ЛАБОРАТОРНАЯ РАБОТА № 1

МЕТОДЫ ФИЗИЧЕСКОЙ ОРГАНИЗАЦИИ ДАННЫХ. ХЕШИРОВАНИЕ

Размещаемые записи идентифицируются с помощью ключа — поля фиксированной длины, располагаемого в каждой записи в одной и той же позиции. Ключ д.б. уникальным.

Хеширование — один из методов, позволяющих по первичному ключу записи получить ее физический адрес.

Ключ записи преобразуется в квазислучайное число, которое используется для определения местоположения записи. Число может указывать на адрес по которому расположена запись или на область в которой расположена группа записей. Область, в которой расположена группа записей называется участком записей или пакетом записей.

При первоначальной загрузке, адрес, по которому д.б. размещена запись определяется следующим образом:

– Ключ записи преобразуется в квазислучайное число от 0 до N, где N — количество пакетов записей;

– Полученное число преобразуется в физический адрес пакета;

– Если в пакете есть свободное место, то запись располагается в нем;

– Если нет, размещается в области переполнения.

Факторами, влияющими на эффективность размещения являются:

– Размер пакета;

– Плотность заполнения (отношение количества записей в пакете к максимальной вместимости пакета);

– Алгоритм преобразования ключа в адрес;

– Организация области переполнения.

Алгоритм преобразования ключа в физический адрес пакета выполняется в 3 этапа:

1. Ключ преобразуется в цифровое представление;
2. Цифровое представление ключа преобразуется в совокупность произвольно-распределенных чисел по возможности равномерно в диапазоне допустимых адресов. Ключ преобразуется в адрес;
3. Адрес умножается на константу для размещения адресов в основной памяти, т.е. фактическое масштабирование адреса.

АЛГОРИТМЫ

1. Метод деления: ключ делится на простое число (или число не имеющее малых делителей) близкое по значению к числу пакетов N. Остаток от деления — относительный адрес пакета.
2. Метод средних квадратов: ключ возводится в квадрат, выбираются центральные цифры. Умножаются на константу.

3. Метод сдвига разрядов: числовое значение ключа делится на две части, младшая часть складывается со старшей. Описанный процесс продолжается до тех пор, пока количество цифр результата не окажется равным количеству цифр числа пакетов N. Результат — относительный адрес пакета.

4. Метод складывания: число разбивается на 3 части; центральная содержит столько же цифр, сколько цифр в числе пакетов N, первая и третья заворачиваются и складываются.

5. Метод преобразования системы счисления: преобразуется основание системы счисления. Число представляется в новой системе счисления. Последние цифры числа — относительный адрес пакета.

6. Метод анализа отдельных разрядов ключа: из числа вычеркиваются те разряды, которые имеют распределение сильно отличающееся от равномерного.

ОРГАНИЗАЦИЯ ОБЛАСТИ ПЕРЕПОЛНЕНИЯ

Цепочки участков переполнения

Для связи основой области памяти с областью переполнения используются цепочки адресов. Т.е. пакет в основной области хранит адрес пакета (записи) в области переполнения.

Метод распределенной области переполнения

Пакеты переполнения размещены через определенные интервалы среди первичных пакетов. Если первичный пакет переполнен, направленная в него запись направляется в ближайший пакет переполнения, следующий за данным первичным пакетом. Достоинства — пакеты переполнения располагаются в непосредственной близости к первичным пакетам — отпадает необходимость в частом перемещении головок чтения-записи дискового устройства.

Алгоритмы перемешивания (хеширования) осуществляют преобразование ключа к номеру первичного пакета. Однако алгоритм, преобразующий номер первичного пакета в его машинный адрес, должен учитывать наличие пакетов переполнения между первичными пакетами.

Если каждый 10-й пакет — пакет переполнения, то адрес текущего N-го пакета записей будет определяться по формуле:

Адрес пакета = $B_0 + B(N+[N/9])$, где

B_0 — адрес первого байта;

B — размер пакета в байтах;

N — номер пакета, полученный на выходе алгоритма хеширования

Метод открытой адресации (Петерсона)

Происходит рассеивание записей переполнения в основной области.

