4.4. ФАЙЛИ ПРЯМОГО ДОСТУПУ. ФУНКЦІЇ ХЕШУВАННЯ

4.4.1. Концепція організації та доступу у файлах прямого доступу

В деяких інформаційних системах задаються високі вимоги до швидкодії обробки запитів (бронювання авіабілетів, біржові брокери і т. ін.), які не може забезпечити навіть індексна організація файлів. В таки випадках застосовують файли прямого доступу. Основна ідея файлів прямого або довільного доступу полягає в тому, що за значенням ключа шуканого запису k_i знайти адресу запису A_i у відведеному адресному просторі. Найпростіший вихід – це пронумерувати записи і знаходити запис за його номером. Але при великих обсягах файлів цей метод не призводить до бажаних результатів, тому що це та ж сама індексація. Тобто чисто лінійна функція перетворення значення ключа в адресу, яка називається функцією хешування, не дає ефекту. Можна було б ще зробити так: відвести на весь діапазон значень ключа адресний простір з урахуванням довжини запису. Але в реальних задачах значення ключів заповнюються

не щільно, тому це може призвести до запиту великих ресурсів, яких ЕОМ не зможе виділити, а навіть тоді, коли обсягів пам'яті вистачає, вона буде використовуватися дуже не ефективно. Особливо це стосується випадків, коли ключ сам по собі «довгий», наприклад, ключ є складеним. Тому такий довгий ключ «згортають» або перемішують (звідси і назва процесу – хешування) у більш короткий. При «короткому» ключі значно зменшуються обсяги під адресний простір. Поки що не існує теоретичних методів визначення ефективних функцій хешування, це скоріше мистецтво, багато в чому вибір хеш-функції залежить від особливостей ключів файлу й інших полів файлу та досвіду розробників. Основною проблемою при хешуванні є колізія – ситуація, коли при різних значеннях ключа функція дає однакове значення адреси.

Процедура перетворення ключа в адресу звичайно відбувається в три етапи:

1. Якщо ключ не цифровий, він перетворюється у відповідне цифрове представлення таким чином, щоб виключити ймовірність загубити хоча б частину інформації. Наприклад, літерні знаки перетворюються у цифровий код у відповідності до таблиці кодування, наприклад, ASCII-коди.

2. Отримані таким чином ключі перетворюються у сукупність довільно розподілених чисел, значення яких мають той же порядок, що і адреси пам'яті. Такий набір по можливості повинен розподілятися рівномірно за діапазоном припустимих адрес.

3. Далі ключі помножуються на константу, що дозволяє розташувати їх у діапазоні значень адрес виділеної області. Наприклад, у результаті виконання п. 2 ми отримали чотиризначні цифри, а виділена область має 7000 пакетів. Тоді для отримання фізичних ключів слід брати їх добуток на константу 0,7, що дозволить розподілити адреси у діапазоні 0000-6999. Ці відносні адреси перетворюються у машинний адрес пакету.

Розроблено велика кількість алгоритмів для етапу 2. Бажано, щоб отримані значення на цьому етапі були розподілені якомога рівномірніше за припустимим діапазоном. Як вже зазначалося, наявні алгоритми не надають такої можливості, що призводить до колізій, і як результат – направлення багатьох записів у файл переповнення або до списку записів з однаковим значенням хеш-ключа (див. нижче), що суттєво зменшує ефективність цього підходу.

4.4.2. Приклади функцій хешування

Наведемо деякі найбільш розповсюджені функції хешування.

Метод ділення

В основі цього алгоритму лежить просте ділення чисел, при цьому бажано, щоб дільник був простим числом, тобто

$$H(x) = (x \bmod m) + 1,$$

де x – значення ключа;

$\bmod$ – операція ділення націль;

m – просте число, яке трішки менше за кількість пакетів (або число, що не має невеликих дільників).

Приклад

Хай значення ключа представлено рядом x_i=2000, 2001, … , 2017; дільник m=101. Тоді значення згорнутого ключа h_i=<82, 83, … , 99>. Розглянемо інший ряд записів з ключами x_i=<3310, 3311, 3313, 3314, … , 3323, 3324>. При тій же функції хешування отримуємо такі значення згорнутого ключа h_i=<79, 80, 82, 83, … , 92, 93>. Як бачимо, має місце колізія для значень основного ключа 2000 та 3313, яким відповідає значення згорнутого ключа 82, і 2001 та 3314, яким відповідає значення згорнутого ключа 83.

