8. ФИЗИЧЕСКАЯ ОРГАНИЗАЦИЯ ДАННЫХ

8.1. Общие сведения

Форма хранения данных в ЭВМ часто существенно отличается от представления пользователя и программиста. Внутреннее (физическое) представление данных связано с тем, как они хранятся на носителях информации. Терминология при этом зависит от того, какой носитель информации имеется в виду - оперативная память ЭВМ, магнитный диск, магнитная лента, оптический диск. Рассмотрим основные из этих терминов.

Физическая запись (ее еще называют блоком или сектором) - элементарная единица данных, которая может быть считана или записана одной командой ввода - вывода ЭВМ. Например, размер сектора для MS DOS - 512 байт. Обычно одна физическая запись состоит из нескольких логических записей. Обмен данных с оперативной памятью происходит только целыми блоками.

Часть оперативной памяти, предназначенная для хранения одной физической записи, называется ***буфером***. Количество буферов задается пользователем при настройке операционной системы.

Кластер - это несколько (2-4) смежных секторов.

Файл - это совокупность физических записей, они могут размещаться как в смежных областях внешней памяти, так и быть “разбросаны”.

Физическая организация данных - это устойчивый порядок расположения записей и способ обеспечения взаимосвязей между ними.

Логическая и физическая организации данных относительно независимы, т.к. преследуют разные цели. Цель логической организации данных - наиболее ясное и полное отображение информации об объектах и связях между ними. Цель физической организации - максимальная скорость выдачи ответа на запрос и эффективное использование памяти ЭВМ.

Обычно при создании ЭИС выбирают ту физическую организацию данных , которая принята для выбранных технических средств. Изменение ее может произвести лишь системный программист. Физическая организация данных определяется типом ЭВМ, ОС и СУБД. При этом учитывают следующие **факторы:**

- эксплуатационные характеристики технических средств (объем памяти, внешней памяти, тип внешних ЗУ, скорость ввода-вывода);
- стоимость технических средств;
- тип преобладающих запросов к ЭИС.

Можно выделить **три основных типа запросов:**

- **создание структуры,** включая размещение основных и вспомогательных данных;
- **поиск** по одному или нескольким ключам;
- **коррекция,** включая поиск ее места и перестройку структуры основных и вспомогательных данных.

Рассмотрим ***критерии оценки эффективности*** физической организации данных:

1. *Эффективность доступа* **Эд = 1/N** , где **N** - среднее число физических обращений к записям для организации одного логического запроса, зависит от типа запроса и организации данных.

2. *Эффективность хранения* **Эх = 1/К** , где **К** - среднее число дополнительных байтов памяти, необходимых для хранения одного байта основных данных (дополнительная память расходуется для хранения указателей, служебных таблиц, индексов, резервных участков и т.п. и зависит от метода организации данных).

3. *Доля выборки* **D = Мнуж / М** , где Мнуж - число нужных записей для запроса; **М** - общее число записей файла.

В зависимости от **D** различают три вида поиска:

- получить все или многие записи [D = 10 - 100%],
- получить уникальную запись [Мнуж = 1],
- получить некоторые записи [D = 0 - 10%].

Совокупность соглашений о физической организации данных и алгоритмов выполнения запросов называют ***методом доступа***. Обычно выделяют **три группы методов доступа:**

1. Последовательные методы (применяют для D = 10 -100%);
2. Индексные методы (для D = 0 -10%)
3. Адресные (эффективны для Мнуж = 1).

8.2. Последовательные методы доступа

Они предполагают последовательный перебор записей, когда расположение в памяти **i**-ой записи определяется в зависимости от места **(i-1)** - ой записи. Первая запись имеет фиксированный адрес. В зависимости от способа вычисления адреса следующей (предыдущей) записи различают несколько разновидностей этого метода доступа. Первые три метода характерны тем, что данные занимают в памяти смежные участки. Остальные методы используют понятие списка.

Список - это множество записей, последовательность обработки которых задается с помощью указателей.

Списковые разновидности последовательного метода доступа позволяют логически соединить разрозненные участки памяти.

а) Последовательный доступ с фиксированной длиной записей

Адрес любой записи **Ai** определяется по формуле:

$$Ai = A1 + (i - 1) * L ,$$

где **A1** - адрес первой записи;

L - длина записи;

Если записи располагаются в ОП, то такая организация соответствует понятию "массив" в языках программирования.