Если пакет переполнен, то запись помещается в следующий за ним пакет.

Данный метод уменьшает время поиска записей т.к. обычно смежные пакеты расположены друг за другом на расстоянии не превышающем длины дорожки. Однако если размер пакетов невелик, то это приводит к необходимости последовательного перебора пакетов в поисках свободного места. При размерах пакета более 10 записей этот метод эффективен при

периодической реорганизации БД. Позволяет увеличивать плотность заполнения.

Справочник свободных пакетов.

Организуется специальный справочник, который содержит сведения о том, какие пакеты еще не заполнены. Если первичный пакет заполнен, то с помощью справочника определяется свободный пакет и после занесения в него записи в первичный пакет заносится ссылка на него. Необходимость в справочнике возрастает в случае частой модификации файлов. В качестве справочника может использоваться простой список заполнения пакетов.

Метод эффективен, если размер пакета более 20 записей. Оптимизация предполагает загрузку файла, когда в основной области помещаются наиболее часто используемые записи.

ЗАДАНИЕ

Разместить данные с помощью методов хеширования, предложенных вариантом задания.

Количество записей – 150.

Плотность заполнения основной области не менее 50%.

Размер пакета записей выбрать самостоятельно. Аргументировать свой выбор.

Оценить:

– Эффективность алгоритмов размещения (быстроту размещения и поиска);

– Зависимость числа записей в области переполнения от плотности заполнения основной области и размера пакета записей.

Варианты заданий:

1. Ключ: 5 символов + 7 разрядное десятичное число. Методы хеширования: "метод средних квадратов", "метод складывания" Область переполнения в виде цепочек пакетов переполнения.
2. Ключ: число из 12 десятичных разрядов. Методы хеширования: "метод деления", "метод преобразования системы счисления". Распределенная область переполнения.
3. Ключ: символьный 12 разрядов. Методы хеширования: "сдвиг разрядов", "анализ отдельных разрядов ключа". Область переполнения в виде цепочек пакетов переполнения.
4. Ключ: 3 символа + 9 разрядное десятичное число. Методы хеширования: "метод деления", "метод складывания". Распределенная область переполнения.
5. Ключ: десятичное число 12 разрядов. Методы хеширования: "метод деления", "анализ отдельных разрядов ключа". Область переполнения в виде цепочек пакетов переполнения.

ЛАБОРАТОРНАЯ РАБОТА № 2
МЕТОДЫ ФИЗИЧЕСКОЙ ОРГАНИЗАЦИИ ДАННЫХ
УПОРЯДОЧЕННОЕ РАЗМЕЩЕНИЕ И СОРТИРОВКА

В стандартной реализации записи упорядочены по возрастанию значений ключа.

МЕТОДЫ СОРТИРОВКИ КЛЮЧЕЙ:

Прямой выбор. Выбирается элемент с наименьшим ключом. Он меняется местами с первым элементом. Затем этот процесс повторяется с оставшимися n-1 элементами, n-2 элементами и т. д.

Метод пузырька. Алгоритм основывается на сравнении и смене мест для пары соседних элементов и продолжении этого процесса до тех пор, пока не будут упорядочены все элементы.

Быстрая сортировка. Выбирается какой-нибудь (например средний) элемент и все элементы переставляются так, чтобы слева от выбранного оказались только элементы с меньшим значением ключа, а справа с большим значением ключа (тем самым выбранный элемент окажется на своем окончательном месте), после чего необходимо рекурсивно применить этот метод к левой и правой частям списка.

Карманная сортировка. Сортируются ключи, содержащие более одной цифры. Выделяются карманы. Число карманов соответствует числу существующих значений каждого разряда ключа. Ключи последовательно сортируются по каждому разряду. Сначала происходит сортировка по младшему разряду. После этого содержимое разных карманов объединяется так, чтобы элементы из кармана с меньшим весом были слева, а карманы с большим — справа. Далее происходит сортировка по карманам предпоследнего разряда ключа с последующим объединением карманов и т.д.

