ГЛАВА 5. ФИЗИЧЕСКИЕ МОДЕЛИ ДАННЫХ (СТРУКТУРЫ ХРАНЕНИЯ)

Как уже отмечалось, концептуальная схема, специфицированная к СУБД, автоматически отображается в структуру хранения программами СУБД. Внешний пользователь может ничего не знать о том, как его представление о данных физически организовано в памяти вычислительной системы. Тем не менее от физического размещения данных в памяти ЭВМ существенно зависит время решения прикладных задач. В связи с этим, даже на одном из начальных этапов проектирования базы данных – этапе выбора СУБД, желательно знать возможности физических структур хранения, представляемых конкретными СУБД, и оценивать временные характеристики проектируемой базы данных с учетом этих возможностей.

Способы физической организации данных в различных СУБД, как правило, различны и определяются типом используемой ЭВМ, инструментальными средствами разработки СУБД, а также критериями, которыми руководствуются разработчики СУБД при выборе методов размещения данных и способов доступа к этим данным. Заметим, что наиболее распространенным критерием служит время доступа к данным, однако в качестве критерия может выбираться, например, трудоемкость реализации соответствующих методов.

В настоящей главе будут рассмотрены типовые физические модели организации данных в конкретных СУБД.

Физические модели данных служат для отображения моделей данных. Основными понятиями модели данных являются поле, логическая запись, логический файл. Слово «логический» введено, чтобы отличать понятия, относящиеся к логической модели данных, от понятий, относящихся к физической модели данных.

Основными понятиями физической модели данных, используемыми для представления логической модели данных, являются поле, физическая запись, физический файл. В частности, логическая запись, состоящая из полей, может быть представлена в виде физической записи (из тех же полей), логический файл – в виде физического файла. Прежде чем конкретизировать понятия, относящиеся к физической модели данных, рассмотрим структуру памяти ЭВМ.

5.1. Структура памяти ЭВМ

Важнейшей особенностью памяти ЭВМ, в значительной степени определяющей методы организации данных и доступа к ним, является её неоднородность. Существуют два разных типа памяти – оперативная (ОП) и внешняя (ВП), причем процессор работает только с данными из оперативной памяти.

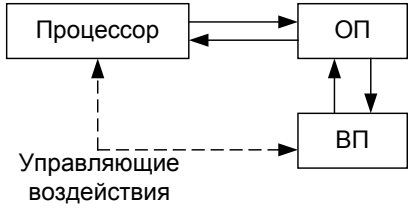


Рис. 32. Схема работы ЭВМ

Как уже многократно отмечалось, базы данных создаются для работы с большими объемами данных, что обусловливает необходимость использования внешней памяти. Поэтому организация данных и доступа к ним должна учитывать как специфику каждого типа памяти, так и способы их взаимодействия.

Отметим основные свойства оперативной памяти:

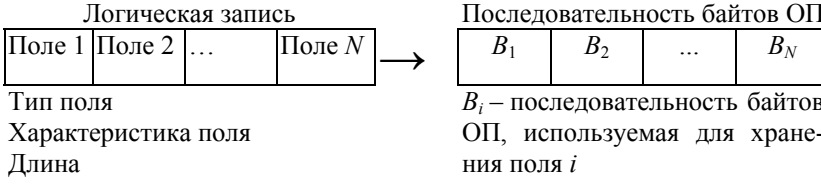
- единицей памяти является байт;
- память прямоадресуема (каждый байт имеет адрес);
- процессор выбирает для обработки нужные данные, непосредственно адресуясь к последовательности байтов, содержащих эти данные.

Отметим основные свойства внешней памяти:

- минимальной адресуемой единицей является физическая запись;
- для последующей обработки (например, работы с полями) запись должна быть считана в оперативную память;
- время чтения записи в ОП (занесения записи из ОП в ВП) на несколько порядков выше времени обработки процессором записи из ОП;
- организация обмена осуществляется порциями, т.к. невозможно считать сразу всю базу данных.

5.2. Представление экземпляра логической записи

Логическая запись представляется в оперативной памяти следующим образом:



Прямая адресация байтов позволяет процессору выбирать для обработки нужное поле.

Заметим, что указанное представление не делает различий для записей в сетевой, иерархической и реляционных моделях. В случае сетевой и иерархической моделей некоторые поля могут являться указателями, тогда последовательность байтов, используемая для хранения этих полей, содержит адрес начала последовательности байтов, соответствующей записи – члену отношения.

В большинстве современных СУБД используется формат записей фиксированной длины. В этом случае все записи имеют одинаковую длину, определяемую суммарной длиной полей, составляющих запись. В СУБД другие форматы записей (переменной длины, неопределенной длины) встречаются гораздо реже, поэтому в данной книге эти форматы не рассматриваются.

