

cji (np. aktualizacja zbioru rachunków klientów przez miesięczne płatności). Jednakże często zdarza się, że każdy zapis jest aktualizowany przez szereg transakcji, np. aktualizacją zbioru o stanie magazynu przez zamówienia klientów, przy czym poszczególne wyroby są zamawiane przez wielu klientów. Rozważmy poprzedni przykład przy założeniu, że trafiony zapis jest aktualizowany (średnio) przez 20 transakcji, tj. cały zbiór - przez 100 000 transakcji, czyli

- aktualizacja bezpośrednia spowoduje 100 000ostępów (zamiast 5000), a więc całkowity czas przetwarzania wyniesie: $100\ 000 (70 + 50 + 5) = 208\ \text{min}$,

- aktualizacja sekwencyjna spowoduje tylko jeden dostęp do wyszukanego zapisu, czyli całkowity czas przetwarzania nie zmieni się i będzie wynosił 3,8 min.

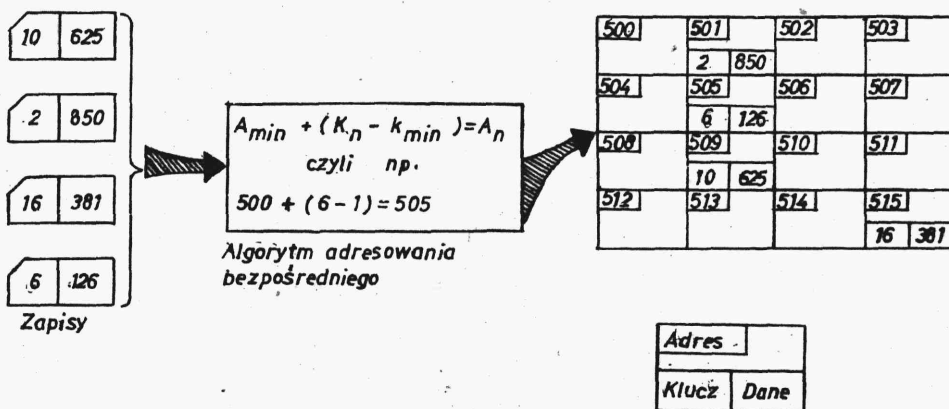
10.4. Zbiory o organizacji losowej

W celu zlokalizowania zapisu w zbiorach o organizacji losowej, stosowane jest odwzorowanie wartości klucza tego zapisu w adres zapisu w pamięci. Odwzorowanie to nazywane jest algorytmem (lub funkcją) mieszającym (ang. hash function, randomizing f.). Adres otrzymany w wyniku działania algorytmu może być bezwzględnym adresem fizycznym (tzn. numer cylindra i głowicy lub nawet bloku na ścieżce) lub adresem względnym (na przykład kolejny numer bloku (porcji) w zbiorze), zamienianym na adres fizyczny przez standardowe programy zarządzania danymi.

Szczególnym przypadkiem funkcji mieszającej, zapewniającej jednoznaczne odwzorowanie zbioru wartości kluczy w równoliczny zbiór adresów jest funkcja liniowa. Metodę tę nazywa się adresowaniem bezpośrednim (ang. direct addressing) (lub samoindeksowaniem (ang. selfindexing)).

10.4.1. Zastosowanie adresowania bezpośredniego

Adresowanie bezpośrednie polega na wzajemnie jednoznacznym odwzorowaniu zbioru możliwych wartości kluczy w zbiór adresów zapisów. Metoda ta może być stosowana dla zapisów stałej długości, których klucze przyjmują wartości ze zbioru liczb naturalnych. Idea metody przedstawiona jest na rys.10.9, przy czym założono, że w danej "komórce" pamięci zawarty jest



Rys.10.9. Adresowanie bezpośrednie

jeden zapis. Najczęściej jednak podstawowymi jednostkami pamięci są bloki wielozapisowe, dlatego też potrzebna jest funkcja przekształcająca wartość klucza w adres bloku, który następnie przetwarzany jest seryjnie.

