Indeksy drzewiaste

ISAM

- **ISAM** - INDEXED SEQUENTIAL ACCESS METHOD
- **Problem:**
  - Dany jest plik uporządkowany – w jaki sposób zrealizować efektywnie zapytanie z przedziałem wartości?
- **Odpowiedź:**
  - Utworzyć drugi plik, zdefiniowany na określonym atrybucie (klucz) zawierający rekordy odpowiadające wartościom kluczy pierwszych rekordów danych w pliku oryginalnym postaci:

    <pierwszy klucz na stronie, wskaźnik do strony>

    uporządkowany według wartości kluczy

ISAM

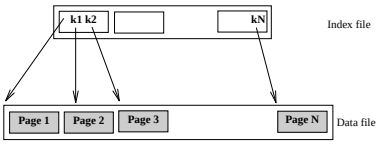
- Format strony pliku indeksowego ma następującą postać:

P0	K1	P1	K2	P2	...	Km	Pm

Parę wartości postaci *<klucz, wskaźnik>* - nazywamy *rekordem indeksu (ang. entry)*

Każda strona indeksu zawiera o jeden wskaźnik więcej aniżeli liczba kluczy – każdy klucz pełni funkcję separatora pomiędzy stronami wskazywanymi przez wskaźniki indeksu

ISAM

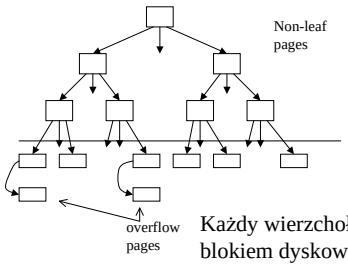


Indeks jednopoziomowy

ISAM

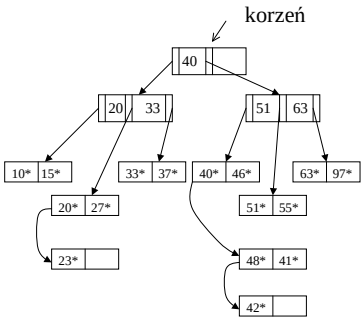
- Realizacja zapytań przedziałowych:
  - realizujemy przeszukiwanie binarne pliku indeksowego w celu zidentyfikowania strony zawierającej pierwszy klucz spełniający warunek selekcji,
  - rozpoczynamy przeglądanie pliku danych od strony wskazanej przez wskaźnik indeksu
- Przeszukiwanie binarne pliku indeksowego może być kosztowne!
- Idea ISAM: **buduj kolejne pliki indeksowe rekursywnie (indeks do indeksu, itd.)!**

ISAM



Każdy wierzchołek indeksu jest blokiem dyskowym, wszystkie rekordy danych znajdują się w Liściach (wariant (1)). Możliwy jest również wariant (2).

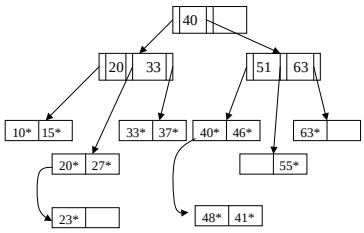
ISAM



ISAM

- Znajdź rekord danych o wartości klucza 27
- Wprowadź rekordy danych o następujących wartościach kluczy 23, 48, 41, 42 (załóż, że każdy liść może zawierać co najwyżej 2 rekordy)
- Usuń rekordy o następujących wartościach kluczy 42, 51, 97

