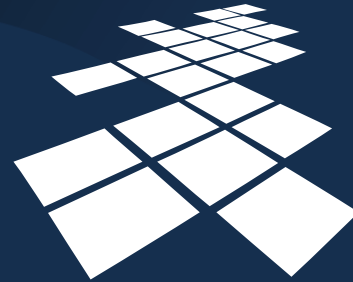


# Organizacja plików

Wykład przygotował:  
Robert Wrembel



**UCZELNIA**  
ONLINE



## Plan wykładu

- Struktura przechowywania danych i organizacja rekordów w blokach
- Rodzaje organizacji plików
  - pliki nieuporządkowane
  - pliki uporządkowane
  - pliki haszowe

Celem wykładu jest przedstawienie podstawowych organizacji plików. W ramach wykładu zostaną omówione następujące zagadnienia:

- struktura przechowywania danych i organizacja rekordów w blokach,
- rodzaje organizacji plików, czyli pliki nieuporządkowane, uporządkowane i haszowe.

Każda z organizacji plików zostanie scharakteryzowana strukturą, mechanizmami dostępu i kosztami dostępu. Ponadto, dla plików haszowych zostaną przedstawione podstawowe techniki rozwiązywania kolizji.



## Wprowadzenie (1)

- Organizacja pliku
  - sposób uporządkowania rekordów w pliku przechowywanym na dysku
  - wspiera wykonywanie operacji na pliku
- Wybór odpowiedniej organizacji zależy od sposobu użytkowania danego pliku
  - wyszukiwanie rekordów opisujących zatrudnionych pracowników w porządku alfabetycznym ⇒ sortowanie pliku według nazwisk
  - wyszukiwanie rekordów opisujących zatrudnionych, których zarobki są w podanym zakresie ⇒ sortowanie nie jest właściwe
  - wybór odpowiedniej organizacji dla każdego pliku ⇒ administrator

Organizacja pliku określa sposób uporządkowania rekordów w pliku przechowywanym na dysku. Wybór właściwej organizacji zależy od sposobu użytkowania danego pliku.

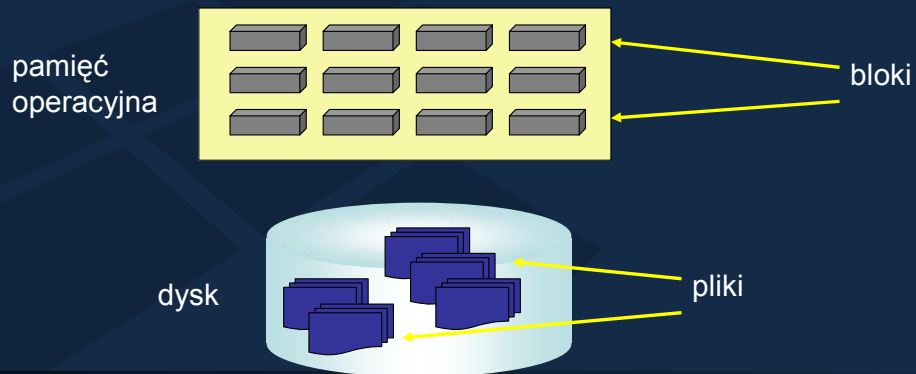
Przykładowo, jeśli chcemy wyszukiwać rekordy opisujące zatrudnionych pracowników w porządku alfabetycznym, sortowanie pliku według nazwisk jest dobrą organizacją pliku. Z drugiej strony, jeśli chcemy wyszukiwać rekordy opisujące zatrudnionych, których zarobki są w podanym zakresie, sortowanie rekordów pracowników według nazwisk nie jest właściwą organizacją pliku.

Wybranie właściwej organizacji dla każdego pliku jest zadaniem administratora BD.



## Wprowadzenie (2)

- Media fizyczne tworzą hierarchię pamięci składającą się z:
  - pamięci operacyjnej o organizacji blokowej
  - pamięci zewnętrznej o organizacji plikowej



BD – wykład 5 (4)

Media fizyczne tworzą hierarchię pamięci składającą się z pamięci operacyjnej i pamięci zewnętrznej. Pamięć zewnętrzna ma organizację plikową, oznacza to, że jednostką alokacji na dysku jest plik. Natomiast pamięć operacyjna ma organizację blokową. Oznacza to, że jednostką alokacji jest blok. Blok alokowany w pamięci operacyjnej jest wielokrotnością rozmiaru fizycznego bloku dyskowego.