Rozważmy zbiór składający się z n_z zapisów stałej długości, których klucze przyjmują wartości z przedziału $\langle k_{min}, k_{max} \rangle$. Znając liczbę zapisów tworzących blok n_{zb} oraz liczbę bloków w ścieżce n_{bs} , można określić [2], [7], [8], [25] dla zapisu z kluczem o wartości $K_o \in \langle k_{min}, k_{max} \rangle$:

- numer porcji (standard ODRA/ICL), nr_p :

$$nr_p = \lfloor (K_o - k_{min} + n_{zb}) / n_{zb} \rfloor,$$

przy czym $nr_p \in \langle 1, n_b \rangle$,

- numer ścieżki nr_s oraz numer bloku na ścieżce nr_{bs}
(standard IBM):

$$nr_s = \lfloor (K_o - k_{min}) / n_{zs} \rfloor ,$$

$$nr_{bs} = \lfloor \text{mod}(K_o - k_{min}, n_{zs}) / n_{zb} \rfloor + 1 ,$$

przy czym:

$$nr_s \in \langle 0, n_s - 1 \rangle , \quad nr_{bs} \in \langle 1, n_{bs} \rangle ,$$

gdzie: n_s - całkowita liczba ścieżek potrzebna do zapisania zbioru *),

$$n_{zs} = n_{zb} \cdot n_{bs} .$$

Zgodnie z przedstawioną metodą należy zostawić puste miejsca dla zapisów nie istniejących. Ograniczenie to prowadzi do słabego zapełnienia obszaru pamięci, jeśli w zbiorze wartości klucza zapisów istniejących występują "luki", tj. grupy kolejnych nie używanych wartości. Jeśli "luk" jest niewiele, można zmniejszyć zajętość pamięci, projektując specjalny dodatkowy algorytm kompresji ciągu istniejących wartości kluczy (Czytelnik proszony jest o zaproponowanie takiego algorytmu).

10.4.2. Zastosowanie algorytmu mieszającego

Omówiona powyżej metoda adresowania jest najszybszą metodą lokalizacji zapisu w pamięci zewnętrznej na podstawie wartości jego klucza. Posiada jednak istotną i często dyskwalifikującą wadę, mianowicie obszar zarezerwowany na zbiór jest na ogół słabo zapełniony, chyba że rozkład wartości kluczy jest bardzo gęsty. Jeśli projektant systemu informatycznego ma wpływ na wartości klucza przydzielone poszczególnym zapisom (na ogół jednak tak nie jest), można stosować adresowanie bezpośrednie. Wtedy jednak użytkownik systemu jest zobowiąza-

*) Znajac numer ścieżki oraz numer pierwszego cylindra zbioru, łatwo można określić numer cylindra i głowicy, którą można przetworzyć szukany zapis.

ny pamiętać często dosyć długie wartości klucza, aby odszukać określony zapis. A więc kłopotami związanymi z szybkim wyszukiwaniem informacji obciążono w tym przypadku użytkownika.

Stosowanie innych algorytmów mieszających zwiększa liczbę dostępów potrzebnych do zlokalizowania zapisu, ale jednocześnie zmniejsza się zajętość pamięci oraz nie narzuca się tak ostrych warunków na rozkład wartości kluczy.

Decydujący wpływ na wybór i działanie algorytmu mieszającego mają: zbiór możliwych wartości klucza, zbiór zapisów, zbiór zarezerwowanych adresów oraz współczynnik wypełnienia, określający stosunek liczby rzeczywistych zapisów do liczby możliwych wartości klucza, czyli określający stosunek liczności zbioru zapisów i zbioru wartości klucza. A więc algorytm mieszający przekształca zazwyczaj duży zbiór wartości klucza w zazwyczaj mniejszy zbiór adresów. Dlatego też w ogólnym przypadku algorytm mieszający może przyporządkowywać zapisom o różnych wartościach klucza ten sam adres. Zapisy te będziemy nazywali synonimami (rys.10.4).

Liczba synonimów zależy przede wszystkim od doboru algorytmu mieszającego oraz od rozkładu wartości kluczy.