ISAM

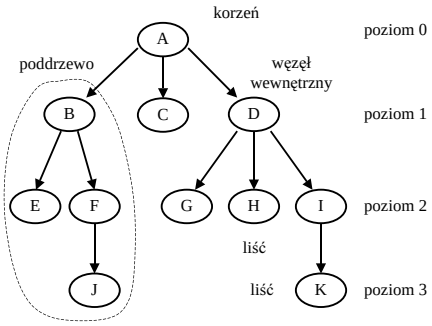


Struktura ISAM po usunięciu rekordów

ISAM

- Liczba operacji I/O jest równa liczbie poziomów drzewa i wynosi $\log_F N$, gdzie N jest liczba stron zajmowanych przez liście a F (fan-out) jest liczbą rekordów indeksu, które znajdują się na pojedynczej stronie indeksu
- Wprowadzanie i usuwanie rekordów danych dotyczy wyłącznie zawartości stron znajdujących się w liściach
- Mogą się rozwijać długie łańcuchy stron przepełnienia wraz ze wzrostem rozmiaru pliku danych (wpływa to na czas wyszukiwania rekordów)
- Struktura indeksu jest statyczna
- Nie ma potrzeby blokowania stron indeksu

Struktura drzewiasta



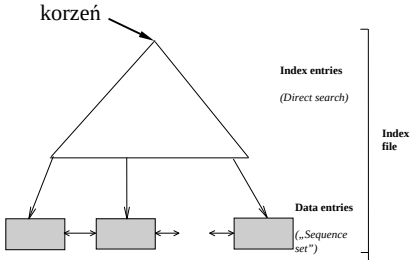
B+ drzewo: indeks o strukturze dynamicznej

- Indeks B+ drzewo jest zrównoważoną strukturą drzewiastą, w której wierzchołki wewnętrzne służą do wspomagania wyszukiwania, natomiast wierzchołki liści zawierają rekordy indeksu ze wskaźnikami do rekordów danych
- W celu zapewnienia odpowiedniej efektywności realizacji zapytań przedziałowych wierzchołki liści stanowią listę dwukierunkową. pointers

Charakterystyka indeksu B+ drzewa

- Operacje wstawiania i usuwania rekordów indeksu pozostawiają indeks zrównoważony
- Każdy wierzchołek jest wypełniony w co najmniej 50% (za wyjątkiem korzenia). Jednakże, w przypadku operacji usuwania rekordu z indeksu lokalizujemy rekord do usunięcia, usuwamy rekord i pozostawiamy indeks nie zmieniony (nie jest gwarantowane 50% wypełnienie wierzchołków), gdyż najczęściej rozmiar plików się zwiększa (rzadziej maleje)
- Wyszukanie rekordu wymaga przejścia od korzenia do liścia. Długość ścieżki od korzenia do dowolnego liścia nazywamy wysokością drzewa indeksu

Struktura indeksu B+ drzewa



Format wierzchołków (1)

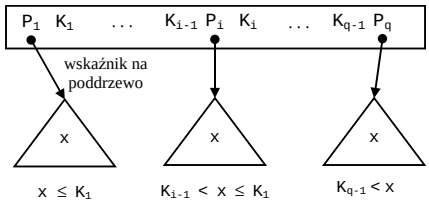
Struktura wierzchołka wewnętrznego indeksu B+-drzewo rzędu p jest następująca:

1. Wierzchołek wewnętrzny ma następującą postać:
   $<P_1, K_1, P_2, ..., P_{Q-1}, K_{Q-1}, P_Q>$
   gdzie $q \leq p$ i każdy P_i jest wskaźnikiem do poddrzewa
2. Dla każdego wierzchołka wewnętrznego zachodzi
   $K_1 < K_2 < ... < K_{Q-1}$
3. Dla wszystkich wartości X wyszukiwanych w poddrzewie wskazywanym przez P_i, zachodzi:
   $K_{i-1} < X \leq K_i$, for $1 < i < Q$
   $X \leq K_i$, for $i = 1$
   $K_{i-1} < X$ for $i = Q$

Format wierzchołków (2)

4. Każdy wierzchołek wewnętrzny posiada co najwyżej p wskaźników do poddrzew
5. Każdy wierzchołek wewnętrzny, za wyjątkiem korzenia, posiada co najmniej $\lceil (p/2) \rceil$ wskaźników do poddrzew. Korzeń posiada co najmniej 2 wskaźniki do poddrzew jeżeli jest wierzchołkiem wewnętrznym
6. Każdy wierzchołek wewnętrzny o Q wskaźnikach posiada Q-1 wartości kluczy