## Wprowadzenie (3)

- Trwałe dane w BD są przechowywane w pamięci zewnętrznej:
  - ze względu na rozmiar danych
  - odporność pamięci zewnętrznej na awarie
  - niski koszt przechowywania
- Buforowanie bloków dyskowych

Trwałe dane w bazie danych są przechowywane w pamięci zewnętrznej z trzech powodów:

- ze względu na rozmiar bazy danych
- odporność pamięci zewnętrznej na awarie
- koszt jednostkowy

W czasie pracy bazy danych, poszukiwane dane są odczytywane z plików dyskowych i umieszczane/buforowane w blokach systemu operacyjnego. Bloki te są często nazywane buforami bazy danych. Stąd dane są następnie udostępniane użytkownikom BD. Zapis danych na dysk również odbywa się za pośrednictwem buforów bazy danych.

Użytkownicy modyfikują dane w buforach. Zawartość tych buforów jest następnie zapisywana do plików.



## Wprowadzenie (4)

- Alokacja danych w plikach ⇔ rekordy
  - rekordy proste / złożone
  - rekordy o stałej / zmiennej długości
- Na poziomie dyskowym rekordy przechowywane w blokach dyskowych

Dane w plikach są reprezentowane w postaci rekordów. Każdy rekord składa się z pól przechowujących wartości.

Wyróżnia się rekordy o strukturze prostej i złożonej (zagnieżdżonej). W pierwszym przypadku, wartością każdego pola rekordu jest wartość elementarna (liczba, łańcuch znaków, data, ciąg bitów). W drugim przypadku, wartością pola rekordu jest inny rekord.

Rekordy mogą mieć stałą długość lub zmienną. Stała długość oznacza, że rekord zawsze zajmuje tyle samo miejsca na dysku, niezależnie od rzeczywistych rozmiarów przechowywanych w nim danych. Rekordy o stałej długości przyjmują zawsze rozmiar będącą sumą maksymalnych zadeklarowanych rozmiarów ich atrybutów. Rekordy o zmiennej długości przyjmują taki rozmiar jaki faktycznie przyjmują przechowywane w nich dane.

Na poziomie dyskowym, rekordy są przechowywane w blokach dyskowych. Rozmiar tych bloków jest określany przez system operacyjny. Często jest to 512B.



## Współczynnik blokowania

- W bloku dyskowym lub bloku pamięci operacyjnej mieści się niecałkowita liczba rekordów
- Współczynnik blokowania  $\Rightarrow$  **bfr**
  - rozmiar bloku B
  - rozmiar rekordu R
  - współczynnik blokowania **bfr =  $\lfloor (B/R) \rfloor$**  1
  - wolny obszar w bloku:  **$B - (bfr * R)$**  2

W praktyce, w jednym bloku dyskowym lub bloku pamięci operacyjnej mieści się niecałkowita liczba rekordów. Wynika to z faktu, że rozmiar bloku nigdy nie jest wielokrotnością rozmiaru rekordu. Maksymalna liczba rekordów, która może się zmieścić w bloku jest określana za pomocą tzw. współczynnika blokowania (blocking factor) - bfr. Jest on definiowany następująco.

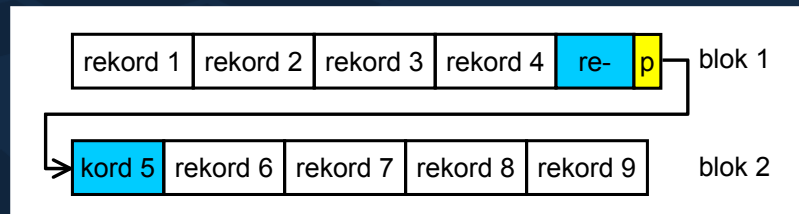
Niech B oznacza rozmiar bloku dyskowego, R - oznacza rozmiar rekordu. Dla uproszczenia zakłada się że albo rozważane są rekordy o stałej długości R bajtów, albo dla rekordów o zmiennej długości R oznacza maksymalny rozmiar rekordu. Współczynnik blokowania jest określany jako największa liczba całkowita mniejsza od ilorazu rozmiaru bloku i rozmiaru rekordu. Wyraża to wzór nr 1 ze slajdu.

Przyjmując taką definicję współczynnika blokowania, łatwo jest obliczyć wolny obszar jaki pozostaje w każdym z bloków. Jest on wyrażony wzorem:  $B - (bfr * R)$ . W przypadku rekordów o zmiennej długości jest to obszar minimalny.