Algorytm mieszający powinien spełniać następujące warunki:

- adres wygenerowany dla każdego klucza powinien należeć do obszaru przeznaczonego na rozważany zbiór,
- wygenerowane adresy powinny posiadać jak najbardziej równomierny rozkład,
- nie powinno powstawać zbyt wiele synonimów.

Poniżej przedstawione zostaną przykłady kilku najczęściej używanych algorytmów mieszających:

1. Dzielenie przez liczbę pierwszą.

Wartość klucza dzielimy przez największą liczbę pierwszą nie przekraczającą liczby dostępnych adresów, zaś adresem jest reszta z dzielenia. Jeśli zbiór adresów składa się z 500 "komórek", wtedy szukana liczba pierwsza wynosi 499. Zapis z kluczem o wartości 2345 będzie umieszczony w "komórce" 349, bowiem $\text{mod}(2345, 499) = 349$. Algorytm ten jest prosty i najczęściej bardzo skuteczny. Wskazane jest, aby przy wybo-

rze najważniejszej metody transformacji kluczy w adresy wypróbować jako pierwszy właśnie ten algorytm.

2. Wybieranie cyfr

Po wykonaniu analizy rozkładu cyfr dziesiętnych na poszczególnych pozycjach klucza wybieramy te pozycje, na których cyfry 0 - 9 pojawiają się najbardziej równomiernie. Liczba utworzona z wybranych cyfr tworzy adres względny zapisu w zbiorze. Rozważmy zbiór 8000 zapisów, których klucze przyjmują wartości z przedziału $\langle 1, 999999 \rangle$ i dla którego zarezerwowano 10 000 adresów. Analiza statystyczna wykazała, że na pozycjach pierwszej i czwartej pojawia się stosunkowo dużo cyfr 0, 1, 2. Względny adres będą tworzyć więc cyfry z pozycji drugiej, trzeciej, piątej, szóstej. Na przykład zapis z kluczem o wartości 749265 będzie miał adres względny 4965. Metoda ta jest szczególnie odpowiednia, jeśli kluczem zapisu jest pole łatwe do analizy, jak np. numer konta, numer polisy ubezpieczeniowej itp.

3. Dzielenie i składanie klucza

Klucz dzielony jest na kilka części tej samej długości co względny adres. Suma tych części z pominięciem przeniesienia jest adresem względnym. Dla 1000 adresów klucz 9-cyfrowy należy podzielić na trzy 3-cyfrowe części. Zapis z kluczem o wartości 258 694 305 będzie przechowywany pod adresem 257, ponieważ $258 + 694 + 305 = 1257$. Jeśli potraktujemy klucz jako łańcuch bitów, można zamiast dodawania arytmetycznego stosować działania logiczne.

4. Zmiana podstawy liczenia

Przyjmujemy, że dany klucz jest zapisany w innym systemie liczenia (z inną podstawą). Zamieniamy go na liczbę dziesiętną i odpowiedniej długości fragment otrzymanej liczby da nam adres względny. Rozważmy klucz 749 625 i podstawę 11. Mamy więc $7 \cdot 11^5 + 4 \cdot 11^4 + 9 \cdot 11^3 + 6 \cdot 11^2 + 2 \cdot 11 + 5 = 1\,198\,653$. Dla zbioru 1000-adresowego bierzemy trzy ostatnie cyfry, czyli adres względny wynosi 653.

Przedstawione powyżej 4 algorytmy oraz metoda adresowania bezpośredniego stanowią podstawę do tworzenia całego szeregu różnorodnych algorytmów adresowania. Interesujące rozważania na temat algorytmów mieszających można znaleźć w [18], [20].

Oszacowanie danej metody adresowania polega na obliczeniu procentów powstających synonimów oraz oszacowaniu średniej liczbyostępów do pamięci, w celu lokalizacji zapisu.

Przykład 10.8

Stosując badany algorytm do zbioru 10 kluczy, otrzymano następujące adresy 1,2,3,4,5,6,7,8,1,2, czyli otrzymano 20% synonimów. Dla pierwszych 8 zapisów wymagane jest po jednym dostępie, zaś dla dwóch pozostałych - po dwa dostępy, całkowita liczbaostępów wynosi 12, a więc średnia liczbaostępów dla jednego rekordu wynosi 1,2.

