Indeksy haszowe

Wprowadzenie

- Podstawowa idea techniki haszowania polega na wykorzystaniu funkcji haszowej, która odwzorowuje wartości atrybutu haszowego w zbiór liczb naturalnych odpowiadających numerom bloków dyskowych w celu znalezienia bloku zawierającego poszukiwany rekord. Zasadniczą wadą haszowania statycznego są potencjalnie długie łańcuchy odwołań, które mogą istotnie obniżać efektywność przetwarzania

Wprowadzenie

- Indeksy haszowe nie wspierają zapytań z przedziałem wartości. Indeksy o strukturze drzewiastej wspierają zapytania z przedziałem wartości, i dodatkowo, są tak samo efektywne jak indeksy haszowe w przypadku zapytań punktowych. Stąd, większość systemów komercyjnych baz danych stosuje wyłącznie indeksy drzewiaste. Okazuje się jednak, że indeksy haszowe okazują się bardzo przydatne w przypadku implementacji niektórych operacji relacyjnych, np. połączeń.

Haszowanie statyczne

- Strony (bloki dyskowe) zawierające dane można postrzegać jako zbiór tzw. zbiorników (ang. buckets**)**, z których każdy składa się ze strony podstawowej i ewentualnie dodatkowych stron przepełnienia (ang. overflow). Początkowo, plik danych zawiera N zbiorników (od 0 do N-1), z których każdy zawiera tylko stronę podstawową

Haszowanie statyczne

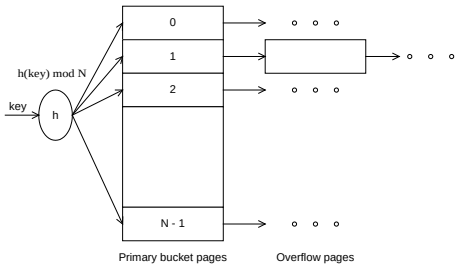


Figure 6.1 Static Hashing

Haszowanie statyczne

- W celu wyszukania rekordu stosujemy funkcję haszową, która wskazuje zbiornik zawierający dany rekord, i następnie, przeszukujemy zbiornik w celu znalezienia rekordu. W celu przyśpieszenia procesu szukania rekordy w zbiorniku mogą być uporządkowane wg. wartości atrybutu haszowego
- Aby wprowadzić rekord do pliku stosujemy funkcję haszową, która wskazuje odpowiedni zbiornik a następnie wstawiamy rekord do zbiornika. W przypadku braku wolnej przestrzeni w zbiorniku alokujemy nową stronę do zbiornika (*overflow page)*, umieszczamy rekord na tej stronie, i dodajemy tę stronę do łańcucha zbiornika

Haszowanie statyczne

- Aby usunąć rekord, stosujemy funkcję haszową, która wskazuje odpowiedni zbiornik, a następnie przeszukujemy zbiornik w celu znalezienia poszukiwanego rekordu i usuwamy ten rekord ze zbiornika. Jeżeli jest to jedyny rekord na stronie, to usuwamy te stronę ze zbiornika
- Ponieważ liczba zbiorników jest znana w momencie tworzenia pliku, strony podstawowe mogą być zaalokowane w kolejnych blokach dyskowych Stąd, proces wyszukiwania wymaga dokładnie 1 dostępu do bloku dyskowego, natomiast wstawianie i usuwanie 2 dostępów do bloków dyskowych. W przypadku występowania łańcuchów odwołań koszt może być wyższy

Haszowanie statyczne

- Wraz z rozwojem pliku mogą tworzyć się długie łańcuchy odwołań do stron przepełnienia obniżające efektywność plików haszowych
- Podstawowym problemem haszowania statycznego jest to, że liczba zbiorników jest stała. Jeżeli plik maleje, to mamy marnotrawstwo przestrzeni dyskowej. Jeżeli plik rośnie, to to tworzą się długie łańcuchy odwołań. Alternatywą jest periodyczna „reorganizacja" pliku. Jednakże, „reorganizacja" pliku zabiera czas i plik w trakcie reorganizacji jest niedostępny