Wierzchołek wewnętrzny



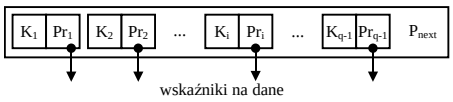
Format wierzchołków (3)

Struktura wierzchołka liścia indeksu B+-drzewo rzędu p jest następująca:

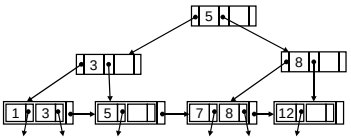
1. Wierzchołek liścia ma następującą postać:
   $<P_{prev}, <K_1, Pr_1>, <K_2, Pr_2>, ..., <K_{Q-1}, P_{Q-1}>, P_{next}>$
   gdzie $q \leq p$ i każdy Pr_i jest wskaźnikiem do bloku danych.
2. Dla każdego wierzchołka liścia zachodzi:
   $K_1 < K_2 < ... < K_{Q-1}$
3. Każdy wierzchołek liść posiada co najmniej $\lfloor (p/2) \rfloor$ wartości kluczy

Format wierzchołków (4)

4. Pr_i jest wskaźnikiem do bloku danych zawierającego rekord danych o wartości klucza K_i lub wskaźnikiem do bloku danych zawierającego rekord (lub blok wskaźników wskazujących na rekordy danych o wartości klucza K_i)
5. Wszystkie liście znajdują się na tym samym poziomie (tej samej wysokości)



Przykładowe B+ drzewo



Jak obliczyć rząd indeksu p

- **Dane**: rozmiar klucza V, rozmiar wskaźnika do bloku P, rozmiar bloku B, liczba rekordów w indeksowanym pliku danych r i liczba bloków pliku b
- Węzły wewnętrzne zawierają maksymalnie p wskaźników i p-1 kluczy. Każdy z węzłów musi zmieścić się w pojedynczym bloku dyskowym. Stąd rząd B^+-drzewa p jest największą liczbą całkowitą, dla której spełniona jest nierówność:

  $(p*P)+((p-1)*V)) \leq B$

  Minimalna wysokość indeksu rzadkiego: $h = \lceil \log_p b \rceil$

  Minimalna wysokość indeksu gęstego: $h = \lceil \log_p r \rceil$

Przykład

- **Dane:**

Rozmiar pliku: r = 30 000 rekordów

Rozmiar bloku: B = 1KB

Rozmiar rekordu: R = 100 bajtów

Rekordy mają stałą długość i nie są dzielone między bloki

Rozmiar klucza: V = 9

Rozmiar wskaźnika: P = 6

Indeks wtórny

Przykład

- **Obliczenia:**

Rząd indeksu:

$$(p*P)+((p-1)*V)) \leq B$$
$$p = \lfloor (B+P)/(V+P) \rfloor$$
$$p = \lfloor (1024+6)/(9+6) \rfloor$$
$$p = 68$$

Minimalna wysokość indeksu:

$$h = \lceil \log_p r \rceil$$
$$h = \log_{68} 30\,000$$
$$h = 3$$

Przykład

- Koszt wyszukiwania rekordu bez pomocy indeksu:
  liczba bloków/2 = **1500** dostępów
- Koszt wyszukiwania danych za pomocą indeksu jednopoziomowego gęstego:
  9 dostępów do indeksu+ 1 dostęp do danych =10 dostępów
- Koszt wyszukiwania danych za pomocą indeksu typu B^+-drzewo:
  wysokość drzewa + 1 = 3 + 1 = **4** dostępy