Заметим, что поля записи, принимающие значения существенно разной длины в различных экземплярах записей, в реальных задачах встречаются достаточно часто. Примером может служить поле резюме в записи СОТРУДНИК. Резюме может составлять полстраницы текста, страницу и т.д. Возникает проблема – как эту информацию переменной длины представить в записи фиксированной длины. Возможным вариантом является установление размера соответствующего поля по максимальному значению. В этом случае у многих экземпляров записи указанное поле будет заполнено не полностью и, таким образом, память ЭВМ будет использоваться неэффективно. Более эффективный и часто используемый в СУБД прием организации таких записей состоит в следующем. Вместо поля (полей), принимающего значение существенно разной длины, в запись включается поле-указатель на область памяти, где будет размещаться значение исходного поля. Как правило,

эта область является областью внешней памяти прямого доступа. В процессе ввода соответствующего значения в выделенной области занимается столько памяти, какова длина этого значения.

На рис. 33 представлен пример вышеуказанного представления экземпляров записей из N полей, причем поле N принимает значения соответственно разной длины у разных экземпляров записей.

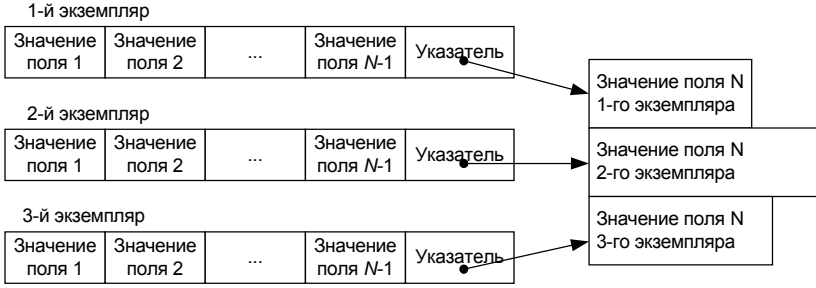


Рис. 33. Представление полей переменной длины

Конкретной реализацией такой схемы является поле типа MEMO в dbase-подобных СУБД (dBase III+, FoxPro и т.д.).

5.3. Организация обмена между оперативной и внешней памятью

Единицей обмена данными между оперативной и внешней памятью является физическая запись. Физическая запись читается (записывается) за одно обращение к внешней памяти. В частности, физическая запись может соответствовать одной логической записи. Число обращений к внешней памяти при работе с базой данных определяет время отклика системы. В связи с этим для уменьшения числа обращений к БД при работе с ней увеличивают длину физической записи (объединяют в одну физическую запись несколько экземпляров логических записей). В этом случае физическую запись называют также блоком, число k экземпляров логических записей, составляющих физическую запись, – коэффициентом блокировки.

Ввод исходных данных в БД осуществляется следующим образом:

- в ОП последовательно вводятся k экземпляров логических записей (кортежей);

- введенные k экземпляров объединяются в физическую запись (блок);
- физическая запись заносится во внешнюю память.

Ввод k экземпляров записей исходной таблицы, составляющих i-ю физическую запись, изображен на рис. 34.

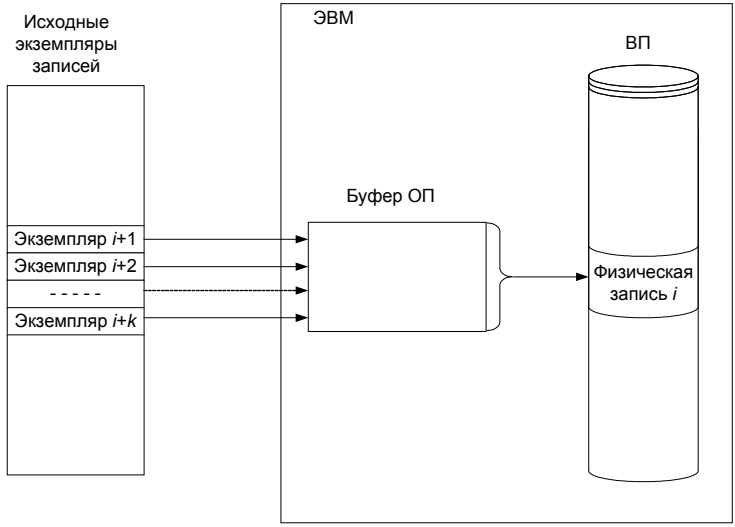


Рис. 34. Схема занесения записей во внешнюю память

Обработка данных, хранящихся во внешней памяти, осуществляется следующим образом:

- физическая запись (блок) считывается в оперативную память;
- обрабатываются экземпляры логических записей внутри блока (выбираются нужные поля, производится сравнение ключевого поля с заданным значением, осуществляется корректировка полей, выполняются операции удаления и т.п.).