## Organizacja rekordów w blokach (1)

- Dzielona (spanned)



Rekordy w blokach są zorganizowane albo jako dzielone (ang. spanned) albo jako niepodzielne (ang. unspanned). W pierwszym przypadku, rekord, który cały nie mieści się w bloku jest dzielony, a jego części są przechowywane w kilku blokach.

Poglądowy schemat dzielonej organizacji rekordów przedstawiono na slajdzie. Rekord 5 nie mieści się w bloku 1, więc jest dzielony na 2 części. Początek rekordu znajduje się w bloku 1, a koniec rekordu - w bloku 2.

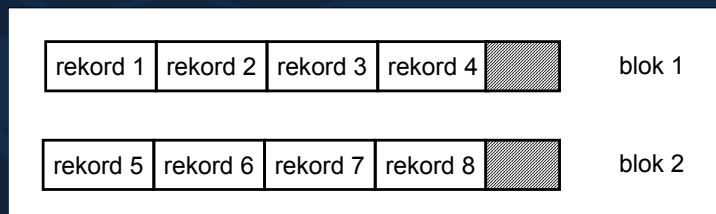
Organizacja dzielona zapewnia lepsze wykorzystanie miejsca w blokach - wszystkie bloki są w całości wypełnione.





## Organizacja rekordów w blokach (2)

- Niepodzielna (unspanned)



W przypadku organizacji niepodzielnej rekord, który nie mieści się w całości w bloku jest przenoszony do takiego bloku, w którym zmieści się cały.

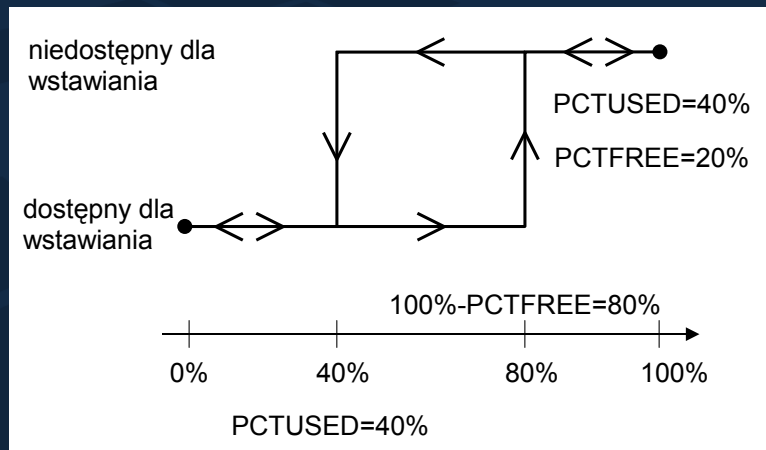
Poglądowy schemat niepodzielnej organizacji rekordów przedstawia slajd. W tym przykładzie, rekord 5 w całości został przeniesiony do bloku 2.

Organizacja niepodzielna powoduje, że w blokach pozostaje niewykorzystane miejsce.



## Zarządzanie rozmiarem bloku danych

- Utrzymywanie wolnej pamięci w bloku dla potencjalnych modyfikacji



W praktyce, w każdym bloku, niezależnie od organizacji rekordów, pozostawia się część wolnego miejsca. Miejsce to jest wykorzystywane na rozszerzanie się rekordów na skutek modyfikowania wartości ich pól. Gdyby nie było wolnego miejsca w bloku, wówczas po modyfikacji rekord mógłby się nie zmieścić w bloku i musiałby zostać przesunięty do innego bloku.

Tę technikę stosuje m.in. SZBD Oracle. W systemie tym, dla bloków definiuje się dwa parametry PCTFREE i PCTUSED. Pierwszy z nich określa ile procent rozmiaru bloku pozostanie wolne. PCTUSED określa kiedy do bloku można wstawiać rekordy. Parametr ten jest wyrażony również procentowym rozmiarem bloku. Jeśli zajętość bloku przewyższa wartość PCTUSED, wówczas blok nie jest wykorzystywany do wstawiania nowych rekordów. Jeśli natomiast zajętość bloku jest niższa niż PCTUSED, wówczas do bloku można wstawiać rekordy.

W przykładzie na slajdzie PCTFREE=20%, a PCTUSED=40%. Oznacza to, że w bloku zawsze pozostaje przynajmniej 20% wolnego miejsca. Do bloku wolno wstawiać dane jeżeli aktualna jego zajętość nie przekracza 40% jego rozmiaru. Jeżeli na skutek wstawiania nowych rekordów do bloku, zajętość wzrośnie powyżej 40%, SZBD przestaje wstawiać rekordy do tego bloku. Jeżeli na skutek usuwania rekordów zajętość bloku spadnie poniżej 40%, blok ten ponownie zostanie wykorzystany do wstawiania rekordów.