Пример: Дана последовательность ключей

42 23 74 11 65 57 94 36 99 87 70 81 61

карманы:	0	1	2	3	4	5	6	7	8	9
	70	11	42	23	74	65	36	57		99
		81			94			87		
		61								

После объединения карманов:

70 11 81 61 42 23 74 94 65 36 57 87 99

Следующий шаг:

0	1	2	3	4	5	6	7	8	9
	11	23	36	42	57	61	70	81	94
						65	74	87	99

Объединяя карманы, заканчиваем сортировку:

11 23 36 42 57 61 65 70 74 81 87 94 99

Каждый карман рациональнее всего представлять в виде динамической очереди.

Сортировка слиянием. Слияние означает объединение двух (или более) последовательностей в одну единственную упорядоченную последовательность с помощью повторяющегося выбора из доступных в данный момент элементов. Сортировка выполняется путем многократного выполнения попарных слияний. Например 16 последовательностей в результате попарного слияния образуют 8 последовательностей. Которые после попарного слияния образуют 4 последовательности и т.д.

Для слияния $2^к$ последовательностей требуется к шагов. Каждую последовательность, содержащую n записей можно реализовать как состоящую из n последовательностей по 1-й записи. Расчленение исходной последовательности на подпоследовательности можно выполнить с учетом внутренних упорядоченностей.

Недостатком является большой объем памяти, требуемый для реализации алгоритма.

Сортировка с помощью бинарного дерева. Алгоритм состоит из 2-х этапов:

1. Построение дерева обеспечивает вставку новых записей
2. Обход дерева

В результате будет сформирована упорядоченная последовательность

СПОСОБЫ ПОИСКА

Говоря о файлах данных, в которых записи размещены последовательно в порядке возрастания первичного ключа нельзя забывать о проблемах, связанных с добавлением и удалением записей в уже созданном файле. В то же время индексный файл проще файла данных — размер фиксирован, всегда упорядочен. Это упрощает поиск информации. Оценивая быстродействие поиска говорят о количестве проведенных испытаний — числе просмотренных записей до обнаружения требуемой. Приведенные ниже способы поиска эффективны лишь для индексных файлов или файлов, хранящих архивные данные.

Блочный поиск. Индексы разбиваются на блоки. Поиск начинается с блока, содержащего элементы с наименьшими значениями. Проверяется наибольшее значение элемента в этом блоке. Если это значение меньше требуемого, то осуществляется переход к следующему блоку. В противном случае производится поиск элемента тем же способом в текущем блоке. Рекомендуемый размер блока $\sqrt{N_i}$, где N_i — число элементов в индексном файле.

Метод применяется при небольших размерах индекса.

Число испытаний, в среднем, $\sqrt{N_i}$.

Метод двоичного деления (двоичный поиск). Рассматривается запись, находящаяся в середине области и ее ключ сравнивается с ключом искомой записи. Затем определяется подобласть в которой будет продолжаться поиск.

Эта подобласть опять делится пополам и процесс повторяется пока не будет найдена требуемая запись. Среднее число испытаний $\log_2 N_i$-1. Применяется для быстрого поиска в основной памяти. Двоичный поиск более пригоден для поиска в индексе файла, чем в самом файле. Недостатки: невозможно сжатие индекса. Сложен процесс пополнения индекса.

Поиск по двоичному дереву. Элементы, среди которых ведется поиск, можно организовать в индексе в виде двоичного дерева, вершины которого связаны указателями. Тогда добавление новых вершин не потребует реорганизации старых. Применяется в случае частых изменений в индексном файле. Но необходим расход памяти на два указателя. Невозможно сжатие данных.

Сбалансированное индексное дерево. В следствии построения сбалансированного дерева, поиск требует меньшего числа испытаний, чем в предыдущих методах. Сокращается число обращений к внешней памяти. Важно выбрать оптимальное число уровней и оптимальное число элементов в одном узле.

Несбалансированное дерево. Ускоряет поиск часто используемых записей, но замедляет поиск редко используемых. Тем самым общее время поиска за отдельный период времени уменьшается. Однако сложна процедура ведения файла.