5.4. Структуры хранения данных во внешней памяти ЭВМ

В современных СУБД наибольшее распространение получили табличные модели данных. В связи с этим, а также для большей определенности в настоящем разделе мы будем говорить о структурах хранения для табличной модели. Однако отметим, что некоторые из рас-

сматриваемых ниже структур хранения могут использоваться и для представления сетевых и иерархических моделей.

В качестве внешней памяти мы рассматриваем наиболее распространенную в современных ЭВМ память прямого доступа. Память прямого доступа дает возможность обращения к любой записи, если известен её адрес.

Для упрощения изложения мы не будем конкретизировать ряд служебных полей, которые содержит физическая запись, и их рассмотрение опускаем.

5.4.1. Последовательное размещение физических записей

В этой структуре хранения записи в памяти размещаются последовательно друг за другом. Как уже отмечалось, считаем, что все записи имеют равную длину. Физический адрес записи может быть легко вычислен по номеру записи (для вычисления необходимо знать формат соответствующей физической записи).

Физическая запись с номером I содержит логические записи с номерами

$$(I-1)\,k+1$$
$$(I-1)\,k+2$$
$$\ldots$$
$$(I-1)\,k+k$$
$$I = 1, 2, \ldots, \lceil N/k \rceil;$$

знаком $\lceil N/k \rceil$ обозначим ближайшее целое, большее или равное N/k, – целое сверху.

Рассмотрим, как реализуются основные элементарные операции модели данных в этой структуре хранения, и оценим число этих операций. Напомним, что с точки зрения пользователя в табличной модели данных эти операции является операциями над строками (столбцами) таблицы.

Поиск записи с заданным значением ключа

При последовательной структуре хранения поиск может осуществляться только перебором. Читается первая физическая запись, в ОП она разбивается на k логических записей (разблокируется), заданное значение ключа сравнивается со значением ключа каждой логической записи. При несовпадении читается следующая физическая запись и процесс повторяется. В лучшем случае нужная запись будет найдена за

одно обращение, в худшем – необходимо считать все физические записи. Среднее число обращений к внешней памяти для поиска нужной записи *TP* определяется следующей формулой

$$TP = (1+\lceil N/k \rceil)/2 ,$$

где N – число логических записей, k – коэффициент блокировки, $\lceil N/k \rceil$ – число физических записей.

Чтение записи с заданным значением ключа

Сначала необходимо найти нужную запись (смотри операцию «поиск»). После окончания операции «поиск» нужная запись уже считана в ОП. Число обращений к ВП равно *TP*.

Корректировка записи

Сначала необходимо найти нужную запись (смотри операцию «поиск»). После окончания операции «поиск» в ОП найденная логическая запись корректируется, формируется физическая запись (блокировка) и заносится во внешнюю память по тому адресу, откуда она была считана. Число обращений к ВП равно *TP*+1.

Удаление записи

Аналогична операции корректировки. Служебное поле соответствующей логической записи помечается как «удаленная запись». Число обращений к ВП равно *TP*+1.

Добавление записи

Рассмотрим два случая. В первом случае пользователь вводит новую логическую запись в конец таблицы. Тогда вводимая логическая запись добавляется в конец файла. Она заносится либо в последнюю физическую запись (если в ней меньше k логических записей – блок неполон), для чего эта запись должна быть считана в ОП, или формируется новая физическая запись, которая заносится в конец файла. Число обращений к ВП равно соответственно либо 2, либо 1.

Во втором случае пользователь вводит новую логическую запись в указываемую им i-ю строку таблицы (i=1, 2, …, n). В этом случае читается физическая запись с номером $\lceil (i-1)/k \rceil$, содержащая i-ю логическую запись. Если соответствующая физическая запись содержит пустые логические записи, то добавляемая запись вставляется в этот блок, блок записывается на свое место в ВП. Число обращений к ВП равно 2. Если указанная физическая запись содержит k экземпляров

логических записей исходной таблицы, читается физическая запись с номером $\lceil i/k \rceil$. Если эта физическая запись содержит пустые логические записи, добавляемая запись вставляется в этот блок, блок записывается на свое место в ВП. Суммарное число обращений в этом случае будет на единицу больше и равно 3.

Если физические записи с номерами $\lceil (i-1)/k \rceil$ и $\lceil i/k \rceil$ содержат по k экземпляров исходных логических записей, необходимо формировать дополнительную физическую запись. Соответствующий блок будет содержать добавляемую логическую запись и k-1 пустых логических записей. Блоки с номерами $\lceil i/k \rceil$, $\lceil (i+1)/k \rceil$, … $\lceil N/k \rceil$ переписываются на одну позицию ниже (сдвигаются). Сформированная физическая запись заносится на освободившееся место (место записи с номером $\lceil i/k \rceil$).