## Operacje na plikach

- Operacje dostępu do pojedynczego rekordu (record-at-a-time)
  - wyszukiwanie: Find, FindFirst, FindNext
  - usuwanie: Delete
  - aktualizacja: Update
  - wstawianie: Insert
- Operacje dostępu do zbioru rekordów (set-at-a-time)
  - wyszukiwanie wszystkich rekordów spełniających kryterium: FindAll
  - wyszukiwanie z sortowaniem: FindOrdered
  - reorganizacja pliku: Reorganize

Ponieważ rekordy są fizycznie przechowywane w plikach, więc dostęp do rekordów musi być realizowany za pomocą dedykowanych i zoptymalizowanych operacji.

Wyróżnia się operacje na pojedynczym rekordzie (ang. record-at-a-time)

i operacje na zbiorze rekordów (ang. set-at-a-time). Do pierwszej grupy zalicza się:

- wyszukiwanie rekordu spełniającego kryterium: Find, FindFirst, FindNext,
- usuwanie: Delete,
- aktualizację: Update,
- wstawianie: Insert.

Do drugiej grupy zalicza się:

- wyszukiwanie wszystkich rekordów spełniających kryterium: FindAll,
- wyszukiwanie rekordów spełniających kryterium z sortowaniem wyników: FindOrdered,
- reorganizację pliku: Reorganize (np. posortowanie rekordów wg nowego kryterium).



## Model kosztów

- Notacja i założenia:
  - $N$   $\Leftrightarrow$  liczba bloków
  - każdy blok zawiera  $R$  rekordów
  - średni czas odczytu/zapisu bloku dyskowego wynosi  $D$
  - średni czas przetwarzania rekordu wynosi  $C$
  - dla plików haszowych czas obliczenia wartości funkcji haszowej wynosi  $H$
- Typowe wartości wynoszą:
  - $D = 15$  ms
  - $C$  i  $H$  od 1 do 10  $\mu$ s
  - dominuje koszt I/O

Każda operacja na pliku posiada swój tzw. koszt, który jest oczywiście zależny od organizacji wewnętrznej pliku. Koszt jest konkretną wartością, której miarą może być np. czas wykonania, liczbaostępów do dysku. Koszt jest wartością wynikającą z tzw. modelu kosztów (ang. cost model).

W celach analizy kosztów dostępu do plików nieuporządkowanych, uporządkowanych i haszowych przyjmujemy następujący model kosztów.

Niech:

- $N$  oznacza liczbę bloków;
- każdy blok zawiera  $R$  rekordów;
- średni czas odczytu/zapisu bloku dyskowego wynosi  $D$ ;
- średni czas przetwarzania rekordu (np., porównanie wartości atrybutu ze stałą) wynosi  $C$ ;
- w przypadku plików haszowych stosujemy funkcję haszową odwzorowującą wartości rekordów na liczby naturalne; czas obliczenia wartości funkcji haszowej wynosi  $H$ ;

Typowe wartości wymienionych parametrów wynoszą  $D = 15$  ms,  $C$  i  $H$  od 1 do 10  $\mu$ s. Jak widać, czas dostępu do dysku (I/O) jest tu dominującym.



## Rodzaje organizacji plików

- Pliki nieuporządkowane (unordered files, heap files)
- Pliki uporządkowane (ordered files)
- Pliki haszowe (hash files)

Omówione operacje na plikach są implementowane i optymalizowane w różny sposób, zależny od organizacji wewnętrznej samego pliku. Ze względu na organizację wewnętrzną wyróżnia się: pliki nieuporządkowane (ang. unordered files, heap files), pliki uporządkowane (ang. ordered files) i pliki haszowe (ang. hash files).



## Plik nieuporządkowany

- Nagłówek pliku zawierający wskaźnik do bloku danych
- Blok danych zawiera wskaźnik do bloku następnego i poprzedniego
- Rekordy wstawiane na koniec pliku



W pliku nieuporządkowanym rekordy są przechowywane w kolejności ich wstawiania. Nagłówek pliku zawiera wskaźnik do pierwszego bloku danych. Bloki danych tworzą listę dwukierunkową. Oznacza to, że blok poprzedni posiada wskaźnik do bloku następnego, a blok następny posiada wskaźnik do bloku poprzedniego. W pliku nieuporządkowanym rekordy są wstawiane zawsze na koniec pliku.



