Базы данных
методы доступа

В.Н.Лукин

18.11.2020

Хранение данных и доступ к ним

Производительность программной системы во многом зависит от методов доступа к данным. В жизни программист для реализации проекта вынужден использовать конкретную СУБД, и возможности маневрировать методами доступа у него небольшие.

Но если можно выбрать СУБД, знание алгоритмов работы с данными в ней может быть полезным.

Кроме того, владение методами хранения данных и доступа к ним позволяет для повышения эффективности иногда прибегать к их собственной реализации

Базовые методы доступа

Проблема эффективного доступа к данным достаточно сложна. Ознакомимся с нею на примере некоторых методов доступа:

- последовательного,
- прямого,
- индексных,
- метода инвертированных списков,
- хеширования (перемешанных таблиц).

Не будем рассматривать методы, основанные на деревьях, а также проблемы физического размещения данных в памяти.

Термины

Эффективность доступа — отношение числа логических обращений к числу физических при выборке элемента данных.

Эффективность хранения — отношение числа информационных байтов к числу физических при хранении.

Ключ поиска – атрибут (группа атрибутов), который должен удовлетворять критерию поиска. Если ключ поиска уникален, его называют *первичным*, иначе говорят о *вторичном* ключе. Далее для простоты будем использовать слово «ключ».

Метод последовательного доступа

Предполагается физическое расположение записей в логической последовательности. Для выборки записи необходимо просмотреть все предшествующие ей.

Эффективность доступа линейно зависит от длины файла, и время доступа слишком велико. Но у него высокая *эффективность хранения*. Кроме того, алгоритм доступа к данным крайне прост.

Метод не применяется там, где необходим быстрый доступ к данным большого объема.

Но его можно использовать при очень небольших объемах данных в силу простоты алгоритма доступа.

Метод прямого доступа

Необходимо взаимно однозначное соответствие между ключом и адресом записи. В этом случае некоторая адресная функция формирует адрес, по которому выбирается запись.

По времени это наиболее эффективный метод, его *эффективность доступа* всегда равна единице. *Эффективность хранения* зависит от плотности размещения ключей.

Метод применяется, когда время — наиболее ценный ресурс (организация таблиц ОС) а также, когда задача предполагает плотное хранение данных с доступом по номеру, например при работе с матрицами.

Индексные методы (1)

Основа индексных методов — вспомогательная структура (*индекс*), с ключами поиска и ссылками на физические адреса данных. В зависимости от вида ключа различают *первичные* и *вторичные* индексы.

Доступ к данным производится в два этапа. Вначале в индексе находят значения ключей, затем из основного файла по ссылке извлекается требуемая информация.

Ни эффективность доступа, ни эффективность хранения не достигают единицы, но производительность системы может стать высокой. Для ее увеличения индекс целиком размещают в оперативной памяти.

Индексные методы (2)

Индексы могут быть устроены по-разному. Если поиск и выборка производится по комбинации атрибутов, индекс называется *составным*. Индекс, построенный на иерархии ссылок, называется *многоуровневым*. Индекс, который содержит ссылки не на все записи, а на диапазон, называется *неплотным*. *Плотный* индекс содержит ссылки на все записи.

Элемент индекса часто называют *статьей*.

Из множества индексных методов рассмотрим три: индексно-последовательный, индексно-произвольный и метод инвертированных списков.

Индексно-последовательный метод (1)

Информационный файл размещается по блокам одного размера, его начало заполняется информационными записями, конец остается свободной.

Для загрузки файл упорядочивается по ключам поиска. Строится индекс, статья которого содержит указатель на блок, в качестве ключа поиска выбирается значение ключа первой или последней записи блока.

Индексы собираются в индексный файл, упорядоченный по значению ключа. Статья индекса ссылается на группу записей в блоке (неплотный индекс).

Порядок следования записей в блоке произвольный.

Индексно-последовательный метод (2)

Информационный файл размещается по блокам одного размера, его начало заполняется информационными записями, конец остается свободной.

