Лекция 10

Физическая организация базы данных

Физическая организация базы данных

Пользователи стандартных СУБД обычно не проводят проектирование физической БД. Однако в большеразмерных и распределенных СУБД (например, Oracle) ведется распределение областей памяти. В силу этого знание характеристик физического расположения данных полезно.

К физической модели предъявляются два основных противоречивых требования:
1) высокая скорость доступа к данным;
2) простота обновления данных.

Хранение на диске

Существует два принципиальных подхода к хранению таблиц.

1. построчное хранение,
2. колоночное хранение.

Основной единицей хранения данных на диске являются страницы, и мы стараемся располагать строки таблицы так, чтобы каждая строка целиком размещалась в одной странице. Страница — это очень небольшое количество данных, поэтому они объединяются в более крупные единицы, называемые экстентами.

Хранение на диске

Выделяют три основных режима работы приложений, связанных с использованием баз данных.

Режим 1. Получить все данные (последовательная обработка).

Режим 2. Получить уникальные (например, одна запись) данные, для чего используют произвольный доступ (хеширование, идентификаторы), индексный метод (первичный ключ), произвольный доступ, последовательный доступ (бинарное B-дерево, B+-дерево).

Режим 3. Получить некоторые (группу записей) данные, для чего применяют вторичные ключи, мультисписок, инвертированный метод, двусвязное дерево.

Физический поиск в БД

сколько страниц нам нужно считать, чтобы добраться до нашей записи.

1. Первый способ организации таблицы — это хранение файлов, так называемое «файлы в виде кучи» или хранение таблицы с добавлением в конец. Среднее количество блоков, которое нам придется посмотреть, это N / 2
2. Второй способ организации таблицы : если мы будем хранить наши данные, отсортированными по какому-то значению ключа. В таком случае у нас, конечно, сложнее будут происходить операции вставки и изменений, потому что нам нужно будет поддерживать порядок записи внутри файла. Зато мы выиграем при такой организации в поиске, потому что наш поиск сократится до двоичного логарифма от количества блоков, используемых для хранения записи в таблице.

Индексы

но двоичный логарифм от количества блоков— это все-таки еще очень большое число. Поэтому могут понадобится какие-то дополнительные структуры, которые позволят производить поиск быстрее.

Такие структуры называются индексами. В общем и целом, можно дать определение индексу, назвав его избыточной структурой, которая служит для ускорения поиска.

никакой новой содержательной информации в данные он не добавляет. Он лишь помогает нам найти нужные данные быстрее.

Индексы

Назначение индексов: это ускорение доступа к данным; это автоматическое упорядочивание записей при выборке; и также с помощью индексов мы можем добиться поддержки уникальности данных.

Индексы создаются автоматически, когда мы определяем свойства некоторых столбцов таблицы, или мы можем создавать индексы специальной командой.

Организация хранения и доступа

Основными методами хранения и поиска являются физически последовательный, прямой, индексно-последовательный и индексно-произвольный.

1) последовательный метод доступа
2) прямой метод
3) индексно-последовательный метод
4) индексно-произвольный метод

Для их сравнительной оценки используем два критерия .

Эффективность хранения - величина, обратная среднему числу байтов вторичной памяти, необходимому для хранения одного байта исходной памяти.

Эффективность доступа - величина, обратная среднему числу физических обращений, необходимых для осуществления логического доступа.

Физически последовательный метод

Физически последовательный метод

Записи хранятся в логической последовательности, файл имеет постоянный размер, указатели могут отсутствовать. Данные хранятся в главном файле, а обновление требует создания нового главного файла с упорядочением, для чего используется вспомогательный файл. Эффективность использования памяти близка к ста процентам, эффективность доступа низка.
Метод удобен для режима 1, однако быстродействие в режиме 2 мало: Время включения и удаления записей значительно.

Режим 1. Получить все данные (последовательная обработка).

Режим 2. Получить уникальные (например, одна запись) данные, для чего используют произвольный доступ (хеширование, идентификаторы), индексный метод (первичный ключ), произвольный доступ, последовательный доступ (бинарное B-дерево, B+-дерево).