Wyszukiwanie danych

dany indeks o rzędzie *p* i korzeniu *n*
Szukaj (K)
begin
    n := wskaźnik na blok zawierający korzeń drzewa;
    odczytaj blok n;
    while (węzeł n nie jest liściem) do
        begin
        q := liczba wskaźników w węźle n;
        if K ≤ n.K_1 (*n.K_1 jest i-tym kluczem w węźle n*)
            then n := n.P_1 ;(*n.P_1 jest i-tym wskaźnikiem w węźle n*)
            else if K > n.K_{q-i}
                then n := n.P_q ;
                else begin
                    znajdź w węźle n element i taki, że n.K_{i-1} < K < n.K_i
                    n := n.P_i ;
                    end;
        odczytaj blok n;
        end;
    znajdź w węźle n element (K_i,Pr_i) taki, że K_i = K; (* przeglądaj liść drzewa *)
    if znaleziony
        then odczytaj blok pliku danych o wskaźniku Pr_i i pobierz rekord danych
        else w pliku danych nie ma rekordu o wartości pola równej K;
end Szukaj

Wstawianie danych (1)

dany indeks o rzędzie p i korzeniu n
Wstaw (K, Pr)
begin
    n := wskaźnik na blok zawierający korzeń drzewa;
    czytaj blok n;
    wyczyść stos S;
    while (węzeł n nie jest liściem) **do**
        begin
        umieść wartość n na stosie S;
        (*stos S przechowuje ścieżkę przeszukiwania drzewa*)
        q := liczba wskaźników w węźle n;
        if K ≤ n.K_1 (*n.K_1 jest i-tym kluczem w węźle n*)
            then n := n.P_1 ;(*n.P_1 jest i-tym wskaźnikiem w węźle n*)
            else if K > n.K_{q-i}
                then n := n.P_q ;
                else begin
                    znajdź w węźle n element i taki, że n.K_{i-1} < K < n.K_i
                    n := n.P_i ;
                    end;
        odczytaj blok n;
        end;
    utwórz nowy element indeksu (K,Pr);
    if liść n nie jest pełny
    then wstaw nowy element (K, Pr) w odpowiednią pozycję węzła n;
    else

Wstawianie danych (2)

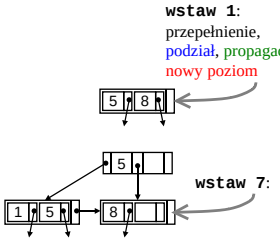
begin (*węzeł n jest pełny – niezbędny podział liścia*)
    kopiuj n do temp; (*temp jest temporalnym blokiem o rozmiarze umożliwiającym przechowanie dodatkowego elementu *);
    wstaw element (K, Pr) w odpowiednie miejsce temp;
    (*temp przechowuje teraz p+1 elementów *)
    new := nowy pusty liść drzewa;
    j := $\lceil (p + 1)/2 \rceil$;
    n := pierwszych j elementów temp (do elementu (K_j,Pr_j));
    n.P_{next} := new;
    new := pozostałe elementy z temp;
    K := Kj;
    (*teraz musimy wstawić element (K,new) do węzła pośredniego, który jest przodkiem liścia n, jeżeli przodek jest pełny nastąpi propagacja podziału*)
    finished := false;
    repeat
        if stos S jest pusty
        then (*nie ma przodka – trzeba utworzyć nowy korzeń drzewa*)
            begin
                root := nowy węzeł wewnętrzny indeksu;
                root := <n, K, new>;
                finished := true;
            end
        else

Wstawianie danych (3)