## Plik nieuporządkowany - operacje (1)

Podstawową organizacją pliku nieuporządkowanego jest stóg (ang. heap)

- Operacje:
  - wstawianie rekordu
    - rekord jest wstawiany do ostatniego bloku pliku; blok ten jest zapisywany na dysk
    - koszt =  $2D+C$
  - wyszukiwanie rekordu:
    - konieczność liniowego przeszukiwania wszystkich bloków
    - średni koszt =  $0.5N(D+RC)$ . 1
    - maksymalny koszt =  $N(D+RC)$  (przejrzenie 2 całego pliku)

Podstawową organizacją pliku nieuporządkowanego jest stóg (ang. heap).

W przypadku operacji wstawiania rekordu, rekord trafia do ostatniego bloku pliku. Przed wstawieniem, blok ten musi zostać odczytany z dysku do bufora pamięci operacyjnej. Tu rekord jest wstawiany i blok ten jest z powrotem zapisywany na dysk. W konsekwencji, koszt wstawienia rekordu wynosi  $2D+C$ . Składnik  $2D$  to odczyt ostatniego bloku z dysku i jego ponowny zapis.  $C$  to koszt przetworzenia rekordu w buforze.

W przypadku operacji wyszukania rekordu spełniającego wskazane kryterium, zachodzi konieczność liniowego przeszukiwania wszystkich bloków. Średni koszt liniowego przeszukania jest wyrażony wzorem nr 1 ze slajdu ( $0.5N(D+RC)$ ). Średnio, odszukanie rekordu wymaga odczytu połowy bloków - stąd składnik kosztu  $0.5ND$ . Odczytane rekordy są następnie przetwarzane w buforze (sprawdzone jest czy rekordy spełniają kryterium poszukiwania), stąd składnik kosztu  $0.5NRC$ .

W przypadku najgorszym, tj. albo jeżeli nie ma rekordów spełniających warunek selekcji albo poszukiwane rekordy znajdują się w ostatnim bloku, system musi przejrzeć cały plik. W tym przypadku koszt jest wyrażony wzorem nr 2 ( $N(D+RC)$ ).



## Plik nieuporządkowany - operacje (2)

- przeglądanie pliku
  - koszt =  $N(D + RC)$  1
  - odczyt  $N$  stron z kosztem  $D$
  - dla każdej strony przetworzyć  $R$  rekordów z kosztem  $C$
- wyszukiwanie z przedziałem wartości
  - odczyt wszystkich bloków
  - koszt =  $N(D + RC)$  2

W przypadku operacji przeglądania całego pliku koszt jest wyrażony wzorem nr 1 ze slajdu ( $N(D + RC)$ ) ponieważ trzeba odczytać  $N$  stron ( $D$  koszt odczytu strony) i dla każdej strony trzeba przetworzyć  $R$  rekordów ( $C$  koszt przetworzenia pojedynczego rekordu).

Wyszukiwanie z przedziałem wartości również wymaga przeszukania całego pliku. W związku z tym, koszt tak jak poprzednio jest wyrażony wzorem nr 2 ( $N(D + RC)$ ).





## Plik nieuporządkowany - operacje (3)

### – usuwanie rekordu

- liniowe przeszukiwanie i zapis bloku na dysk
- koszt wyszukania + koszt (C + D)
  - średnio:  $0.5N(D+RC)+(C+D)$  **1**
  - maksymalnie:  $N(D+RC)+(C+D)$  **2**
- pozostaje zwolniona pamięć  $\Rightarrow$  konieczność okresowej reorganizacji pliku

### – sortowanie

- trudne
- stosowane tzw. sortowanie zewnętrzne
  - plik sortowany fragmentami mieszczącymi się w pamięci operacyjnej
  - fragmenty są łączone w większe w kolejnych przebiegach algorytmu sortowania

Usunięcie rekordu wymaga najpierw jego odszukania, stąd konieczne jest liniowe przeszukiwanie pliku, odczyt do bufora bloku zawierającego usuwany rekord i zapis tego bloku na dysk już po usunięciu rekordu. Stąd, na całkowity koszt operacji usunięcia rekordu składa się koszt wyszukania rekordu oraz koszt przetworzenia (czyli usunięcia) rekordu i koszt zapisu bloku na dysk. Średni koszt usunięcia rekordu jest wyrażony wzorem nr 1:  $0.5N(D+RC) + (C + D)$ , a maksymalny koszt - wzorem nr 2:  $N(D+RC) + (C + D)$ .