Haszowanie rozwijalne

- **Idea**: w celu rozwiązania problemu haszowania statycznego stosujemy katalog (ang. directory**)** wskaźników do zbiorników i zwiększamy liczbę zbiorników poprzez dublowanie rozmiaru katalogu i podział zbiorników, które są przepełnione

Haszowanie rozwijalne

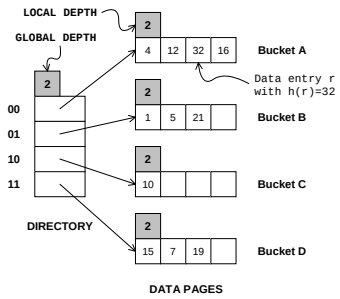


Figure 2 Example of an Extendible Hashed File

Haszowanie rozwijalne

- Katalogiem jest tablica o rozmiarze 4 składająca się ze wskaźników do zbiorników. Aby zlokalizować dany rekord stosujemy funkcję haszową i bierzemy pod uwagę dwa ostatnie bity reprezentacji binarnej wyniku haszowania (liczba z przedziału 0 – 3). Wskaźnik w tablicy o danej pozycji wskazuje na szukany zbiornik; załóżmy, że każdy zbiornik (strona) może przechowywać 4 rekordy. Stąd, aby zlokalizować (wprowadzić) rekord 13 o wartości funkcji haszowej 5 (binarnie 101), sprawdzamy element tablicy o adresie 01 i korzystając ze wskaźnika wczytujemy odpowiednią stronę (zbiornik B).

Haszowanie rozwijalne

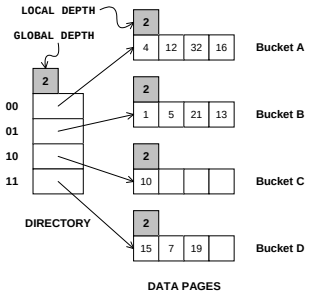


Figure 3 After Inserting Entry *r* with *h(r)*=13

Haszowanie rozwijalne

- Rozważmy problem wprowadzenia nowego rekordu do pełnego zbiornika. Rozważmy wprowadzenie rekordu o wartości atrybutu haszowego 20 (binarnie 10100). Sprawdzamy element katalogu adresie 00, a następnie przechodzimy do zbiornika A, który jest już pełen. Musimy podzielić ten zbiornik i zaalokować nowy blok dyskowy do zbiornika (nową stronę), a następnie dokonać redystrybucji rekordów pomiędzy strony zbiornika. Aby dokonać tej redystrybucji bierzemy pod uwagę 3 ostatnie bity *h(r)*; dwa ostatnie są równe 00, trzeci bit dzieli rekordy pomiędzy nowe strony. discriminates between these buckets. Redystrybucję rekordów ilustruje rysunek Figure 4

Haszowanie rozwijalne

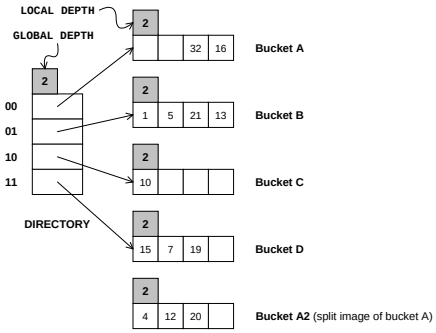


Figure 4 While Inserting Entry *r* with *h(r)*=20

Haszowanie rozwijalne

- Zauważmy problem – do rozróżnienia rekordów w zbiorniku A zaalokowanych pomiędzy strony A i A2 potrzebujemy 3 bitów, natomiast katalog uwzględnia tylko 2 bity reprezentacji i zawiera 4 wskaźniki do zbiorników
- Rozwiązanie – zwiększyć rozmiar katalogu. Rekordy, których wartość funkcji haszowej różni się tylko na 3 bicie są zaalokowane do dwóch stron. W przykładzie, zbiornik 0 został podzielony, nowy element katalogu 000 wskazuje na pierwszą stronę (A), natomiast element 100 wskazuje na drugą stronę (A2). Ilustruje to rysunek Figure 5