Для загрузки файл упорядочивается по ключам поиска. Строится индекс, статья которого содержит указатель на блок, в качестве ключа поиска выбирается значение ключа первой или последней записи блока.

Индексы собираются в индексный файл, упорядоченный по значению ключа. Статья индекса ссылается на группу записей в блоке (неплотный индекс).

Порядок следования записей в блоке произвольный.

Индексно-последовательный метод (3)

Неплотность индекса позволяет уменьшить количество статей кратно размеру блока.

Но индекс все равно может не помещаться в память. Тогда либо увеличивают блок, либо реорганизуют индекс.

Блок увеличивается лишь в ограниченных пределах: он должен помещаться в память, и поиск в нем ключа не должен заметно сказываться на производительности.

С индексом поступают иначе: его рассматривают как информационный файл и тоже индексируют.

Получается иерархия индексов, каждый элемент которой способен разместиться в память.

Индексно-последовательный метод (4)

Для поиска в оперативную память загружается индекс, в нем выбирается ссылка на диапазон ключей, в котором предположительно находится запись. Затем в память загружается нужный блок, в котором последовательным методом ищется нужный ключ.

Новая запись добавляется в свободное место блока. Если его нет, запись либо размещают в связанный блок (*область переполнения*), либо делят блок пополам. В первом случае формирование блока проще, но зона поиска увеличивается. Во втором процесс деления занимает много времени, но время поиска увеличивается незначительно.

Индексно-последовательный метод (5)

Эффективность доступа зависит от размера индексов и числа уровней их иерархии. Кроме того, на нее влияет размер блоков, наличие в них свободных мест, наличие областей переполнения.

Эффективность хранения в основном зависит от объема свободного места в блоках и от величины индексов.

Индексно-последовательный метод
пример (1)

Список студентов с оценками размещается в файле, разделенном на блоки размером в четыре записи. Ссылки на блоки хранятся в индексном файле, ключ – последние фамилии блока.

После начальной загрузки блоки содержат по две записи.

Затем в первый блок добавилась запись с ключом «Баженова», в четвертый — с ключом «Савельева», а второй блок после добавления двух записей оказался полностью занятым. При попытке добавления в него записи с ключом «Лобов» образовался блок расширения, в который позже добавилась запись с ключом «Лобанов».

Индексно-последовательный метод
пример (2)

Фамилия	*Блок*
Болотин	1
Мамчур	2
Пашкевич	3
Смирнова	4

Блок	*Фамилия*	*Имя*	*Оц1*	*Оц2*	*Оц3*
1	Архипова	Мария	5	4	4
	Болотин	Александр	4	3	4
	Баженова	Надежда	3	5	3
2	Бурмистрова	Карина	5	5	5
	Мамчур	Александра	5	5	5
	Захаров	Артём	4	3	4
	Лихарев	Сергей	4	3	4
3	Морозова	Марина	5	4	5
	Пашкевич	Андрей	5	3	5
4	Потапов	Денис	3	4	4
	Смирнова	Александра	4	4	4
	Савельева	Вера	5	5	5

Лобов	Владислав	4	4	3
Лобанов	Александр	4	5	5

Индексно-произвольный метод (1)

Основан на использовании плотного индекса, число его статей равно количеству информационных записей.

Для информационной структуры формируется индекс, который содержит значения ключей поиска и ссылки на записи.

При поиске в индексе находится статья с искомым ключом, затем по ссылке выбирается требуемая запись. Поиск однозначен, если он производится по уникальному индексу. Для вторичного ключа результат поиска зависит от его алгоритма.

.

Индексно-произвольный метод (2)

Нужно стремиться к тому, чтобы весь индекс размещался в памяти. Но в силу плотности индекса индекс может не помещаться из-за большого его размера. Иногда он может даже превышать размер информационного файла.

Уменьшение области поиска достигается, например, построением многоуровневого индекса (дерева поиска). Ключи обычно бывают упорядоченными для последующего дихотомического поиска, но не исключаются и другие алгоритмы.

.