Если записи расположены на магнитном диске или ленте, то порядок ввода или вывода данных зависит от языка программирования. Чаще всего допускается только перебор записей от начала к концу файла.

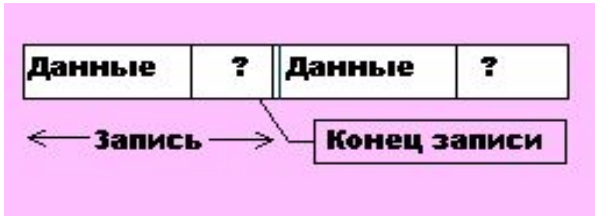
б) Последовательный доступ с записями переменной длины

Обычно каждая запись содержит значение Li - длины этой записи.

Например, в ЕС ЭВМ структура записей переменной длины такова:

в) Последовательный доступ с записями неопределенной длины.

Конец каждой записи помечается специальным символом. Например, в MS DOS текстовые файлы состоят из записей, каждая из которых кончается символом “конец строки”.

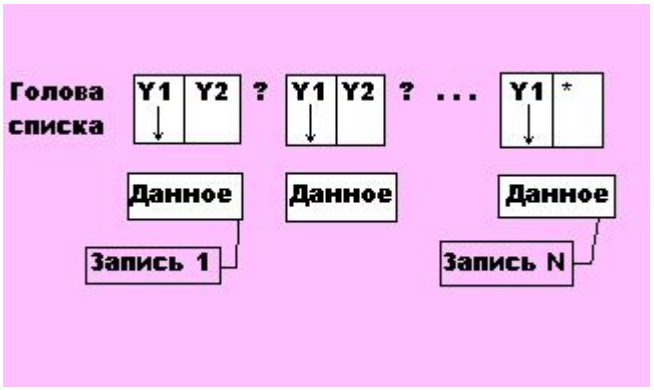


г) Простой линейный список.

Возможны два варианта: когда указатель хранится вместе с записью и когда указатели хранятся отдельно от данных.

Вариант 1:

Вариант 2:

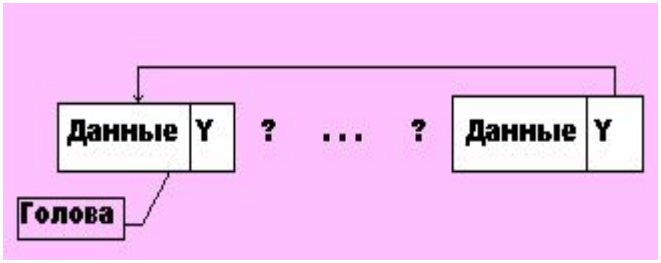


где **Y1** - указатель на данные, а **Y2** - указатель на следующую запись.

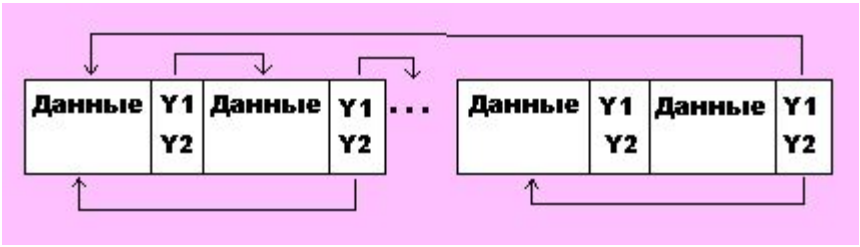
Голова списка (адрес начала списка) хранится в специальной фиксированной ячейке памяти и может быть, как в ОП, так и файле.

д) циклический однонаправленный список

Связь от последней записи идет к первой. Здесь можно получить доступ к любой записи, начиная с любой.



е) Двунаправленный список (простой и циклический).



Каждая запись имеет два указателя, кроме того, список имеет две головы. Достоинство списка: ускоряет доступ к записям.

Заключение

Основой построения списковых структур являются указатели, в качестве которых применяют либо физический адрес записи, либо символический указатель (номер записи или ключ). Первый тип указателей применяют, когда надо получить наибольшую скорость обработки данных. Второй тип указателей позволяет перемещать записи друг относительно друга, включать и удалять записи в список без изменения указателей во всех остальных записях списка, но скорость доступа меньше, чем в первом случае.

Если указатели хранятся вместе с записью, то они называются встроенными; если же отдельно - то они образуют справочник, или индекс.