Ponadto, przy częstym usuwaniu rekordów, w blokach pozostaje zwolniona pamięć. Konieczna jest zatem okresowa reorganizacja pliku w celu odzyskania tej pamięci.

Ze względów efektywnościowych, sortowanie należy wykonywać w pamięci operacyjnej. Sortowanie dużych plików jest zagadnieniem trudnym implementacyjnie. W praktyce bowiem, rozmiar pliku jest znacznie większy niż rozmiar dostępnej pamięci operacyjnej. Technika sortowania stosowaną najczęściej jest tzw. sortowanie zewnętrzne (ang. external sorting). Koncepcyjnie, polega ono na sortowaniu pliku fragmentami, które mieszczą się w pamięci operacyjnej. Każdy posortowany fragment jest w drugiej fazie sortowania łączony z innymi fragmentami. Łączenie to wymaga zwykle wielu przebiegów.



## Plik nieuporządkowany - charakterystyka

- Efektywne wstawianie pojedynczych rekordów i dużych zbiorów rekordów
- Efektywne pozostałe operacji w przypadku plików o rozmiarze kilku bloków
- Struktura właściwa dla odczytu wszystkich rekordów
- Struktura stosowana z innymi strukturami dostępu do danych (np. indeksy)

Charakterystyka dostępu do pliku nieuporządkowanego jest następująca:

Plik taki umożliwia efektywne wstawianie pojedynczych rekordów i dużych zbiorów rekordów. Pliki o rozmiarze kilku bloków są również efektywne dla pozostałych operacji tj. wyszukiwania rekordów, modyfikowania i usuwania. Plik nieuporządkowany można stosować w przypadkach, gdy są często czytane wszystkie rekordy. Ponadto, jest to struktura stosowana z innymi strukturami dostępu do danych (np. indeksami).



## Plik uporządkowany

- Rekordy pliku są porządkowane według pola porządkującego (ang. ordering field)

Aaron Brian			
Abbott Jim			
Acosta Phil			
Adams Jim			
Adams Robin			
Adler John			
...			
Winslet Kathy			
Zobel Mick			
Zezula Pavol			

Drugim omawianym rodzajem organizacji plików jest plik uporządkowany. W pliku takim, rekordy w nim składowane są uporządkowane wg wartości tzw. pola porządkującego (ang. ordering field). W przykładowym pliku ze slajdu rekordy zostały uporządkowane wg wartości nazwiska i imienia.



## Plik uporządkowany - operacje (1)

- Operacje:
  - przeglądanie pliku
    - koszt =  $N(D+RC)$  **1**
    - ponieważ wszystkie strony muszą być odczytane
  - wyszukiwanie rekordu
    - wyszukiwanie binarne
    - koszt =  $D \cdot \log_2 B + C \cdot \log_2 R$  **2**
  - wyszukiwanie z przedziałem wartości
    - koszt wyszukania pierwszego rekordu + koszt selekcji zbioru rekordów

Operacja przeglądnięcia całego pliku wymaga odczytu wszystkich bloków danych, stąd jej koszt jest wyrażony wzorem 1:  $N(D+RC)$ , podobnie jak dla pliku nieuporządkowanego.

Wyszukanie rekordu spełniającego warunek selekcji jest realizowane z wykorzystaniem algorytmu wyszukiwania binarnego (połowienia binarnego). Stąd jego koszt jest wyrażony wzorem 2:  $D \cdot \log_2 B + C \cdot \log_2 R$ .

Operacja wyszukania rekordów spełniających warunek z zadanego przedziału wymaga odszukania pierwszego rekordu spełniającego warunek lewej strony przedziału, a następnie odczytania kolejnych rekordów aż do ostatniego rekordu spełniającego warunek prawej strony przedziału. Koszt odszukania pierwszego rekordu to:  $D \cdot \log_2 B + C \cdot \log_2 R$ . Koszt odczytania kolejnych rekordów to:  $ND$ , gdzie  $N$  jest liczbą bloków, w których te rekordy się znajdują.



## Plik uporządkowany - operacje (2)

- wstawianie rekordu
  - koszt wyszukania miejsca wstawienia rekordu
  - koszt przepisania średnio połowy rekordów
  - koszt całkowity =  $2*(0.5N(D+RC))$  **1**
- usuwanie rekordu
  - koszt znalezienia usuwanego rekordu
  - koszt przepisania średnio połowy rekordów
  - identyczny jak koszt wstawiania
  - koszt całkowity =  $2*(0.5N(D+RC))$ ; **2**

Wstawianie i usuwanie rekordu są operacjami kosztownymi, ponieważ po zakończeniu tych operacji rekordy muszą pozostać posortowane.