Индексно-произвольный метод (3)

Упорядоченность записей в информационном файле не принципиальна, но иногда она позволяет сократить время работы. Так, выдача отчета по всему файлу с индексом будет оптимальной по времени, если файл упорядочен по ключу поиска. Но если записи по ключу сильно перемешаны, время будет большим из-за хаотичного выбора записей и частого механического перемещения головки дисковода.

Решение — сортировка по ключу. К замедлению поиска приводит и дублирование значений ключей, этот метод наиболее эффективен для первичных индексов.

.

Индексно-произвольный метод (4)

Эффективность доступа во многом зависит от способа поиска статьи индекса, то есть от способа его организации.

Кроме того, на него могут оказывать влияние некоторые свойства ключей (случайное расположение в файле, повторяемость).

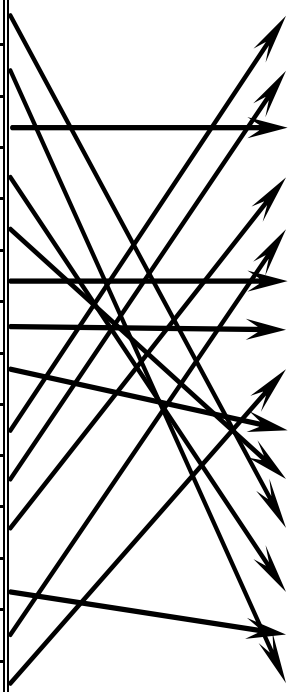
Эффективность хранения зависит от размера индекса.

Индексно-произвольный метод
пример (1)

Список студентов из предыдущего примера размещается в информационном файле (на рисунке справа), ссылки хранятся в индексе (на рисунке слева), в качестве ключа выбираются фамилии.

Индексно-произвольный метод
пример (2)

Фамилия	*№*
Архипова	11
Баженова	14
Болотин	3
Бурмистрова	12
Захаров	10
Лихарев	6
Лобанов	7
Лобов	9
Мамчур	1
Морозова	2
Пашкевич	4
Потапов	13
Савельева	5
Смирнова	8



№	*Фамилия*	*Имя*	*Оц1*	*Оц2*	*Оц3*
1	Мамчур	Александра	5		
2	Морозова	Марина	5	4	5
3	Болотин	Александр	4	3	4
4	Пашкевич	Андрей	5		
5	Савельева	Вера			
6	Лихарев	Сергей	4	3	4
7	Лобанов	Александр	4	5	5
8	Смирнова	Александра			
9	Лобов	Владислав			
10	Захаров	Артём			
11	Архипова	Мария			
12	Бурмистрова	Карина			
13	Потапов	Денис			
14	Баженова	Надежда	3	5	3

Инвертированные списки (1)

Два предыдущих метода ориентировались в основном на поиск записей с уникальным значением ключа.

Однако нередко возникает задача выбора группы записей по определенным параметрам, каждый из которых не уникален.

Это характерно, например, для библиотечного поиска, когда требуется подобрать книгу с заданным годом издания, автором, издательством и т. п. Для подобных задач существуют специальные методы, типичный представитель которых — метод инвертированных списков или *инвертированный* метод.

Инвертированные списки (2)

Поиск может проводиться по значениям любых полей или их комбинации. Для каждого из них создается индекс. В нем для значения ключа формируется список указателей на записи файла с этим значением.

Таким образом, инвертированный индекс группируется по именам полей, которые группируются по значениям.

При поиске выбирается нужный индекс, в нем статья с заданным значением ключа и список ссылок на записи.

Дальнейший выбор записей с одинаковым значением вторичного ключа производится по ссылкам, содержащимся в выбранном списке.

Инвертированные списки (3)

Поиск по комбинации значений полей сводится к выбору соответствующих списков и их пересечению (операция **и**) или объединению (операция **или**).

Действительно, в пересечении списков содержатся ссылки на записи, удовлетворяющие обоим критериям, а в объединении — хотя бы одному.

Критерии могут включать условия как на один ключ, так и на разные. При этом можно использовать не только равенство, но и другие операции отношения.