Вычисление оценки местоположения. Поиск упростится, если известно примерное положение элемента в индексе. Эту оценку можно вычислить, например, в упорядоченном файле, с помощью смещения первого и второго символа по отношению к началу индекса. Результат —таблица процентных отношений местоположения 2-х первых символов индекса.

ЗАДАНИЕ

I. Отсортировать два файла, содержащие последовательности ключей. Длина каждой последовательности не менее 1000 элементов. 1-я последовательность содержит 16-ричные ключи (8 разрядов); 2-я — символьные (6 символов). Оценить быстродействие двух предложенных методов сортировки ключей и объем памяти, требуемой для сортировки.

1.Прямой выбор. Быстрая сортировка.
2.Прямой выбор. Карманная сортировка.
3.Метод пузырька. Бинарное дерево.
4.Метод пузырька. Сортировка слиянием.
5.Быстрая сортировка. Карманная сортировка.
6.Бинарное дерево. Быстрая сортировка.
7.Метод пузырька. Карманная сортировка.
8.Прямой выбор. Бинарное дерево.
9.Карманная сортировка. Сортировка слиянием.

II. Произвести поиск элементов в индексном файле или в файле записей двумя предложенными способами.

1.Блочный поиск. Метод двоичного деления.
2.Блочный поиск. Оценка местоположения.
3.Поиск по двоичному дереву. Метод двоичного деления.
4.Сбалансированное индексное дерево. Оценка местоположения.
5.Блочный поиск. Несбалансированное индексное дерево.
6. Несбалансированное индексное дерево. Оценка местоположения.
7.Несбалансированное индексное дерево. Метод двоичного деления.
8.Двоичное дерево. Оценка местоположения.
9. Сбалансированное индексное дерево. Метод двоичного деления.

III. Оценить среднее время поиска одной записи. (Время построения дерева не учитывать)

ЛАБОРАТОРНАЯ РАБОТА № 3

МЕТОДЫ ФИЗИЧЕСКОЙ ОРГАНИЗАЦИИ ДАННЫХ. СЖАТИЕ ДАННЫХ

Наиболее эффективны методы сжатия для хранения архивных файлов с невысокой частотой использования а так же в системах передачи данных.
Методы сжатия можно разделить на 2 класса:

– Методы, ориентированные на конкретную структуру или содержимое записей и разработанные для конкретного приложения;

– Методы, которые используются во многих приложениях и м.б. встроены в общее программное обеспечение.

Работу по сокращению размеров файла можно начать с методов, ориентированных на содержимое записей и затем продолжить с помощью универсальных методов кодирования:

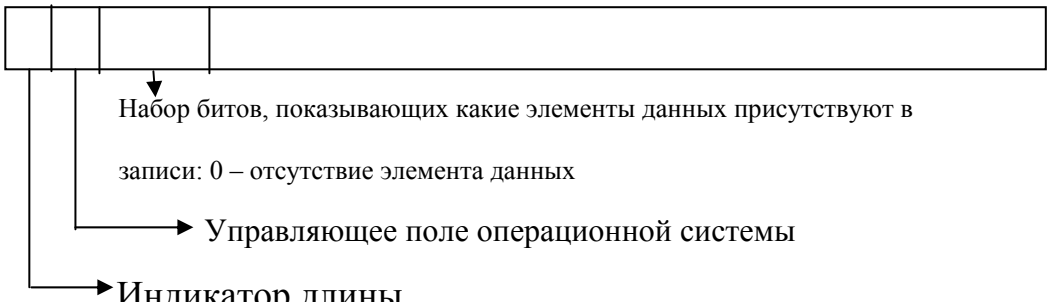
– Исключение избыточных элементов в файле. Одни и те же элементы данных могут хранится в составе нескольких файлов.

– Переход от естественных обозначений к более компактным, например для записи даты кодировать месяц достаточно 4-мя битами, день — 5-ю и т.д.

– Подавление повторяющихся символов. Например: числовые поля в некоторых файлах нередко содержат старшие и младшие 0, если их число больше 2-х, то его всегда можно свести к 2-м (упакованным десятичным) символам. Один— признак повторяемости. Второй — число повторений.