Haszowanie rozwijalne

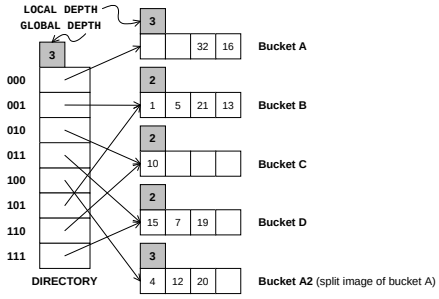


Figure 5 After Inserting Entry *r* with *h(r)*=20

Haszowanie rozwijalne

- Zauważmy, że idea haszowania rozwijalnego polega na wykorzystaniu reprezentacji binarnej wartości funkcji haszowej i uwzględnianiu przy alokacji rekordów ostatnich d bitów, gdzie d zależy od rozmiaru katalogu. Liczbę d nazywamy globalną głębokością pliku haszowego
- **Pytanie**: czy podział zbiornika pociąga za sobą zawsze konieczność zwiększania rozmiaru katalogu?
- Wprowadźmy rekord o wartości 9.Należy on do zbiornika B. Zbiornik jest już pełen. Można to rozwiązać przez podział zbiornika B i wykorzystanie elementów katalogu 001 i 101 jak pokazano na rysunku

Haszowanie rozwijalne

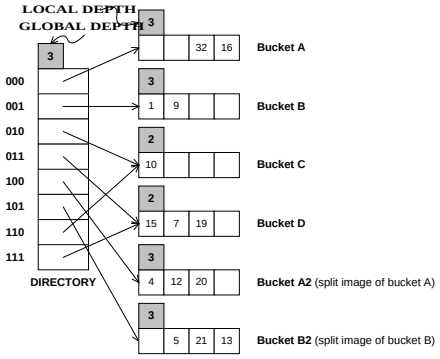
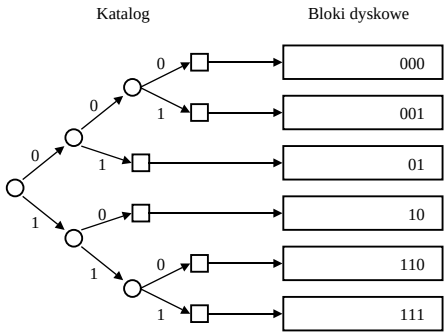


Figure 6.6 After Inserting Entry *r* with *h(r)*=9

Haszowanie rozwijalne

- Dla każdego zbiornika przechowujemy tzw. lokalną głębokość zbiornika. W przypadku podziału zbiornika, którego lokalna głębokość jest równa głębokości globalnej katalogu, system zawsze zwiększa rozmiar katalogu (rozmiar * 2)
- Początkowo, wszystkie głębokości lokalne są równe głębokości globalnej katalogu. Każdorazowo zwiększamy głębokość globalną o 1 gdy zwiększamy rozmiar katalogu. Każdorazowo zwiększamy głębokość lokalną o 1, gdy następuje podział zbiornika. Nowa wartość głębokości lokalnej jest przypisywana również do bloku nowo przydzielonego do zbiornika

Haszowanie dynamiczne



Haszowanie dynamiczne

- Dzięki zastąpieniu statycznej tablicy haszowej - dynamicznym drzewem binarnym, haszowanie dynamiczne gwarantuje lepsze dopasowanie rozmiaru pliku haszowego do liczby składowanych rekordów
- Liczba bloków pliku jest zmienna. Na początku plik zawiera tylko jeden blok. Następnie jest budowana drzewiasta struktura katalogu (drzewo binarne) o dwóch typach wierzchołków:
  - wierzchołki wewnętrzne
  - wierzchołki liści
- Jeżeli funkcja haszowa rozprasza rekordy równomiernie, to indeks jest zrównoważony