Индексно - последовательный метод

Индексно-последовательный метод.

Индексный файл упорядочен по первичному ключу (главному атрибуту физической записи). Индекс содержит ссылки не на каждую запись, а на группу записей.

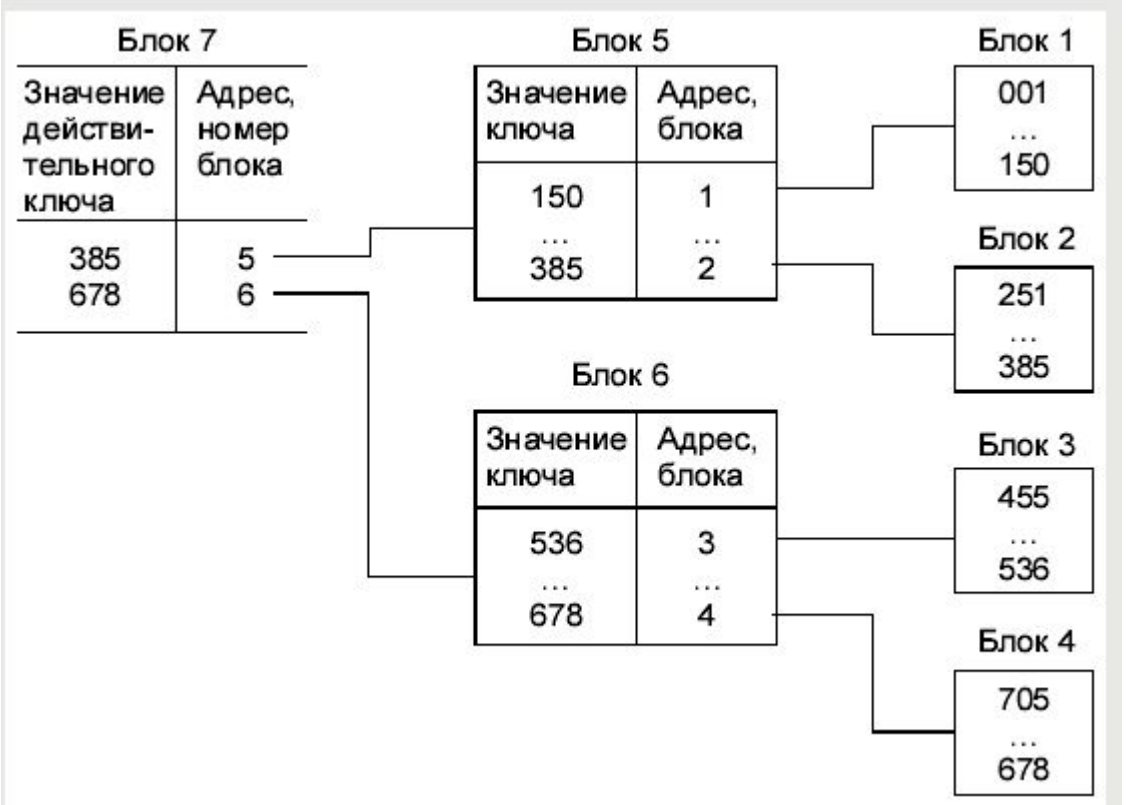
Индексно - последовательный метод

Процедура добавления возможна в двух видах.

1. Новая запись запоминается в отдельном файле (области), называемом областью переполнения. Блок этой области связывается в цепочку с блоком, которому логически принадлежит запись. Запись вводится в основной файл.
2. Если места в блоке основного файла нет, запись делится пополам и в индексном файле создается новый блок.