Stosując inny algorytm otrzymano adresy 1,2,3,4,5,6,7,8, 1,1, czyli mamy także 20% synonimów. Jednak liczbaostępów wyniesie 13, mianowicie: 8 dostępów dla pierwszych ośmiu zapisów, 2 - dla dziewiątego i 3 - dla dziesiątego. Średnia liczbaostępów na jeden zapis wyniesie 1,3.

Na ogół przyjmuje się, że algorytm mieszający jest dobry, jeśli średnia liczbaostępów dla jednego zapisu nie przekracza wartości 1,2.

Poza doborem algorytmu mieszającego bardzo istotnym zagadnieniem jest dobór odpowiedniej metody zarządzania synonimami. Poniżej przedstawiono dwie podstawowe metody lokalizacji zapisu, gdy pod wygenerowanym dla niego adresem znajduje się już inny zapis:

- zapis-synonim zapisywany jest pod pierwszym wolnym adresem, począwszy od adresu pierwotnego. Przy wyszukiwaniu przeszukiwany jest cały zarezerwowany obszar, począwszy od adresu pierwotnego;

- zapisy-synonimy przechowywane są w specjalnie zarezerwowanym obszarze nadmiarów (por. zbiory indeksowo-sekwencyjne). W zapisie w obszarze podstawowym wprowadzony jest odsyłacz do pierwszego zapisu listy synonimów związanych z danym adresem. Gdy wystąpi synonim, przy operacji zapisywania należy

sprawdzić całą listę synonimów w celu stwierdzenia, czy nie jest podejmowana próba zapisania dwóch różnych zapisów z tym samym kluczem.

Ogólnie można powiedzieć, że dostęp do zbiorów o organizacji losowej wymaga mniej czasu niż do zbiorów o innej organizacji, jednak tracony jest pewien obszar pamięci. Organizacja ta jest stosowana wtedy, gdy czynnikiem krytycznym jest czas dostępu do informacji. Chcąc przyspieszyć wyszukiwanie grupy zapisów w zbiorze o organizacji losowej można wstępnie wygenerować adresy na podstawie wartości kluczy szukanych zapisów, a następnie transakcje wejściowe posortować wg wartości adresów. W ten sposób unikniemy zbędnych ruchów głowic w porównaniu z wyszukiwaniem zapisów osobno.

10.5. Zbiory inwersyjne

W dotychczasowych rozważaniach o organizacji zbiorów zajmowaliśmy się lokalizacją zapisu na podstawie wartości jego klucza pierwotnego, czyli klucza jednoznacznie identyfikującego dany zapis. Przedstawione powyżej organizacje zbiorów ułatwiały odpowiedzi na pytania i cechy (atrybuty^{*)} określonego obiektu, czyli o wartości kluczy wtórnych zapisu o danej wartości klucza pierwotnego. Na przykład dla zbioru KARTOTEKA OSOBOWA mogą to być pytania o zawód, staż pracy i zarobki pracownika o określonym numerze ewidencyjnym.

Czasami jednak, zwłaszcza w systemach wyszukiwania informacji, interesują nas pytania, dotyczące znalezienia obiektów o określonych cechach, czyli o określonych wartościach atrybutów. A więc zadanie polega na wyszukaniu zapisów (czyli pewnego podzbioru wartości klucza pierwotnego), których klucze wtórne przyjmują określone wartości.

^{*)} Klucze danego zapisu, opisującego pewien obiekt, nazywane są często (zwłaszcza w tzw. systemach wyszukiwania informacji) atrybutami tego obiektu lub atrybutami tego zapisu. Oczywiście atrybuty, podobnie jak klucze, przyjmują określone wartości, np. atrybut ZAWÓD może przyjmować wartości: ELEKTRONIK, ELEKTRYK lub INFORMATYK, zaś atrybut WIEK - wartości 25, 36, 48 itp.