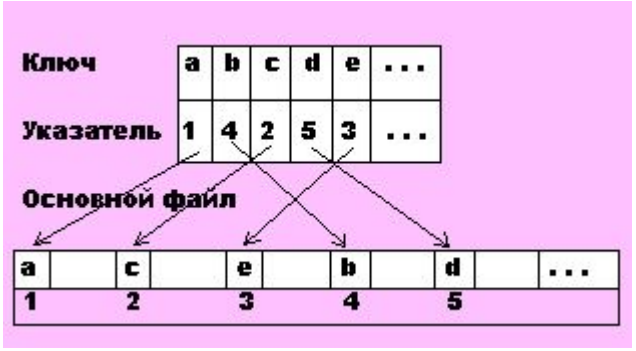
8.3. Индексные методы доступа

Основная идея состоит в следующем: в дополнение к основному файлу данных создается индексный файл, который содержит значения ключей и указатели на соответствующую запись. Индексный файл упорядочен по значению ключа. Обычно индексный файл меньше по размеру, чем основной, и скорость поиска в нем выше, так как он отсортирован.

ж) Метод доступа с полным индексом

Для каждой записи основного файла в индексном файле существует отдельная запись (ключ + адрес блока записей).

Индексный файл организован последовательно (метод а):

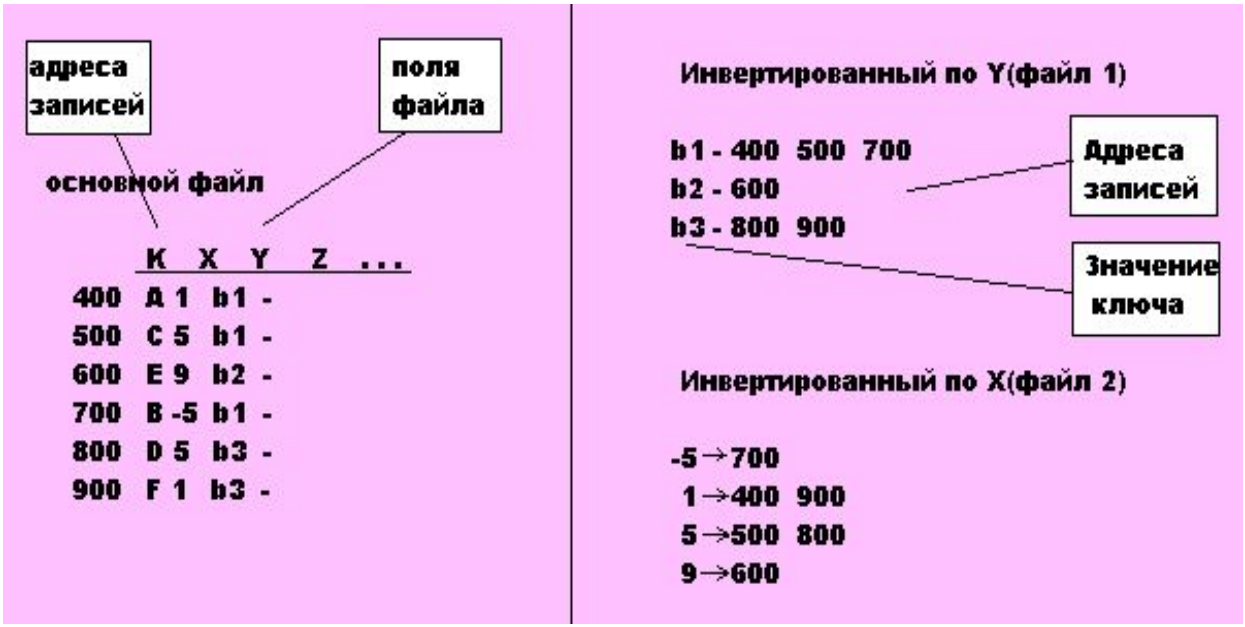


Такая организация используется, когда основной файл не очень велик. Если надо упорядочить исходный файл по нескольким ключам, для него создается столько же индексных файлов.

Когда индексный файл содержит вторичные ключи, он называется ***инвертированным*** по данному ключу по отношению к основному файлу.

Так как значения вторичных ключей не уникальны, то индексный файл организован по методу б) или в).

Пример:



Найдем записи, для которых X = 5 и Y = b1. По файлу 2 находим адреса для X=5 (500 800). По файлу 1 находим адреса для Y=b1 (400 500 700). Пересечение этих массивов дает результат: запись 500.