Индексно - последовательный метод

Последовательная организация индексного файла допускает, в свою очередь, его индексацию - многоуровневая индексация

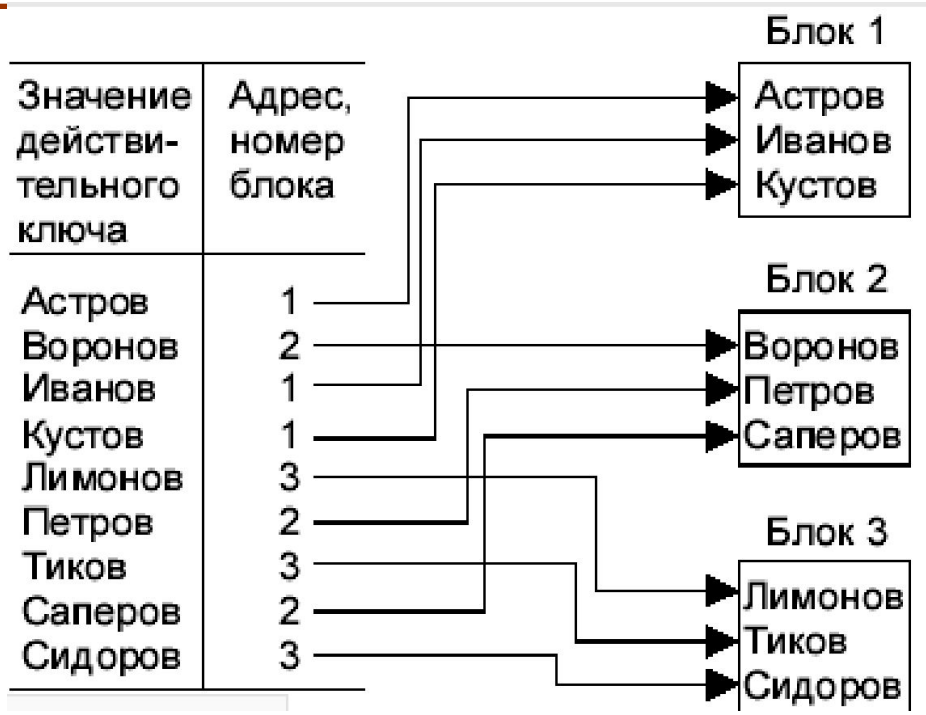


Индексно - последовательный метод

Наличие индексного файла большого размера снижает эффективность доступа. В большой БД основным параметром становится скорость выборки индексов. При большой интенсивности обновления данных следует периодически проводить реорганизацию БД. Эффективность хранения зависит от размера и изменяемости БД, а эффективность доступа - от числа уровней индексации, распределения памяти для хранения индекса, числа записей в БД, уровня переполнения.

Индексно - произвольный метод

Индексно-произвольный метод. Записи хранятся в произвольном порядке. Создается отдельный файл, хранящий значение действительного ключа и физического адреса (индекса). Каждой записи соответствует индекс..



Разновидностью этого метода является наличие плотного индекса: кроме главного файла создается вспомогательный, называемый плотным индексом. Он состоит из записи (v, p) для любого значения ключа v в главном файле, где p - указатель на запись главного файла со значением ключа v.

Индексно - произвольный метод

При операции добавления осуществляется запись в конец основной области. В индексной области необходимо произвести занесение информации в конкретное место, чтобы не нарушать упорядоченности.

N записи		
1	10–44/01	Иванов
2	11–44/01	Степанов
3	20–44/02	Кустов
4	45–32/01	Чернов
5	06–40/00	Трусов
6	05–40/00	Терпухов
7	01–30/01	Павлов
8	02–40/02	Мухина
9	06–50/01	Артемьев
10	07–35/00	Васильев
11	15–40/01	Васильева
12	17–40/02	Уваров
13	48–40/03	Шишкин
14	50–44/00	Удалов

Основная область

	Ключ	Ссылка на номер записи	
Блок 1	01–30/01	7	
	02–40/02	8	
	05–40/00	5	
			Свободное пространство
Блок 2	06–40/00	6	
	06–50/01	9	
	07–35/00	10	
			Свободное пространство
Блок 3	10–44/01	1	
	11–44/01	2	
	15–40/01	11	
Блок 4	17–40/02	12	
	20–44/02	3	
	45–32/01	4	
Блок 5	48–40/03	13	
	50–44/00	14	