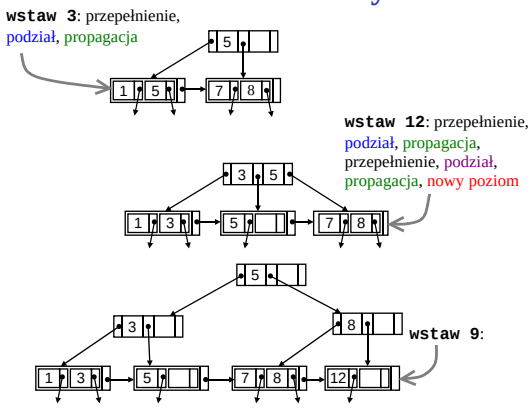
            begin
            n := wskaźnik ze stosu S;
            if węzeł wewnętrzny n nie jest pełny
            then
                begin (*podział węzła niepotrzebny*)
                wstaw (K, new) w odpowiednie miejsce węzła n;
                finished := true;
                end
            else
                begin (*węzeł wewnętrzny jest pełny – niezbędny podział*)
                przepisz zawartość n do temp;
                wstaw (K,new) w odpowiednią pozycję do temp;
    (*temp ma teraz p+1 wskaźników*)
        new := nowy węzeł wewnętrzny indeksu;
        j := $\lfloor ((p + 1)/2 \rfloor$;
        n := pierwszych j elementów temp (do wskaźnika P_j);
            (*n zawiera <P_1, K_1, P_2, K_2, ..., P_{j-1}, K_{j-1},P_j) > *)
        new := pozostałe elementy temp;
            (*new zawiera <P_{j+1}, K_{j+1}, ..., K_{p-1}, P_p, K_p,P_{p+1}) > *)
        K := Kj ;
            (*teraz musimy wstawić element (K,new) do przodka węzła n*)
        end ;
        end ;
    until finished
    end;
    end;
end Wstaw;

Przykład operacji wstawiania do indeksu

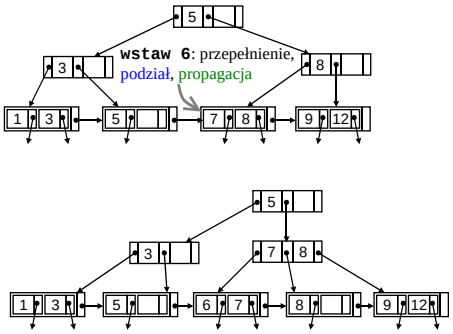
- Sekwencja operacji wstawiania: 8, 5, 1, 7, 3, 12, 9, 6



Przykład c.d.



Przykład c.d.



Indeksy wielowymiarowe

- Indeks B+ drzewa wspiera efektywną realizację zapytań przedziałowych dzięki liniowemu uporządkowaniu rekordów indeksu
- Inne sposoby porządkowania rekordów indeksu prowadzą do innych struktur danych:
  - Indeksy jednowymiarowe: liniowy porządek zdefiniowany na zbiorze kluczy (i zbiorze rekordów indeksu). Przykładem takiego indeksu – indeks B+ drzewo
  - Indeksy wielowymiarowe: brak liniowego porządku. Rekordy indeksu uporządkowane wg. relacji przestrzennej, w której każda wartość klucza jest punktem w przestrzeni k-wymiarowej, gdzie *k* jest liczbą atrybutów klucza złożonego

Indeksy wielowymiarowe

- **Przykład:** Indeks jednowymiarowy linearyzuje 2-wymiarową przestrzeń wartości klucza złożonego *<age, sal>* sortując rekordy indeksu najpierw wg. wartości atrybutu *age*, a następnie wg. wartości atrybutu *sal*. Przykładowo, klucze *<age, sal>* są przechowywane w indeksie typu B+-drzewo w następującym porządku: *<11, 80>*, *<12, 10>*, *<12, 20>*, *<13, 75>*
- Zapytanie przedziałowe na atrybucie (*age*) może być zrealizowane efektywnie ze względu na porządek rekordów indeksu. Zapytanie przedziałowe na atrybucie *sal* może być kosztowne. Przykładowo, wyszukanie rekordów dla których *age* < 12 jest łatwe, ale wyszukanie rekordów, dla których *sal* < 20 wymaga przeszukania całego indeksu

Indeksy wielowymiarowe

- W indeksie wielowymiarowym rekordy indeksu są przechowywane w porządku wynikającym z odległości pomiędzy wartościami kluczy w przestrzeni k wymiarowej: *<12, 10>* i *<12, 20>* są pamiętane kolejno, podobnie klucze *<11, 80>* i *<13, 75>*