Основной эффект инвертированных массивов проявляется при поиске уникальной записи по нескольким условиям. Алгоритм поиска можно построить так, чтобы обращаться только к индексным файлам. Это ускоряет поиск, особенно если весь индексный файл помещается в оперативной памяти. Рассмотренный метод становится неэффективным, если требуется обработка всех записей.

з) Индексно - последовательный метод.

Основной файл всегда упорядочен по первичному ключу; индексный файл содержит ссылки не на каждую запись, а на группу (блок) записей и значение одного из ключей в группе - первого или последнего ключа. Повторение ключей в группе недопустимо. В группе должно быть одинаковое число записей. Блок обычно образуется как физический блок записей (дорожка МД, цилиндр МД, сектор МД и т.п.).

Таким образом, объем индексного файла уменьшается. Основной эффект достигается, когда основной файл на внешнем ЗУ, а индексный в ОП.

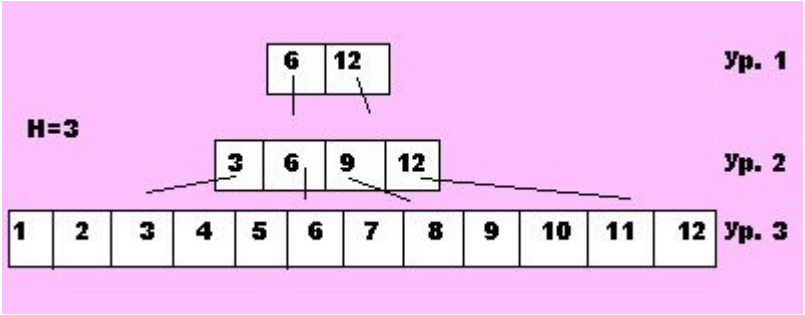
На рис. 8.1. приведен **пример** индексно-последовательной организации данных.

Так как индексный файл упорядочен по ключу, к нему можно построить еще один индексный файл и т. д., пока на верхнем уровне не окажется всего один блок. Такая многоуровневая структура называется ***бинарным деревом (B-деревом)***.

Достоинство B-дерева: меньший объем индексного файла, простота расширения индекса, высокая скорость поиска. На эффективность доступа и хранения влияет размер блока: чем больше размер блока, тем их меньше и меньше элементов индекса, но больше время перебора внутри блока.

Рис. 8.1

Пример. Построим бинарное дерево, содержащее ссылки на конец группы (будем изображать только значения ключа).



Поиск записи требует H обращений к индексному файлу, где H - число уровней индексов. Например, найдем запись с ключом К=8. Путь поиска: Ур1 - 12 (6<8<12), Ур2 - 9 (6<8<9), ур.3 - перебор двух ячеек (от 9 к 8).

8.4. Адресные методы доступа

В этих методах значение ключа К с помощью специальной адресной функции преобразуется в значение адреса записи. Адресные функции бывают двух видов:

- реализуют взаимно-однозначное соответствие А <-> К (прямой доступ к записи);
- реализуют однозначное соответствие К->А, но не наоборот (хеширование).

и) прямой доступ к записи

Здесь физический адрес записи непосредственно определяется из записи. **Например:** ключ файла “СТУДЕНТ” - НОМЕР ГРУППЫ + НОМЕР СТУДЕНТА В ГРУППЕ. Пусть Номер группы Nгр = 1...100, Номер_студента Nстуд = 1..25. Тогда Номер записи Nзап = (Nгр - 1) * 25 + Nстуд

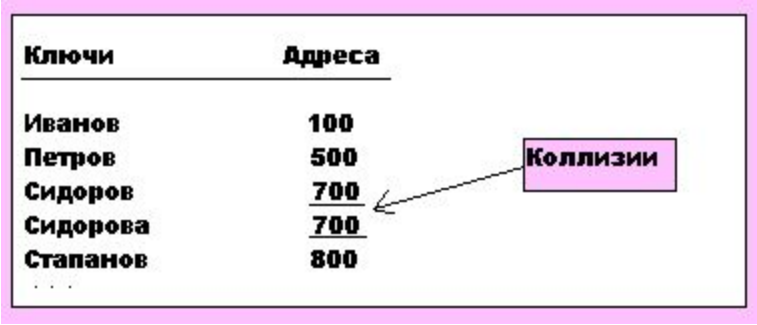
Недостатки метода:

- низкая эффективность хранения данных при нерациональной кодировке или низкой плотности значений ключа;
- обязательна уникальность ключа.

Достоинство: высокая скорость выборки и простота.