– Ликвидация пустых мест в файле (для записей переменной длины). Атрибуты, связанные с одним описываемым объектом, могут иметь несколько значений. Некоторые атрибуты со временем сами становятся объектами с собственным набором атрибутов. Если выделить место для хранения некоторого оптимального количества значений атрибутов, то это приведет к возникновению пустых мест в файлах. 1-й вариант ликвидации пустых мест сводится к тому, что можно рассматривать запись, как запись с фиксированным числом элементов, но с возможным отсутствием некоторых из них. Записи представляются в определенном формате: перед элементами данных

расположена совокупность битов показывающих, какие элементы данных отсутствуют или присутствуют.



Второй вариант ликвидации пустых мест в файле — битовая матрица.

– Кодирование часто используемых элементов данных. Если атрибут обладает ограниченным набором значений, то можно не выписывать эти значения целиком, а использовать вместо этого специальные коды. Например атрибут "Имя" можно закодировать 8-ю битами: 1-й бит — признак пола, остальными 7-ю 128 мужских + 128 женских имен.

– Сжатие упорядоченных данных. В упорядоченном наборе данных начальные символы часто совпадают. Эти элементы данных можно сократить введением чисел вместо повторяющихся символов и последующим кодированием наиболее распространенных имен. Если список используется как индекс или справочник, то возможно дальнейшее уплотнение. В частности достаточно сохранить первые 4 буквы фамилии и 2 буквы инициалов.

– Коды переменной длины (коды Хафмана). Коды с переменным числом битов на символ позволяют достичь более плотной упаковки данных. В этих кодах часто используемые символы кодируются короткими кодами, а символы с низкой частотой использования — длинными кодами.

ЗАДАНИЕ

Сжать файл данных предложенным преподавателем методом. Сравнить размер памяти занимаемой файлом до сжатия и после него. Разархивировать сжатый файл и убедиться в том, что он идентичен исходному.

ЛАБОРАТОРНАЯ РАБОТА №4

МЕТОДЫ ФИЗИЧЕСКОЙ ОРГАНИЗАЦИИ ДАННЫХ

Наиболее общий метод упорядочения записей — размещение записей в соответствии с возрастанием их ключей. Если ключи следуют не в строго определенном порядке, то прямая адресация затруднительна и для выполнения поиска какой-либо записи требуется специальный индекс, чтобы не просматривать весь файл.

Для работы с индексно-последовательным файлом можно использовать два режима обработки:

1. Последовательная обработка, при которой записи обрабатываются в последовательности их размещения на ВЗУ.
2. Произвольная обработка, при которой записи обрабатываются в произвольной последовательности, не связанной с физической организацией записей на ВЗУ.

СВЯЗЬ С ТЕХНИЧЕСКИМ ОБЕСПЕЧЕНИЕМ

Индексно-последовательная организация может быть разработана таким образом, чтобы в наибольшей степени соответствовать техническому обеспечению системы. Достоинствами такого способа являются уменьшение времени доступа к записям и экономия памяти. Но имеется и существенный недостаток привязки к конкретному типу оборудования. Проблема возникает при перемещении (в случае необходимости) файла на ЗУ другого типа. Исходя из вышесказанного существуют две разновидности организации индексно-последовательных файлов:

1. ISAM — индексно-последовательный метод доступа.
2. VSAM — виртуальный метод доступа.

ИНДЕКСНО-ПОСЛЕДОВАТЕЛЬНЫЙ МЕТОД ДОСТУПА (ISAM)

ISAM — записи группируются так, чтобы могли располагаться на отдельных дорожках цилиндров модуля дисков, при этом одна дорожка на каждом цилиндре отводится для индексов, указывающих на записи, расположенные на данном цилиндре.

Файл состоит из:

- Основной области;
- Индексной области;
- Области переполнения.

При создании файла записи помещаются в основной области последовательно и упорядочены по значениям ключей. Индексная область может быть организована в ОП или на внешнем устройстве. Для доступа к данным можно использовать несколько уровней индексов. Самый нижний — индекс дорожки —размещается на 1-й дорожке цилиндра. Для каждой записи основной области индекс дорожки содержит:

– Нормальный элемент, который состоит из адреса дорожки и самого большого из ключей записей, расположенных на этой дорожке.