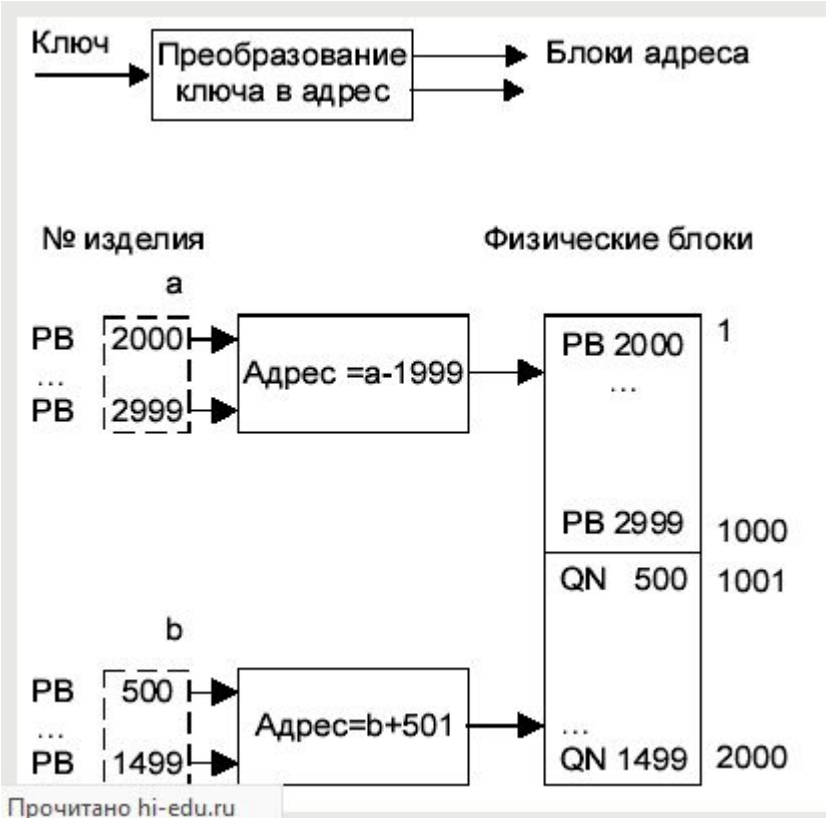
Индексная область

Поэтому вся индексная область файла разбивается на блоки и при начальном заполнении в каждом блоке остается свободная область (процент расширения)

Прямой метод

Прямой метод. Имеется взаимно-однозначное соответствие между ключом записи и ее физическим адресом.

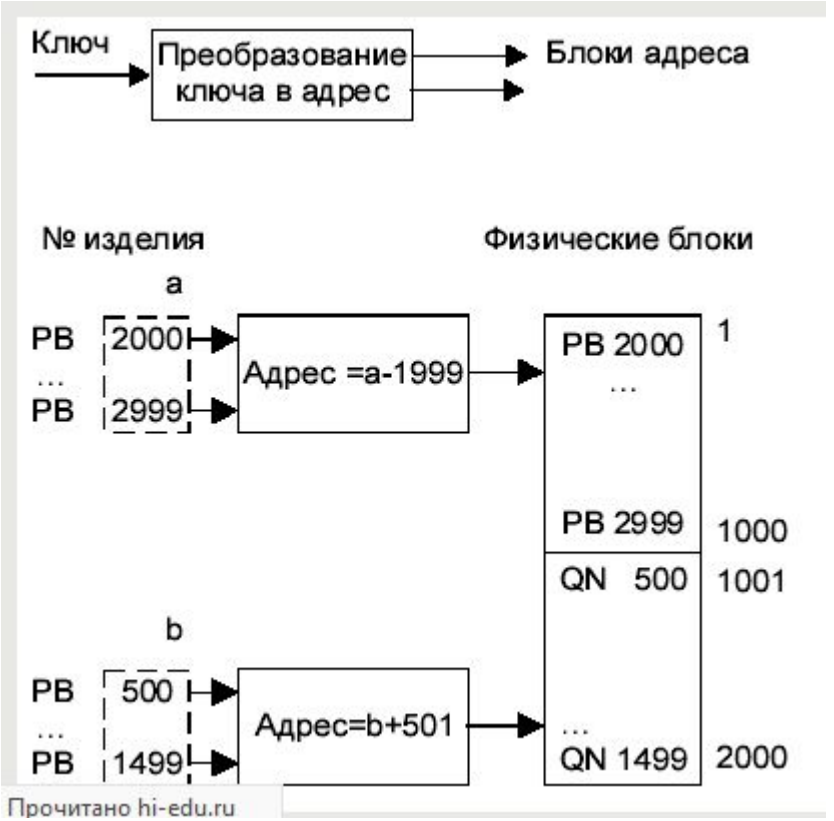
В этом методе необходимо преобразование и надо четко знать исходные данные и организацию памяти.
Ключи должны быть уникальными.