В лучшем случае ($i = N$) ни один блок не сдвигается. В худшем случае ($i = 1$) сдвигаются все блоки. Среднее число обращений к ВП для перезаписи блоков (чтение + запись) составит $2\lceil N/k \rceil/2$. Тогда суммарное число обращений к ВП при добавлении записи в этом случае будет равно $3+\lceil N/k \rceil$.

5.4.2. Последовательное размещение физических записей с упорядочением по ключу

В этом случае исходные логические записи упорядочиваются по ключу, затем объединяются в физические записи. Значением ключа физической записи является максимальное значение ключей логических записей соответствующего блока. Таким образом, физические записи становятся также упорядочены. Этот порядок определяет последовательность размещения записей в ВП. Заметим, что как и в предыдущем случае, можно вычислить физический адрес записи по номеру и непосредственно обратиться к записи по этому адресу. Рассмотрим, как реализуются основные элементарные операции.

Поиск записи с заданным значением ключа

Поиск осуществляется дихотомическим методом. Номер физической записи, соответствующей середине файла, определяется ближайшим целым числом к числу $\lceil N/k \rceil/2$. По этому номеру определяется номер физической записи, и запись считывается в ОП. Проверяется, лежит ли запись с заданным значением ключа выше или ниже указан-

ной записи. Соответствующая половина файла делится пополам, процесс повторяется до тех пор, пока не будет найдена искомая физическая запись. Число обращений к памяти

$$TP = C \log_2 \lceil N/k \rceil,$$

где C – некоторая константа.

После этого в ОП найденная запись разблокируется, в ней ищется нужная логическая запись.

Чтение записи

Аналогично операции поиска. Число обращений к ВП равно TP.

Корректировка и удаление записи

По аналогии с соответствующим описанием в п. 5.4.1. Число обращений к ВП равно TP+1.

Добавление записи

Необходимость сохранения упорядочения по ключу обусловливает повышенную трудоемкость соответствующей операции. Прежде всего находится физическая запись, после которой должна быть размещена добавляемая запись. Читается следующая физическая запись, и проверяется полнота соответствующего блока. Если блок не полный, логическая запись вставляется в этот блок, блок записывается на свое место в ВП. Если блок полон, то формируется новая физическая запись, которая будет состоять из одной добавляемой логической записи. Число обращений к ВП при этом равно TP+2. Для высвобождения места этой физической записи все последующие блоки переписываются на одну позицию ниже (сдвигаются). Сформированная физическая запись записывается на освободившееся место.

Среднее число обращений к ВП для перезаписи блоков (чтение+запись) оценивается так же, как в п. 5.4.1. и составляет $\lceil N/k \rceil$. Тогда суммарное число обращений к ВП при добавлении записи равно TP+2+$\lceil N/k \rceil$.

Заметим, что при таком способе организации структуры хранения время поиска, чтения, корректировки, удаления будет существенно меньше, чем в предыдущем случае: $C \log_2 \lceil N/k \rceil \ll (1+\lceil N/k \rceil)/2$.

Время добавления записей будет существенно больше. Кроме того, необходимо заметить, что, как правило, вышеуказанные операции производятся с записями, задаваемыми значениями ряда вторичных

ключей, и упорядочение по первичному ключу не сокращает времени соответствующих операций. В связи с этим вышеуказанный способ организации структуры хранения в современных СУБД практически не используется.

5.4.3. Размещение физических записей в виде списковой структуры

Основная проблема в использовании изложенных в п. 5.4.1 и 5.4.2 способов организации записей состоит в отображении добавления логической записи в произвольное место таблицы. При этом приходится переписывать в памяти (сдвигать на одну позицию) физические записи, соответствующие логическим записям таблицы, расположенным ниже места вставки добавляемой строки. Соответствующую проблему можно устранить, используя для представления физических записей связный список (рис. 35).

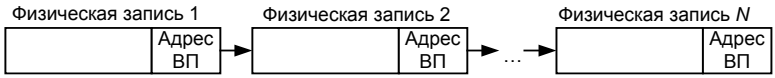


Рис. 35. Список физических записей

Кроме этого списка в ВП формируется список свободных элементов («пустых» физических записей), элементы которого используются при вводе новой записи с данными (рис. 36).

Напомним, что каждая физическая запись состоит, как и ранее, из *k* логических записей.

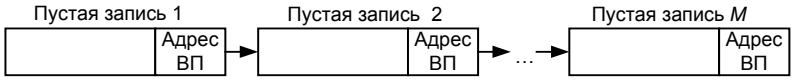


Рис. 36. Список свободных элементов

Рассмотрим, как реализуются основные элементарные операции модели данных в этой структуре хранения.