к) произвольный метод доступа (хеширование)

Хеширование в переводе означает “перемешивание”. **Суть метода** - вычисление адреса записи в соответствии с алгоритмом хеширования, который в общем случае для разных значений ключей может дать один адрес записи. Этот случай называется коллизией. Варианты данного метода доступа отличаются друг от друга видом хеш-функции и методом разрешения коллизий.



Хеш - функция должна равномерно отображать множество возможных значений ключа в множество адресов записей. Существуют различные хеш - функции. **Например,** остаток от деления ключа на число M : H(K) = K * mod(M) . В данном случае выбор M - решающий для числа коллизий. Рекомендуется выбирать в качестве M простое число, равное или близкое к размеру участка памяти.

Когда память не заполнена, коллизия возникает редко, но со временем память заполняется записями и вероятность коллизии возрастает. Есть **два способа разрешения коллизий:**

1. *Метод открытой адресации* - при появлении коллизии во время добавления записи ищется свободный участок памяти, куда и добавляется новая запись. При поиске записи поступают аналогично: сначала производят поиск по адресу, а затем - если запись с искомым ключом не найдена - последовательно просматриваются остальные участки памяти.

2. *Метод цепочек* - каждая запись имеет ссылку на область переполнения, где записаны цепочки коллизий (см. рис. 8.2).

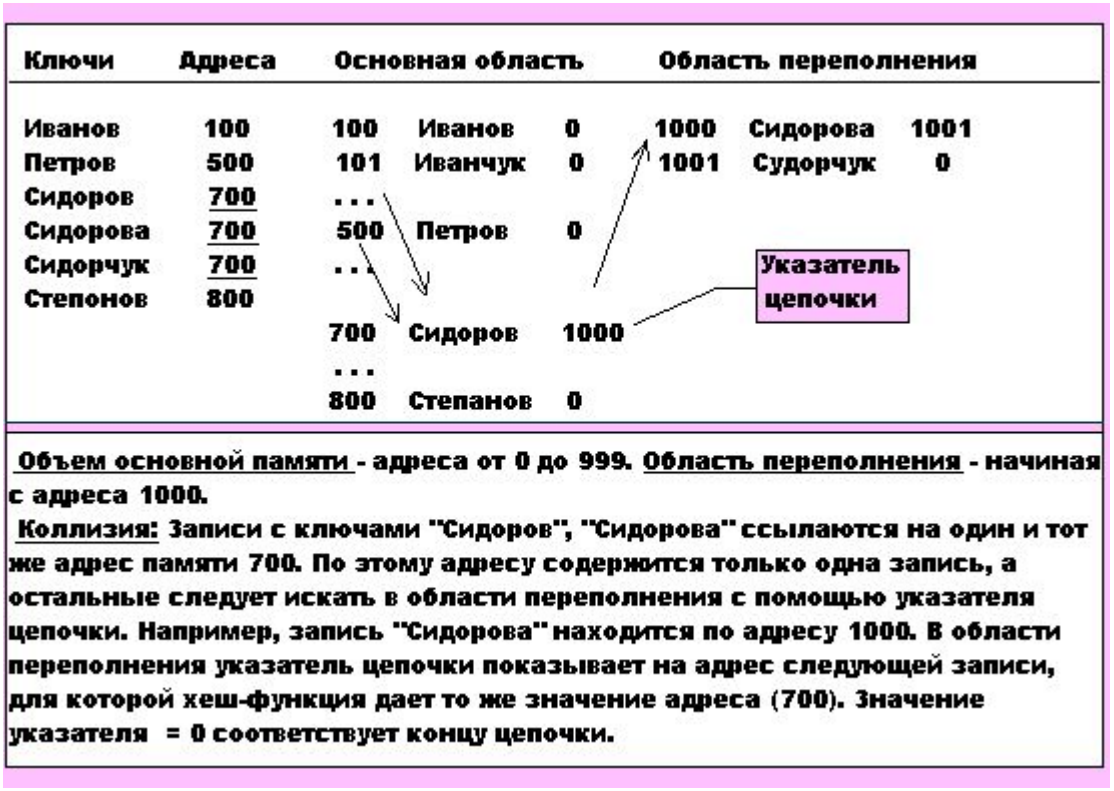


Рис. 8.2.

На эффективность данного метода влияют три фактора:

- распределение исходных ключей (чаще всего оно отличается от равномерного),
- распределение памяти (то есть объем памяти определяет вероятность коллизии),
- соответствие функций распределения ключей записей и адресов памяти.