– Элемент переполнения, хранящий ссылку на последний элемент, который был вытолкнут в область переполнения.

Индекс цилиндров — ускоряет поиск нужного цилиндра на котором размещена запись, аналогичен индексу дорожек, хранится в ОП или отдельным файлом на ВЗУ. Заполнение записями основной области начинается со 2-й дорожки *n*-го цилиндра. После заполнения которого процесс размещения

записей продолжается начиная со второй дорожки *n*+1-го цилиндра и т.д. до окончания создания файла. При реорганизации выполняется полная перезапись.

Область переполнения может располагаться на одной или нескольких дорожках того же цилиндра, на котором находятся первоначальные данные и в этом случае доступ к записям ускоряется — нет необходимости перемещения головок чтения. С другой стороны, допускается расположение области переполнения за пределами данного цилиндра — независимая область переполнения.

При поиске какой-либо конкретной записи необходимо вначале определить на каком цилиндре она находится. Для этого последовательно просматриваются все индексы цилиндров пока не находится 1-й, содержащий ключ, превышающий ключ искомой записи. Затем та же операция производится над индексами дорожек в найденном цилиндре. И, наконец, считывается с дорожки необходимая запись.

Если необходимо включить в файл новую запись, а файл уже создан, то эта запись включается на то место, где она должна располагаться в порядке возрастания ключа. Если это место уже занимает другая запись ее необходимо передвинуть вправо. В случае, когда сдвигаемая запись не вмещается на дорожке ее необходимо поместить в область переполнения.

При обращении к записям, попавшим в область переполнения требуется дополнительная операция чтения. Для повышения эффективности работы необходима периодическая реорганизация файла. Записи файла заново сортируются и переразмещаются, оставляя свободными области переполнения. Затем производятся соответствующие изменения в индексе. Такая процедура называется процедурой ведения файла.

ВИРТУАЛЬНЫЙ МЕТОД ДОСТУПА (VSAM)

В данном методе используются управляемые области, подразделяющиеся на управляемые интервалы. Управляемые интервалы, относящиеся к одному набору данных, имеют одинаковую длину. Выбор длин возможен в широких пределах. Несколько управляемых интервалов могут располагаться на одной дорожке, в тоже время один управляемый интервал может занимать несколько дорожек.

В методе доступа VSAM используется индекс управляемых интервалов, который называется набором указателей – на одну управляемую область имеется один набор указателей.

Набор указателей сам индексируется с помощью иерархии(дерева) указателей. Эта иерархия указателей носит название набора индексов. Набор указателей содержит максимальное значение ключа для каждого управляемого интервала и указывает на данный управляемый интервал. Высший уровень набора индексов содержит максимальное значение ключа для каждой управляемой области и указатель на блок набора указателей для данной управляемой области.

При создании файла осуществляется включение записей в управляемые интервалы, при этом их не заполняют до предела, что приводит к образованию распределенной свободной памяти в каждом из интервалов. Кроме этого, внутри управляемой области некоторые управляемые интервалы остаются свободными. Каждый элемент набора указателей указывает на один управляемый интервал. Если управляемый интервал полностью свободен, то элемент набора вместо максимального значения ключа записей, расположенных в интервале, содержит признак свободной памяти.

Наличие значительного объема свободной памяти в каждом управляемом интервале приводит к тому, что большая часть вновь поступающих записей умещается в пределах соответствующих интервалов. Тем не менее неизбежны случаи нехватки распределенной свободной памяти в интервалах для включения новых записей. В таких случаях осуществляется “расщепление” интервала. Предположим, что необходимо включить запись с некоторым значением ключевого поля. В соответствии со значением ключевого поля определяется управляемый интервал, в который следует включить запись. Но интервал полностью заполнен, и поэтому осуществляется его расщепление, заключающееся в том, что около половины его записей пересылается в свободный интервал, входящий в состав той же управляемой области. Программы VSAM находят свободный управляемый интервал, для которого соответствующий элемент в наборе указателей помечен как свободный.