Поиск записи с заданным значением ключа

Заметим, что упорядочение записей по значениям ключа не дает здесь ускорения процедуры поиска (в отличие от п. 5.4.2) . Это связано с тем, что после ряда добавлений новых записей и удаления каких-то имеющихся записей физическая и логическая последовательность за-

писей в списке будут существенно различаться. При этом будет невозможно по номеру записи определить ее адрес и обращаться к записи, соответствующей середине таблицы, для реализации дихотомического метода поиска. Поэтому поиск можно вести только с помощью перебора. В ОП читается первая запись списка, разблокируется, значения ключевых полей логических записей этой физической записи сравниваются с заданным значением. Если значения совпали, нужная запись найдена, если не совпали, из записи выбирается адрес следующей записи списка, читается эта запись. Далее процедура повторяется. Среднее число обращений к ВП будет равно, как и в 5.4.1, $(1+\lceil N/k \rceil) / 2$.

Чтение записи

После завершения предыдущей операции запись считана в ОП. Оценка числа обращений к ВП та же.

Корректировка записи

Считанная запись корректируется и заносится в ВП на свое место (по своему адресу). Число обращений к ВП на единицу больше, чем при чтении.

Удаление записи

Заметим, что мы говорим об операциях над логическими записями. Операция удаления логической записи аналогична операции корректировки. Служебное поле соответствующей логической записи помечается как «удаленная запись». Сформированная физическая запись заносится в ВП. Число обращений к ВП равно TP+1.

Добавление записи

Для определенности будем считать, что задан ключ логической записи, после которой должна быть добавлена новая запись. Осуществляется операция поиска и чтения физической записи, в которой расположена запись с ключом PK. Если в этом блоке есть логическая запись, помеченная как удаленная, добавляемая запись заносится на ее место. Блок записывается в ВП. Число обращений к ВП равно TP+1. Если в этом блоке нет логических записей, помеченных как удаленные, необходимо добавлять новую физическую запись, выбираемую из списка свободных элементов. С этой целью адрес связи найденной ранее физической записи заменяется на адрес начала списка свободных элементов.

Читается первая физическая запись списка свободных элементов. Адрес связи этой записи заменяет адрес начала пустого списка. В ОП формируется новая физическая запись, содержащая добавляемую логическую запись. В качестве ее адреса связи заносится адрес связи из физической записи, предшествующей добавляемой. Каждая из этих записей заносится в ВП. Число обращений к ВП при добавлении записи будет примерно равно *TP*+3.

Рассмотренный метод организации структуры хранения достаточно эффективно решает проблемы добавления и удаления записей, но не уходит от перебора при поиске нужной записи.

5.4.4. Использование индексов (индексирование)

Как уже отмечалось, упорядочение записей позволяет использовать дихотомический метод поиска нужной записи и тем самым существенно сократить одну из основных составляющих времени поиска – число обращений к ВП. Однако при этом возникают проблемы с добавлением записей, связанные с необходимостью перезаписи части физических записей (сдвига).

Для того чтобы использовать дихотомический поиск и не перемещать физические записи при добавлении новых записей, используется так называемое логическое упорядочение физических записей (индексирование). Основная структура хранения содержит записи исходной таблицы и представлена в виде неупорядоченной последовательности физических записей (см. п. 5.4.1). Для возможной реализации дихотомического поиска по определенному ключу создается дополнительная структура хранения (так называемый индекс). Число записей в индексе равно числу записей исходной таблицы (числу физических записей в основной структуре хранения). Каждая запись индекса имеет два поля: ключевое поле записи основной структуры и указатель – адрес записи основной структуры с соответствующим значением ключа.

Записи индекса (индексного файла) упорядочены по значению ключа. Адреса связи этих записей определяют логическое упорядочение записей основной структуры хранения. Пример соответствующей структуры хранения приводится на рис. 37.

Рассматриваемую структуру хранения называют еще инвертированным списком. Смысл этого термина состоит в следующем. Можно было бы упорядочить записи основной структуры хранения, объединив их в соответствующий упорядоченный список. В нашем случае адреса связи как бы удаляются из списка и включаются в состав фай-

ла-индекса (инвертируются). Поэтому полученная структура интерпретируется как инвертированный список.

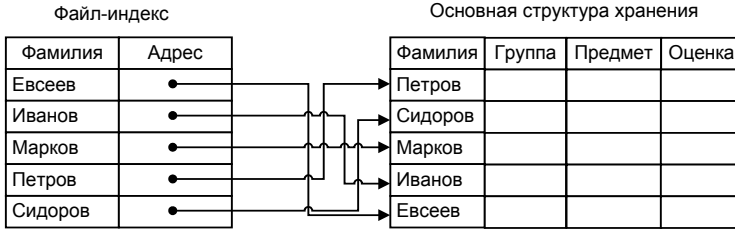


Рис. 37. Индексирование

Поиск нужной записи по заданному значению ключа осуществляется в индексном файле методом половинного деления. Заметим, что так как записи индекса содержат всего два поля, суммарный объем записей индекса невелик, поэтому индекс, как правило, целиком считывается для обработки в ОП за одно обращение к ВП. После того как в индексном файле обнаружена искомая запись, по адресу связи читается полная соответствующая запись основной структуры хранения. Если необходим поиск по другому ключу, строится еще один индекс по соответствующему ключу. Таким образом, по любому ключу поиск можно осуществлять дихотомическим методом.