Haszowanie liniowe

- **Idea**: Korzystamy ze zbioru funkcji haszowych h_0, h_1, h_2, ... , o takiej własności, że przedział wartości funkcji h_{i+1} jest dwa razy większy od przedziału wartości funkcji h_i. Innymi słowy, jeżeli h_i alokuje rekordy do jednego z M zbiorników, to h_{i+1} alokuje rekordy do 2*M* zbiorników. Najczęściej,wybieramy funkcję *h* i początkową liczbę zbiorników *N* i definiujemy funkcję h_i następująco

  $h_i(value) = h(value) \bmod (2^i N)$
- Ideę można przedstawić w terminach rund podziałów. W czasie rundy o numerze *Level*, wyłącznie funkcje haszowe h_{level} i $h_{level+1}$ są stosowane. Zbiorniki pliku są dzielone jeden po drugim zaczynając od pierwszego zbiornika. Pod koniec rundy mamy dwa razy więcej zbiorników

Haszowanie liniowe

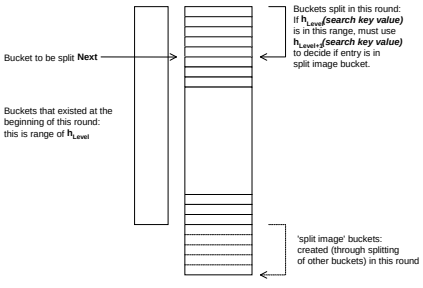


Figure 7 Buckets during a Round in Linear Hashing

Haszowanie liniowe

- Wyszukiwanie rekordu: stosujemy funkcję haszową h_{level}, jeżeli wskazuje ona na nie podzielony zbiornik, przeglądamy wskazany zbiornik. Jeżeli wskazuje ona na podzielony zbiornik, to rekord może znajdować się we wskazanym zbiorniku lub jego „odbiciu”. Dlatego w stosunku do zbiorników podzielonych w danej rundzie stosujemy funkcję haszową $h_{level+1}$.
- Licznik *Level* wskazuje numer rundy i początkowo jest równy 0. Zbiornik do podziału jest wskazywany przez indeks oznaczony *Next* początkowo jest równy 0 (pierwszy zbiornik). Liczbę zbiorników w pliku na początku rundy *Level* oznaczymy przez N_{Level}. Łatwo zauważyć że $N_{Level} = N * 2^{Level}$. Niech liczba zbiorników na początku rundy 0, oznaczona N_0, wynosi *N*

Haszowanie liniowe

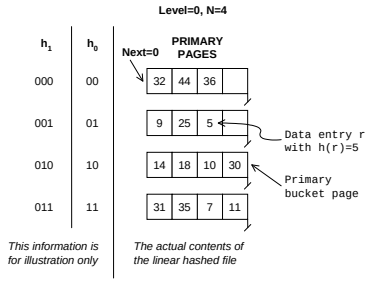


Figure 8 Example of a Linear Hashed File

Haszowanie liniowe

- Wprowadzenie nowego rekordu do pełnego zbiornika powoduje przydzielenie bloku przepełnienia
- Przydzielenie bloku przepełnienia powoduje, że zbiornik wskazywany przez indeks *Next* jest dzielony i funkcja haszowa $h_{Level+1}$ redystrybuuje rekordy tego zbiornika pomiędzy dwa zbiorniki; zbiornik oryginalny (np. b) i jego „odbicie”. „Odbicie” zbiornika b otrzymuje numer $b + N_{Level}$. Po podziale zbiornika, wartość indeksu *Next* jest zwiększana o 1. Poniżej, wprowadzenie rekordu o wartości 43 powoduje podział zbiornika