Инвертированные списки (4)

Более эффективна по времени работа со списками при использовании метода битовых карт, который тоже предназначен для работы с вторичными ключами. Во многом он эквивалентен предыдущему, только вместо списков используются битовые шкалы, длина которых равна количеству информационных записей.

Наличие единицы в позиции *N* означает, что в *N*-й записи значение соответствующего ключа совпадает с искомым значением, наличие нуля — нет.

Инвертированные списки (5)

Здесь вместо работы со списками выполняются логические операции с битовыми шкалами. Для выбора записей просматривают результирующую битовую шкалу и отобирают записи, номера которых равны позициям шкалы, содержащим единицы.

Метод хорош, когда количество различных значений вторичных ключей, невелико, что увеличивает среднее количество единиц в шкале и повышает эффективность работы. В предыдущем методе в таких условиях удлиняются списки, что, наоборот, снижает эффективность.

Инвертированные списки (6)

Особенно удобно использовать битовые карты при задании сложных условий выбора: операции над битовыми шкалами гораздо проще и быстрее, чем со списками.

Понятно, что с ростом количества информационных записей и количества различных вторичных ключей эффективность этого метода падает, особенно эффективность хранения.

Инвертированные списки (7)

Привлекательные стороны метода — независимость от объема файла при выборе данных по произвольным значениям ключа, отбор списка записей по сложным условиям без обращения к файлу.

Особенно эффективно применение инвертированных списков при выборке данных по совокупности критериев, если атрибуты имеют сравнительно небольшой диапазон значений.

Но при создании и обновлении инвертированных индексов тратится много времени, причем время зависит от объема данных. Поэтому данный метод обычно используется лишь для поиска.

Инвертированные списки (8)

Эффективность доступа зависит от эффективности поиска в индексе, но в любом случае она ниже 0,5 (доступ к индексу и доступ к записи файла). Для повышения эффективности следует размещать индексы в оперативной памяти.

Эффективность хранения зависит от метода хранения индекса, от числа инвертируемых полей и от множества значений каждого вторичного ключа (от длины инвертированного списка).

Инвертированные списки
пример (1)

Информационный файл содержит список студентов с оценками по трем предметам.

Левый прямоугольник символизирует индекс, в котором находится единственный вторичный ключ — «Оц1». Каждому его значению соответствует список с номерами записей информационного файла.

Справа от информационного файла такая же конструкция для ключа «Оц2».

Инвертированные списки
пример (2)

Выбор всех, получивших «5» по первому предмету, сводится к нахождению в индексе этого значения ключа «Оц1» и загрузки записей, указанных в списке.

Если нужно найти тех, кто получил «4» или «5», следует найти и объединить соответствующие списки.

Для выборки тех, кто получил пятерки по обоим предметам, следует в соответствующих индексах найти списки для требуемых значений и взять их пересечение.

Инвертированные списки

пример (3)

Оц1
3
4
5

2	
4	
12	
1	
5	
8	
14	
3	
6	
7	
9	
10	
11	
13	

№	Фамилия	Имя	Оц1	Оц2	Оц3
1	Архипова	Мария	5	4	4
2	Баженова	Надежда	3	5	3
3	Болотин	Александр	4	3	4
4	Бурмистрова	Карина	3	3	4
5	Захаров	Артём	4	3	4
6	Лихарев	Сергей	4	3	4
7	Лобанов	Александр	4	5	5
8	Лобов	Владислав	4	4	3
9	Мамчур	Александра	5	5	5
10	Морозова	Марина	5	4	5
11	Пашкевич	Андрей	5	3	5
12	Потапов	Денис	3	4	4
13	Савельева	Вера	5	5	5
14	Смирнова	Александра	4	4	4

1
4
5
6
11
3
10
12
14
2
7
8
9
13

Оц2
3
4
5

Литература

1. *Когаловский М. Р.* Энциклопедия технологий баз данных.
2. *Кнут Д.* Искусство программирования для ЭВМ.
3. *Лукин В.Н.* Введение в проектирование баз данных.
4. Сибуя М., Ямамото Т. Алгоритмы обработки данных
5. *Тиори Д., Фрай Дж.* Проектирование структур баз данных.