Wstawienie rekordu wymaga: po pierwsze, znalezienia właściwej pozycji w pliku na wstawienie rekordu, zgodnie z wartością atrybutu porządkującego. Po drugie, wymaga utworzenia pustego miejsca w pliku, w które zostanie wstawiony nowy rekord. Operacja utworzenia pustego miejsca wymaga średnio przesunięcia (przepisania) połowy rekordów w miejsce o 1 dalej w pliku. Koszt całkowity operacji wstawienia rekordu jest wyrażony wzorem 1.

Operacja usunięcia rekordu wymaga: po pierwsze znalezienia usuwanego rekordu, i po drugie, oznaczenia tego rekordu jako usunięty. Miejsce po usuniętym rekordzie pozostaje w pliku i nie jest wykorzystywane. W wyniku wielu operacji usunięcia, w pliku będzie wiele pustych miejsc, co ze względów efektywnościowych nie będzie dobre. Z tego względu, plik należy okresowo reorganizować, co jest operacją bardzo kosztowną. W skrajnym przypadku wymaga to przesunięcia wszystkich rekordów. Koszt całkowity operacji usunięcia rekordu jest wyrażony wzorem 2.



## Plik uporządkowany - rozwiązanie problemu wstawiania rekordów

- Przesuwanie średnio połowy rekordów  $\Rightarrow$  nieefektywne
- Pozostawienie wolnej pamięci w każdym bloku na wstawiane rekordy
  - wstawienie rekordu wymaga przesunięcia rekordów tylko w ramach bieżącego bloku
- Tworzenie nieuporządkowanego pliku rekordów, a następnie łączenie go z plikiem głównym
  - efektywne wstawianie
  - nieefektywne wyszukiwanie

Problem wstawiania rekordów można rozwiązać na cztery sposoby. Pierwszym jest omówione wcześniej przesuwanie średnio połowy rekordów w pliku, w celu zagwarantowania właściwego porządku rekordów w pliku.

Drugie rozwiązanie zakłada pozostawienie w każdym bloku dyskowym pustego miejsca na wstawiane rekordy. W tym przypadku przesunięcia rekordów należy dokonać tylko w tym bloku, do którego jest wstawiany rekord.

Trzeci sposób wykorzystuje nieuporządkowany tymczasowy plik, zwany nadmiarowym (ang. overflow file) lub transakcyjnym (ang. transaction file). Plik uporządkowany jest nazywany plikiem master (ang. master file) lub głównym (ang. main file). Wstawiane rekordy są umieszczane w pliku nadmiarowym, na jego końcu. Okresowo, plik nadmiarowy jest sortowany i scalany z plikiem głównym. Przy takim podejściu wstawianie jest efektywne, ale wyszukiwanie jest kosztowne. Wymaga ono przeszukania pliku głównego metodą połowienia binarnego. Następnie plik nadmiarowy jest przeszukiwany z wykorzystaniem pełnego jego odczytu. Dla aplikacji, które nie wymagają danych aktualnych, plik nadmiarowy może być ignorowany.



## Plik uporządkowany - modyfikowanie rekordu

- Znalezienie i odczyt modyfikowanego rekordu do bufora
  - połowienie binarne dla warunku na polu porządkującym
  - przeszukanie całego pliku dla innego warunku
- Modyfikowanie wartości atrybutu nieporządkującego
  - zmodyfikowanie w buforze i zapis na dysk w to samo miejsce
- Modyfikowanie wartości atrybut porządkującego
  - zmiana pozycji rekordu w pliku
  - usunięcie rekordu + wstawienie nowego

Modyfikowanie rekordu wymaga jego znalezienia w pliku i odczytania do bufora. Jeżeli modyfikowany rekord spełnia warunek nałożony na pole porządkujące, wówczas wyszukanie rekordu realizuje się metodą połowienia binarnego. Jeśli natomiast wyspecyfikowano warunek nałożony na inne pole, wówczas wyszukanie rekordu wymaga w najgorszym przypadku odczytania całego pliku.

Jeżeli w znalezionym rekordzie jest modyfikowana wartość pola nieporządkującego, wówczas jest on modyfikowany w buforze i zapisywany na dysk w to samo miejsce. Jeśli natomiast modyfikacji ulega wartość pola porządkującego, wówczas rekord zmieni swoją pozycję w pliku. Implementacyjnie oryginalny rekord jest usuwany i wstawiany jest nowy rekord ze zmodyfikowaną wartością atrybutu porządkującego.