Оценим число обращений к ВП при реализации элементарных операций. Соответствующие оценки сделаны для случая, когда физическая запись состоит из одной логической записи (коэффициент блокировки k равен 1). Расчет оценок для произвольного k производится по аналогии с расчетами пп. 5.4.1–5.4.3.

Поиск записи с заданным значением ключа

Из ВП читается индексный файл (число обращений к ВП для этого зависит от объема индексного файла, как правило, невелико и много меньше числа записей N). После нахождения нужной записи в индексном файле читается соответствующая запись основного файла (одно обращение к ВП).

Чтение записи

В ходе операции поиска искомая запись считана в ОП.

Корректировка записи

Считанная запись корректируется и заносится на свое место (еще одно обращение к ВП).

Удаление записи

Найденная запись помечается как удаленная в основном файле, соответствующая запись в индексном файле удаляется, измененный индекс записывается в ВП. Число обращений к ВП в этом случае по сравнению с числом обращений к ВП при поиске увеличивается на два.

Добавление записи

Добавляемая запись заносится в конец основного файла. Формируется новая запись индекса, соответствующая добавляемой записи. Записи индекса переупорядочиваются по значениям ключа, и индекс заносится в ВП. Число обращений к ВП в этом случае, в основном, определяется чтением-записью индекса.

Таким образом, использование индексов позволяет ценой некоторого увеличения объема используемой памяти (за счет индекса) существенно сократить время реализации основных операций. В связи с этим индексирование используется во многих современных СУБД.

5.4.5. B-дерево

Если исходный файл имеет большой размер, то его индекс может быть тоже очень велик. Возникает идея аналогично проиндексировать индекс файла (построить индекс над индексом). Такую процедуру можно повторить несколько раз. Таким образом будет построена древовидная структура (дерево). Если количество уровней между корнем и каждым листом одинаково, то в этом случае дерево называется сбалансированным или B-деревом.

B-дерево может быть построено следующим образом. Последовательность записей, соответствующая записям исходной таблицы, упорядочивается по значениям первичного ключа. Логические записи объединяются в блоки (по k записей в блоках).

Значением ключа блока является минимальное значение ключа у записей, входящих в блок. Последовательность блоков представляет собой последний уровень B-дерева. Строится индекс предыдущего уровня. Записи этого уровня содержат значение ключа блока следующего уровня и указатель-адрес связи соответствующего блока; записи

этого уровня также объединяются в блоки (по k записей). Затем аналогично строится индекс более высокого уровня и т.д., пока количество записей индекса на определенном уровне будет не более k. Полученная структура изображена на рис. 38 (для упрощения рисунка на уровне 4 представлены только ключи логических записей и не представлены значения других полей этих записей).

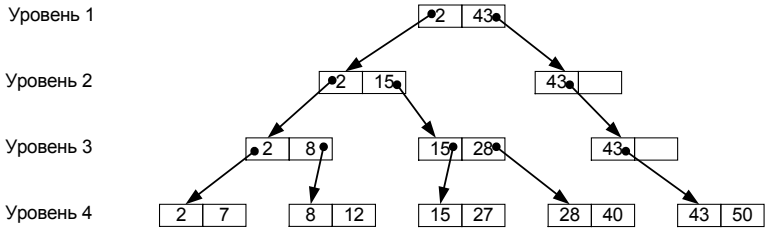


Рис. 38. B-дерево

В блоках указано значение ключа соответствующего блока. Значение k принято равным 2.

По построению B-дерева все исходные записи находятся на одном расстоянии от верхнего индекса (дерево является сбалансированным).

Рассмотрим реализацию основных операций.

Поиск и чтение записи с заданным значением ключа

Читается верхний индекс. Сравниваем заданное значение ключа со значением ключа записей индекса. Если заданное значение ключа больше, чем значение ключа очередной записи индекса (если такая запись имеется), или равно ему, то по адресу связи, указанному в текущей записи, читается блок записей индекса следующего уровня. Далее процесс повторяется.

Считаем, что все блоки расположены в ВП. Тогда число обращений к ВП при поиске информации будет равно числу уровней дерева. Число уровней дерева равно минимальному значению l, при котором выполняется условие $k^l \geq N$ (N – число логических записей).

Модификация (корректировка) записи

После поиска и чтения записи изменяются корректируемые поля. Если корректируется не ключ записи, то измененная запись заносится на свое место. Если изменено значение ключа, то старая запись удаля-

ется (в соответствующем блоке появляется «пустая» запись), а измененная запись заносится так же, как вновь добавляемая.