Прямой метод

Эффективность доступа равна 1, а эффективность хранения зависит от плотности ключей.

Если не требовать взаимно-однозначности, то получается разновидность метода с использованием хеширования - быстрого поиска данных, особенно при добавлении элементов непредсказуемым образом.



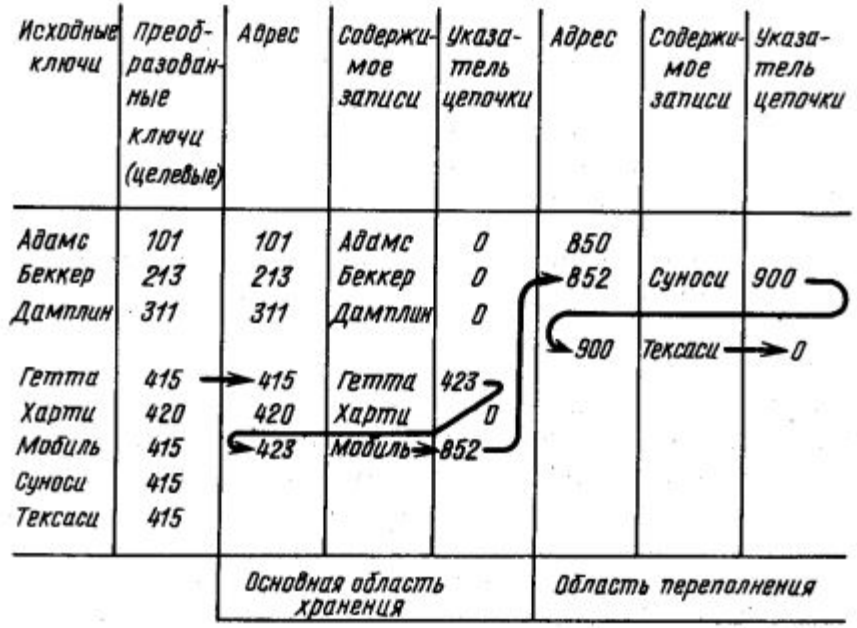
метод с использованием хеширования

Хеширование - метод доступа, обеспечивающий прямую адресацию данных путем преобразования значений ключа в относительный или абсолютный физический адрес.
Как и прямой метод доступа, метод доступа посредством хеширования основан на алгоритмическом определении адреса физической записи по значению ее ключа.

Различие состоит в том, что при прямом методе доступа имеет место взаимно однозначное отображение ключа в адрес, а метод доступа посредством хеширования допускает возможность отображения многих ключей в один адрес (т. е. один и тот же адрес или блок может быть идентифицирован более чем одним значением ключа).