Метод згортання

При цьому методі значення основного ключа поділяють зліва направо на рівні, за кількістю цифр (символів, байтів), частини крім, можливо, останньої частини, а потім отримують суму цих частин, яка і буде значенням згорнутого ключа. При сумуванні інколи не переносять старший розряд, якщо сума за кількістю цифр буде більшою за обрану довжину.

Приклад

Хай значення ключа x_i=187249653, а згорнутий ключ має бути тризначним. Поділяємо основній ключ на три частини по три цифри: 187|249|653. Їх сума дорівнює 187+249+653=1089, відкидаємо старший розряд оскільки значення чотиризначне, і отримуємо значення згорнутого ключа h=89.

Метод середини квадрату

Варіант 1. Ключ підноситься у квадрат, після чого із середини квадрата виділяється стільки знаків, скільки є у кількості пакетів виділеної області пам'яті. Наприклад, виділено 7000 пакетів, ключ пакету – 172148, його квадрат – 029 634 933 904 з середини квадрата адреси виділяємо 4 числа, це – 3493, беремо добуток цього числа на константу 0.7 і отримаємо адресу пакету – 2445.

Варіант 2. При цьому методі значення основного ключа підноситься у ступінь (квадрат), після чого відсікають цифри (біти) зліва і справа до потрібної довжини поля.

Наприклад, хай значення ключа x_1=113586. Тоді $x_1 * x_1$=12901779396. Видалимо зліва чотири цифри, а зліва три, отримаємо значення згорнутого ключа h=1779.

Зсув розрядів

Всі розряди ключа розбиваються на дві частини: старші та молодші. Далі обидві частини зсовуються за напрямком один до одного таким чином, щоб кількість перекритих цифр дорівнювалася кількості цифр пакету. Пари цифр складаються по вертикалі, отримана сума вирівнюється за діапазоном адрес з допомогою константи як у алгоритмі середини квадрату.

Перетворення в іншу систему числення

За цим методом значення основного ключа, представленого в одній системі числення, переводять в іншу (меншу за основою) систему числення, а потім ще видаляють певну частину цифр зліва, доводячи до необхідної довжини: $a_p \rightarrow a_q$, де $p > q$. При цьому бажано, щоб p і q були взаємно простими.

Приклад

Хай значення ключа одинадцятиковій системі числення x_1=530476. При перетворення його у десяткову систему числення воно дорівнює 859745. Якщо буде заданий простір для згорнутого ключа {0, 1, ... , 999}, то відкинемо зліва три цифри і отримаємо значення згорнутого ключа h=745.

Алгебраїчне кодування

Алгебраїчне кодування широко використовуються при криптографуванні інформації, коли певні частини інформації, вважаючи окрему частину за змінну, перетворюються за допомогою якось поліному. У якості такої змінної може розглядатися значення ключа. Після застосування формули (полінома) з новим кодом можуть виконуватися додаткові операції видалення старших розрядів, зсув та інші, але тут ці деталі не розглядаються.

Вибір алгоритму хешування

Як правило, у загальному випадку це можна зробити тільки експериментальним шляхом. Для цього беруть достатньо представницький набір ключів, застосовують до нього всі можливі алгоритми хешування і визначають кількість записів, які направляються у кожний пакет, а також число записів в області переповнення. Експерименти довели,

що метод ділення дає найкращі результати. Треба враховувати, що кращій алгоритм не той, що забезпечує випадкове розподілення записів, а той, що забезпечує рівномірне розподілення записів по всьому адресному простору. В ідеалі файл переповнення залишиться порожнім.

4.4.3 Підходи щодо усунення колізій

Для розв'язання проблеми колізій в ФПД передбачається область переповнення (спільна для всіх пакетів), куди направляються записи з пакетів, які вже заповнені. Пошук даних відбувається наступним шляхом: для запису, який шукається, визначається ключ і відбувається перехід до відповідного пакету записів. Якщо запис не знайдено, відбувається пошук в області переповнення (звичайно невеликій). Якщо запис не знайдено – він у базі відсутній.

Якщо колізії будуть зустрічатися достатньо часто, то загальна область переповнення стає дуже великою і пошук у неї послідовним методом стає неефективним. У цьому випадку виділяють області переповнення при кожному з окремих пакетів. При цьому область переповнення, яку необхідно переглядати, навіть при великій кількості колізій стає достатньо малою, і процес пошуку суттєво прискорюється.

Збільшення елементів, які розташовуються в окремому пакеті можуть суттєво зменшити час пошуку, тому що файл переповнення буде мати значно менший об'єм. У такому випадку цей прийом дещо нагадує сторінкову організацію файлу, але з меншим обсягом сторінки і прямим доступом до неї.