Удаление записи

После поиска найденная запись удаляется (в соответствующий блок на место этой записи заносится «пустая» запись).

Добавление записи

Прежде всего определяется, где должна быть расположена добавляемая запись с заданным значением ключа. Процедура поиска блока, где должна быть расположена эта запись, аналогична вышеописанной процедуре поиска записей с заданным значением ключа. Если в найденном блоке низшего уровня есть «пустая» запись, добавляемая запись заносится в этот блок (с необходимым переупорядочением записей внутри блока).

Если в соответствующем блоке низшего уровня нет пустого места, блок делится на два блока. В первый из них заносится $[k/2]$ записей, во второй заносятся остальные. Значением ключа каждого из указанных блоков будет являться, как и описано ранее, минимальное значение ключей у записей, входящих в блок. Добавляемая запись заносится в тот блок, значение ключа которого меньше значения ключа добавляемой записи. Появление нового блока с новым значением ключа обусловливает необходимость формирования соответствующей новой записи в индексе на предыдущем уровне. Эта запись содержит новое значение ключа нового блока и указатель на его месторасположение. Процедура добавления такой записи аналогична описанной выше. Находится блок предыдущего уровня, куда должна быть помещена эта запись. Если в блоке есть пустое место, запись добавляется в блок, если блок полон, он делится на два блока, запись заносится в один из блоков, формируется запись индекса предыдущего уровня и т.д.

Возможен вариант, когда придется делить блок самого верхнего уровня и формировать еще один уровень дерева.

Рассмотрим для примера, изображенного на рис. 38, добавление записи с ключом 10.

1. Сравнение на первом уровне.
   $2<10<43$
   Движение по левой ветви.
2. Сравнение на втором уровне.
   $2<10<15$
   Движение по левой ветви.

3. Сравнение на третьем уровне.
   2<8<10
   Движение по правой ветви.
   Искомый блок | 8 | 12 |
4. Блок заполнен.
   Он делится на 2 блока
   | 8 | | | | 12 |
   Сравнение 8<10<12.
   Запись с ключом 10 заносится в блок 1
   | 8 | 10 | | | 12 |
5. На низшем уровне появилась новая запись с значением ключа 12. Необходимо добавление новой записи с ключом 12 и указателем на запись низшего уровня к индексу предыдущего уровня.
6. Запись с ключом 12 уровня 3 должна добавляться в блок | 2 | 8 |.
   Блок полон, он делится на два блока
   | 2 | | | 8 | |
   Сравнение 8<12.
   Запись добавляется во второй блок | 8 | 12 |
7. На уровне 3 появился блок с новым ключом 8. Необходимо добавление новой записи с ключом 8 и указателем на соответствующий блок уровня 3 на уровне 2.
8. Запись с ключом 8 уровня 2 должна добавиться в блок | 2 | 15 |.
   Блок полон, он делится на два блока.
   | 2 | | | 15 | |
   2<8<15
   Запись добавляется в блок 1 | 2 | 8 |.
9. На уровне 2 появился блок с новым ключом 15, необходимо добавление новой записи с ключом 15 и указателем на соответствующий блок уровня 2 на уровне 1.
10. Запись с ключом 15 уровня 1 должна добавляться в блок | 2 | 43 |.
    Блок полон, он делится на два блока.
    | 2 | | | 43 | |
    2<15<43
    Запись с ключом 15 добавляется в первый блок
    | 2 | 15 | | 43 | |
11. Необходимо сформировать еще один уровень дерева | 2 | 43 |.

Полученная структура будет иметь вид, представленный на рисунке 39.

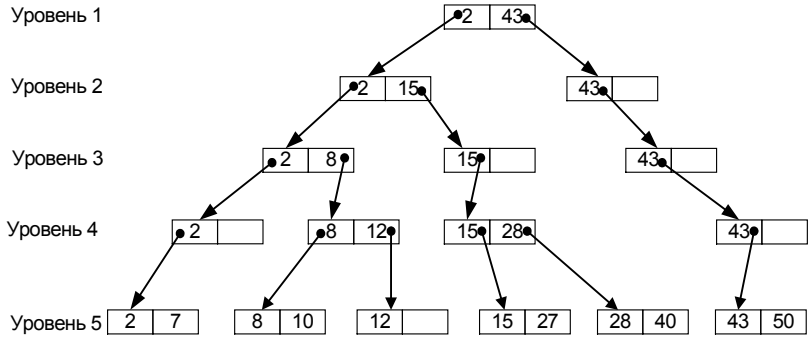


Рис. 39. B-дерево после добавления элемента

Необходимо заметить, что используемый прием деления пополам полностью заполненного блока при добавлении в него записи приведет к тому, что блоки будут заполнены, в среднем, наполовину. Тогда процедура добавления записи будет существенно менее трудоемкой (если в нужном блоке есть место, запись добавляется в этот блок и вышестоящие уровни не перестраиваются).