Haszowanie liniowe

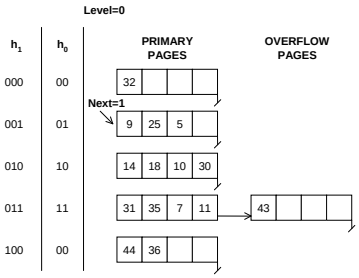


Figure 9 After Inserting Record *r* with *h(r)*=43

Haszowanie liniowe

- W dowolnej chwili wszystkie zbiorniki poniżej indeksu *Next* zostały podzielone. Plik zawiera zatem zbiorniki i ich “odbicia”. Zbiorniki od *Next* do N_{Level} nie zostały podzielone. Jeżeli stosujemy funkcję haszową h_{Level} na rekordzie i otrzymujemy liczbę *b* z przedziału *Next* do N_{Level}, to rekord należy do zbiornika *b*. Przykładowo, $h_0(18)$ wynosi 2; ponieważ wartość ta należy do przedziału *Next* (=1) i N_1 (=4), ten zbiornik nie został podzielony
- Jeżeli wartość funkcji haszowej *b* należy do przedziału 0 - *Next*, rekord może znajdować się w oryginalnym zbiorniku (b) lub jego “odbiciu” ($b + N_{Level}$); stosujemy wówczas funkcję $h_{Level+1}$ aby określić do którego zbiornika rekord należy

Haszowanie liniowe

- Przykładowo, $h_0(32)$ i $h_0(44)$ są równe 0. ponieważ *Next* jest równy 1, oznacza to, że musimy zastosować funkcję h_1. $h_1(32) = 0$ and $h_1(44) = 4$. Stąd, 32 należy do zbiornika A a 44 należy do “odbicia” zbiornika A, tj. zbiornika A2.
- Nie każde wstawienie rekordu powoduje podział zbiornika. Rozważmy wstawienie 37 do pliku z rys 9. Zbiornik nie jest pełen. Plik po wstawieniu rekordu 37 ma następującą postać przedstawioną na rys. 10.

Haszowanie liniowe

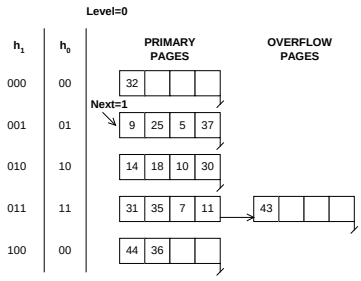


Figure 6.10 After Inserting Record *r* with *h(r)*=37

Haszowanie liniowe

- Jeżeli Next jest równy N_{Level} - 1 i nastąpił podział ostatniego zbiornika, to kończy to bieżącą rundę. Liczba zbiorników jest dwukrotnie większa niż na początku rundy i rozpoczynamy nowa rundę. Zwiększamy wartość *Level* o 1 i *Next* zerujemy 0. Rozważmy przykład przedstawiony na rys. 12 i 13. Rysunek 12 ilustruje sytuację p[o wstawieniu 22, 66 i 34. Wstawienie 50 powoduje podział ostatniego zbiornika (rys. 13).

Haszowanie liniowe

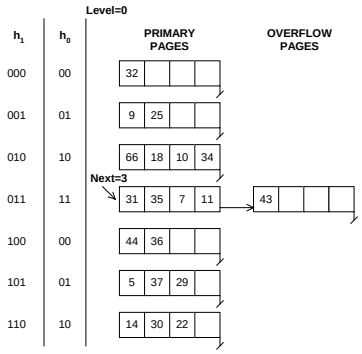


Figure 6.12 After Inserting Records with *h(r)*=22, 66 and 34

Haszowanie liniowe

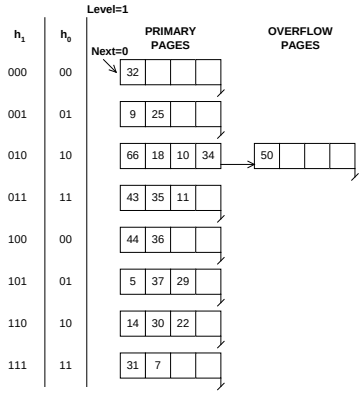


Figure 6.13 After Inserting Record *r* with *h(r)*=50