторичный ключ – ***произвольный*** набор атрибутов, которому соответствует набор искомых записей в операции выборки (значения ключа – не уникальные)

Вторичные ключи



Значения вторичных ключей могут быть изменены, поэтому перестройка индекса требуется не только при INSERT и DELETE, но и при UPDATE

Индексы в Transact-SQL

Виды индексов:

- ***Кластерный*** (неплотный индекс). Один на таблицу. Для первичного ключа автоматически создается кластерный индекс, если не указан тип NONCLUSTERED.
- ***Некластерный*** (плотный индекс). До 249 для таблицы. Если в таблице не существует кластерный индекс, то ссылки указывают на записи в основной области. Если в таблице имеется кластерный индекс, то все некластерные индексы ссылаются на записи кластерного индекса. Это позволяет избежать перестройку некластерных индексов при упорядочении записей (как это требует кластерный индекс). Если допускаются неуникальные значения ключей, то сервер БД добавляет к ним дополнительные значения, делая их уникальными.

Уникальность:

- ***Уникальный индекс*** (кластерный или некластерный) гарантирует уникальность значений в индексируемом столбце. Вставка дубликатов будет отклоняться. Вместо требования уникальности индекса можно использовать ограничения PRIMARY KEY или UNIQUE.
- ***Не уникальный индекс***. Не гарантирует уникальность значений.

Длинные (символьные и составные) ключи снижают производительность и требуют затрат памяти (значения ключей дублируются в индексе)

Индексирование увеличивает время вставки, поэтому индексирование должно быть оправданным. Для часто изменяемых столбцов следует использовать некластерные индексы.

Индексы в Transact-SQL

Способы задания индексов:

- Автоматическое создание при объявлении первичного ключа. Объявление PRIMARY KEY создает кластерный индекс.
- Автоматическое создание при объявлении ограничения целостности UNIQE. Объявление UNIQUE создает уникальный некластерный индекс. Их может быть более одного.
- Команда CREATE INDEX

Индексы в Transact-SQL

Сокращенное описание

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX *index_name*
ON *table* **(** *column* [ASC | DESC] [**,**...*n*] **)**
[WITH **(** <relational_index_option> [**,**...*n*] **)**]

<relational_index_option> ::=
{
PAD_INDEX = { ON | OFF } |
FILLFACTOR = *fillfactor* |
IGNORE_DUP_KEY = { ON | OFF }
}

- *index_name* – имя индекса должно быть уникальным в пределах одной таблицы
- PAD_INDEX – резервировать свободное пространство на страницах (используется вместе с FILLFACTOR. Используется только при перестроении индекса
- FILLFACTOR – задает процент (от 1 до 100) заполнения страниц. Используется только при перестроении индекса
- IGNORE_DUP_KEY – используется только для уникальных индексов. Если параметр указан, то дубликаты игнорируются. Если параметр не указан, то откатывается вся транзакция

Примеры

Автоматическое создание кластерного индекса

```
CREATE TABLE t1 (a int, b int, c int PRIMARY KEY);
```

Явное создание уникального кластерного индекса

```
CREATE TABLE t1 (a int, b int, c int);
CREATE UNIQUE CLUSTERED INDEX Idx1 ON t1(c);
```

Простой некластерный индекс

```
CREATE INDEX IX_ProductVendor_VendorID
ON ProductVendor (VendorID);
```

Составной некластерный индекс

```
CREATE NONCLUSTERED INDEX IX_SalesPerson_SalesQuota_SalesYTD ON
SalesPerson (SalesQuota, SalesYTD);
```

Уникальный некластерный индекс

```
CREATE UNIQUE INDEX AK_UnitMeasure_Name ON Production.UnitMeasure(Name);
```

Уникальный индекс, игнорирующий дубликаты

```
CREATE UNIQUE INDEX AK_Index ON Test (C2)
WITH (IGNORE_DUP_KEY = ON);
```