Структура хранения в виде B-дерева позволяет эффективно проводить операции поиска, чтения, удаления, модификации с оценкой числа обращений к внешней памяти числом уровней дерева l ($l \approx log_k N$), что существенно меньше числа обращений при переборе $\lceil N/k \rceil$.

Процедура добавления записи тоже достаточно эффективна. Соответствующая структура хранения, в частности, используется в отечественной СУБД НИКА (ранее использовалась в системе ИНЕС) и на реальных задачах показала высокую эффективность.

5.4.6. Размещение записей с использованием хэширования

Как в любом другом способе организации структур хранения, логические записи группируются в физические записи (блоки) по k штук. Однако в отличие от всех других способов организации структур хранения здесь выбран особенный способ группировки. Определенным образом выбирается так называемая хэш-функция f. Аргументом этой функции является значение x первичного ключа логической записи.

Тогда $f(x)$ указывает адрес расположения блока, в котором должна находиться логическая запись со значением ключа x.

Функция f должна, по возможности, равномерно распределять значения x по физическим блокам. Обсуждению возможных хэш-функций посвящено достаточно много литературы, поэтому здесь мы не будем касаться этого вопроса. Можно лишь добавить, что иногда, исходя из специфики множества значений x первичного ключа, можно построить функцию f, удовлетворяющую всем необходимым условиям. Таким образом, логическая запись таблицы со значением x первичного ключа размещается в блоке внешней памяти по адресу $f(x)$. В этом блоке может находиться не более k записей. Может оказаться, что выбранная функция отображает в один адрес памяти (один блок) более k записей. Возникает так называемая коллизия. Возможным способом разрешения коллизий является использование дополнительной области переполнения следующим образом. Если очередная запись распределяется с помощью функции хэширования в блок, а он полностью заполнен, то в области переполнения формируется список записей, соответствующих этому блоку, с включением в него указанной записи, а в сам блок заносится указатель – адрес связи на первую запись этого списка. Возможны и другие способы разрешения коллизий.

Рассмотрим реализацию основных операций и дадим оценку числа обращений к ВП при их выполнении.

Поиск записи с заданным значением ключа и чтение

По заданному значению ключа x подсчитывается значение функции $f(x)$. Далее из ВП считывается блок, находящийся по адресу $f(x)$. В ОП внутри этого блока перебором ищется нужная запись. Если записей в блоке нет, то по указателю в блоке (адресу связи) читается первая запись списка переполнения, относящаяся к этому блоку. Далее необходимая запись ищется по этому списку. Число обращений к ВП при этом равно:

– единице, если запись находится в блоке;
– единице плюс число записей в соответствующем этому блоку списке области переполнения (как правило, небольшое число).

Модификации записи

Осуществляется поиск и чтение записи, затем в ОП модифицируются поля записи (не являющиеся первичным ключом), запись заносится на свое место. Число обращений к ВП в этом случае на единицу больше, чем при чтении записи. Если модифицируется значение клю-

ча, то занесение записи осуществляется как ввод новой записи (добавление).

Удаление записи

Осуществляется поиск и чтение записи. Если удаляемая запись находилась в блоке основной памяти, на ее место заносится «пустая» запись (или признак «пустой» записи). Если удаляемая запись находилась в списке области переполнения, удаление ее производится по правилам удаления элемента списка. Число обращений к ВП при удалении находится примерно в тех же пределах, что и для предыдущих операций.

Добавление записи

При добавлении записи со значением ключа *x* подсчитывается адрес соответствующего блока *f(x)*. Блок считывается в ОП. Если в нем есть место, запись заносится в блок, блок записывается в ВП по своему адресу. Если блок заполнен, из него выбирается адрес начала списка записей, переполняющих блок. Далее добавление записи в список производится по правилам добавления элемента в список. Число обращений к ВП при добавлении записей находится примерно в тех же пределах, что и для предыдущих операций.

Таким образом, описанная структура хранения с использованием хэширования является наиболее эффективной (из рассмотренных выше) по критерию минимизации числа обращений к ВП при реализации основных операций.

5.4.7. Комбинированные структуры хранения

Необходимо заметить, что в СУБД могут использоваться как каждая из вышерассмотренных структур в отдельности, так и их комбинация. Так, например, в ряде промышленных систем UNIBAD, БАНК для ЭВМ типа IBM 360/370 (ЕС ЭВМ), PARADOX для персональных ЭВМ используются следующие комбинации методов:

- размещение записей по первичному ключу организовано с использованием хэширования;
- последовательность записей по вторичному ключу задается с помощью списковой структуры.