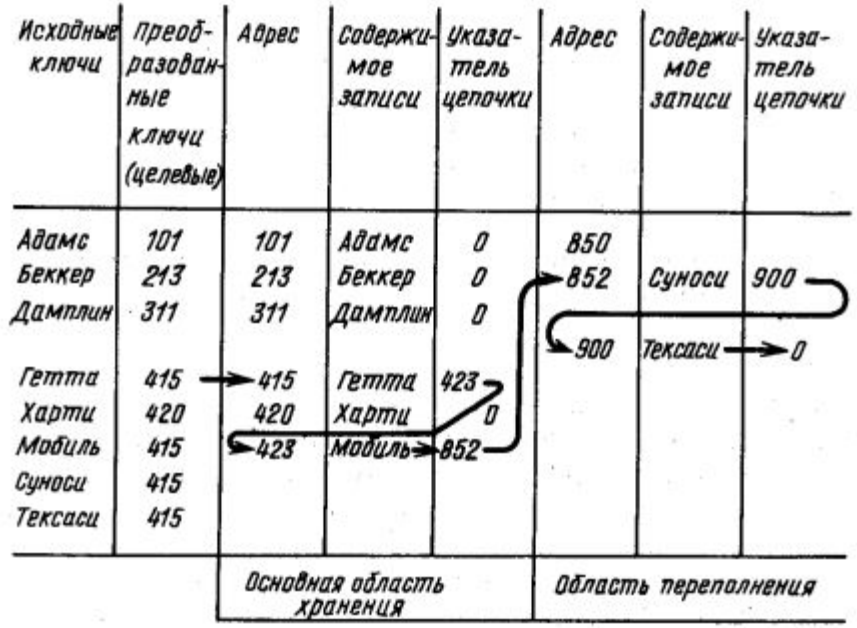
метод с использованием хеширования

Алгоритм преобразования ключа в адрес часто называют подпрограммой рандомизации или подпрограммой хеширования. Одинаковые ключи подпрограмма преобразует в одинаковые адреса: подпрограмма рандомизации отображает набор значений исходных ключей в набор адресов, причем число ключей превышает число адресов.



метод с использованием хеширования

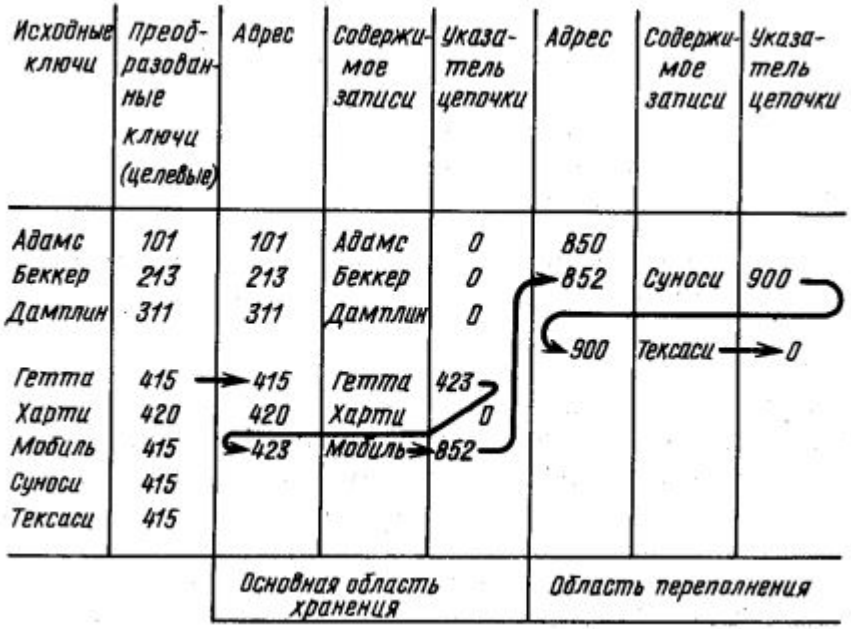
Записи, ключи которых отображаются в
один и тот же физический адрес, называются синонимами. Поскольку по адресу, определяемому подпрограммой рандомизации, может храниться только одна запись, синонимы должны храниться в каких-нибудь других ячейках памяти. В то же время необходимо наличие механизма поиска этих синонимов.