Но управляемая область может быть также заполненной(т.е. в ее составе нет интервалов, помеченных в наборе указателей как свободные). В таком случае осуществляется расщепление управляемой области. В конце набора данных программы VSAM выделяют новую область, в которую переписывается около половины интервалов из расщепляемой области. Место для новых областей может быть выделено заблаговременно, или новая область создается путем добавления дополнительного экстента к наборам данных. Расщепление интервалов и областей сопровождается соответствующими изменениями в индексах.

Необходимо отметить, что процесс расщепления свободной памяти может быть организован таким образом, что отпадает необходимость выполнения процедур ведения файла, так как расщепление фактически обеспечивает выполнение функции ведения.

Следует отметить, что в любом случае (до или после расщепления интервала) записи в пределах управляемого интервала расположены в порядке, соответствующем возрастанию их ключей. Однако в масштабе управляемой области возрастание ключей может быть нарушено. Тем не менее, адресация в блоке индексов организована таким образом, что позволяет осуществлять доступ к записям, содержащимся в области, в порядке возрастания значений ключей. После расщепления управляемой области интервалы в образованных областях остаются в той же последовательности, что и до расщепления. Многократные расщепления могут нарушить последовательность управляемых областей, тем не менее, доступ к записям в соответствии с последовательностью ключей возможен благодаря наличию набора индексов.

Набор индексов может иметь многоуровневую структуру. Индексы, расположенные на высшем уровне этой структуры, содержат указатели, позволяющие осуществлять доступ к управляемым областям в соответствии с возрастанием ключей.

Расщепление областей приводит лишь к незначительному росту времени последовательного доступа к записям файла. Блоки наборов указателей, каждый из которых относится к одной управляемой области, образуют горизонтальную цепочку. Независимо от последовательности физического расположения управляемых областей блоки наборов указателей будут связаны в цепочки в соответствии с возрастанием значений ключей записей в интервалах. Таким образом, последовательная обработка записей не требует обращения к набору индексов, поскольку достаточно использования наборов указателей интервалов. Набор индексов применяется лишь при прямом доступе к записям файла.

Таким образом, в методе доступа VSAM не требуется периодического выполнения процедур ведения файла.

Задание

Произвести размещение последовательности из 1000 записей методом VSAM. Размещение производить в соответствии со следующим алгоритмом
Шаг 1: Сгенерировать последовательность из 1000 записей
Шаг 2: Отсортировать первые 800 записей
Шаг 3: Произвести размещение отсортированных записей по методу VSAM в структуры согласно варианту, заданному преподавателем
Шаг 4: Произвести поэлементное добавление оставшихся 200 записей
Предусмотреть возможность визуального просмотра процесса добавления записей

Вариант 1

Управляемая область состоит из 8-ми управляемых интервалов. При начальном размещении каждый 4-й интервал – свободный. В каждом интервале при начальном размещении зарезервировать (оставить свободным) ≈10% общей размерности управляемого интервала. Создать 3 уровня индексов: главный индекс, индекс управляемых областей, индекс управляемых интервалов.

Вариант 2

Управляемая область состоит из 15-ти управляемых интервалов. При начальном размещении каждый 5-й интервал – свободный. В каждом интервале при начальном размещении зарезервировать (оставить свободным) ≈20%

общей размерности управляемого интервала. Создать 2 уровня индексов: индекс управляемых областей, индекс управляемых интервалов.

Вариант 3

Управляемая область состоит из 12-ти управляемых интервалов. При начальном размещении каждый 4-й интервал – свободный. В каждом интервале при начальном размещении зарезервировать (оставить свободным) ≈20% общей размерности управляемого интервала. Создать 2 уровня индексов: индекс управляемых областей, индекс управляемых интервалов.

Вариант 4

Управляемая область состоит из 9-ти управляемых интервалов. При начальном размещении каждый 3-й интервал – свободный. В каждом интервале при начальном размещении зарезервировать (оставить свободным) ≈20% общей размерности управляемого интервала. Создать 3 уровня индексов: главный индекс, индекс управляемых областей, индекс управляемых интервалов.