метод с использованием хеширования

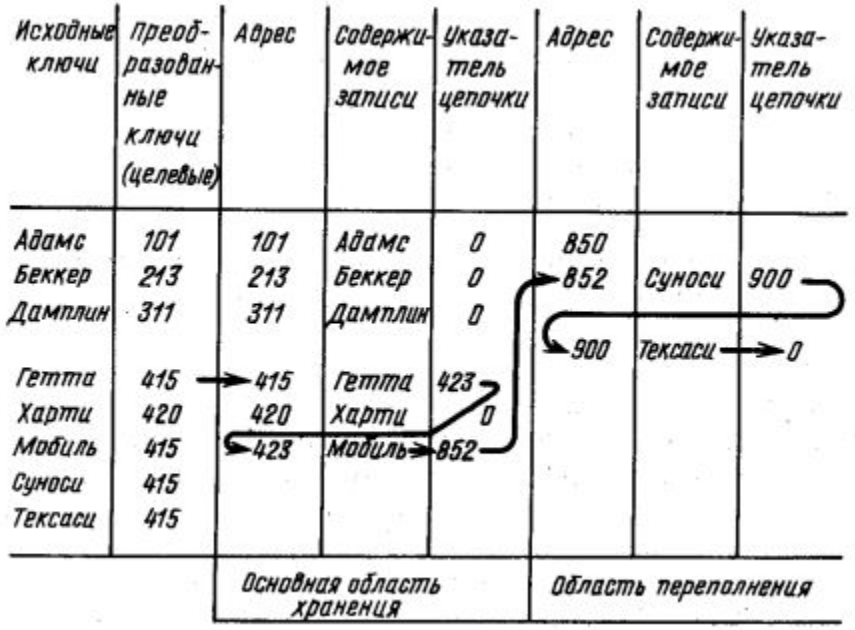
Обратите внимание на цепочку синонимов для записей с ключами Гетта, Мобиль, Суноси и Тексаси. Значение исходного ключа Гетта преобразуется в адрес первой записи цепочки синонимов – 415. Запись Гетта содержит указатель на адрес 423, по которому размещается запись Мобиль.

В свою очередь запись Мобиль содержит указатель на адрес 852, по которому размещается запись Суноси. Запись Суноси содержит указатель на адрес 900, по которому размещается запись Тексаси. Эта последняя запись в цепочке



метод с использованием хеширования

Любой элемент хеш-таблицы имеет особый ключ, а само занесение осуществляется с помощью хеш-функции, отображающей ключи на множество целых чисел, которые лежат внутри диапазона адресов таблицы.



метод с использованием хеширования

Простейшим алгоритмом хеширования может являться функция f(k)=k (mod 10), где k - ключ, целое число

Хеш-функция должна обеспечивать равномерное распределение ключей по адресам таблицы, однако двум разным ключам может соответствовать один адрес. Если адрес уже занят, возникает состояние, называемое коллизией, которое устраняется специальными алгоритмами: проверка идет к следующей ячейке до обнаружения своей ячейки. Элемент с ключом помещается в эту ячейку.

Для поиска используется аналогичный алгоритм: вычисляется значение хеш-функции, соответствующее ключу, проверяется элемент таблицы, находящийся по указанному адресу. Если обнаруживается пустая ячейка, то элемента нет.

метод с использованием хеширования

Размер хеш-таблицы должен быть больше числа размещаемых элементов. Если таблица заполняется на шестьдесят процентов, то, как показывает практика, для размещения нового элемента проверяется в среднем не более двух ячеек.

Режим 1. Получить все данные (последовательная обработка).

Режим 2. Получить уникальные (например, одна запись) данные

Платой за скорость поиска и обновления в режиме 2 являются нарушение упорядоченности файла, потеря возможности выполнять пакетную обработку по первичному ключу. Время обработки в режиме 1 велико.
Эффективность хранения и эффективность доступа при использовании хеширования зависят от распределения ключей, алгоритма хеширования и распределения памяти.