## Plik uporządkowany - zalety

- Efektywny odczyt rekordów w kolejności pola porządkującego ⇒ posortowane
- Znalezienie następnego rekordu, według określonego porządku, jest bardzo proste
- Metoda połowienia binarnego do wyszukiwania rekordu z warunkiem opartym o pole porządkujące

Operacja odczytu i sortowania rekordów wg pola porządkującego jest efektywna ponieważ odczytywane rekordy mają już właściwy porządek wynikający z organizacji pliku.

Znalezienie następnego rekordu, według określonego porządku, jest bardzo proste i wymaga odczytania następnego rekordu. Rekord ten znajduje się albo w tym samym bloku albo w bloku następnym.

Jeżeli warunek wyszukiwania rekordu jest oparty na polu porządkującym, wówczas w celu wyszukiwania danych można zastosować szybką metodę wyszukiwania binarnego (połowienia binarnego).





## Plik uporządkowany - wady

- Uporządkowanie pliku jest nieprzydatne, gdy wyszukiwanie jest realizowane według wartości pola nie porządkującego pliku danych,
- Kosztowne wstawianie i usuwanie rekordów ze względu na konieczność zachowania porządku w pliku

Uporządkowanie pliku jest nieprzydatne, gdy wyszukiwanie jest realizowane według wartości pola nie porządkującego pliku danych. W takim przypadku trzeba w najgorszym razie odczytać cały plik.

Wstawianie i usuwanie rekordów jest kosztowne ze względu na konieczność zachowania porządku w pliku.



## Plik haszowy

- Plik o strukturze wykorzystującej technikę haszowania ⇒ pliku haszowy (bezpośredni)
- Porządek rekordów w pliku określony na podstawie tzw. pola haszowego
- Koncepcja
  - zdefiniowanie funkcji haszowej (ang. hash function) z argumentem, którym jest wartość pola haszowego
  - wynikiem funkcji haszowej jest adres bloku dyskowego, w którym powinien znaleźć się dany rekord
- Haszowanie
  - wewnętrzne
  - zewnętrzne

Plik o strukturze wykorzystującej technikę haszowania nosi nazwę pliku haszowego (ang. hash file) lub bezpośredniego.

Podstawą określającą porządek rekordów w pliku jest jedno z pól rekordu zwane polem haszowym (ang. hash field).

Koncepcja organizacji tego pliku polega na zdefiniowaniu funkcji haszowej (ang. hash function), która dla argumentu równego wartości pola haszowego wyznacza pewien wynik. Wynik ten jest adresem bloku dyskowego, w którym powinien znaleźć się dany rekord.

W praktyce stosuje się dwie techniki haszowania, tj. haszowanie wewnętrzne i haszowanie zewnętrzne. Obie zostaną omówione w niniejszym wykładzie.



## Haszowanie wewnętrzne

Dana jest tablica rekordów o  $M$  szczelinach, których adresy odpowiadają indeksom tablicy haszowej

Indeks tablicy:  $0 \Rightarrow M-1$

	Id_prac	Nazwisko	Etat	Płaca
0	450	Nowak	kierownik	2000
1	120	Kuzio	portier	1000
2	220	Misiek	z-ca kier.	1800
M-2	100	Dziubek	dyrektor	5000
M-1	110	Gulczas	portier	800

Funkcja haszowa  $H(K=\text{wartość pola haszowego}) \rightarrow \{0, \dots, M-1\}$

Najczęściej spotykana funkcja haszowa  $H(K)=K \text{ MOD } M$

Koncepcja haszowania wewnętrznego jest następująca.

Przyjmijmy, że dana jest tablica rekordów o  $M$  szczelinach. Adresy szczelin odpowiadają indeksom tablicy haszowej. Indeksy te przyjmują wartości od 0 do  $M-1$ .

Przyjmijmy funkcję haszową postaci:  $H(K=\text{wartość pola haszowego}) \rightarrow \{0, \dots, M-1\}$ .

Funkcja ta transformuje wartość pola haszowego do liczby całkowitej z zakresu od 0 do  $M-1$ .

Najczęściej spotykaną funkcją haszową jest funkcja  $h(K)=K \text{ MOD } M$ .

W przykładzie ze slajdu polem haszowym mógłby być Id\_prac lub nazwisko. Wynikiem działania funkcji haszowej byłyby numer odpowiedniej szczeliny, w której umieszczony zostałby konkretny rekord.



## Haszowanie wewnętrzne - przykład

Id_Prac	Nazwisko	Etat
16	Kociołek	portier
14	Misiek	kierownik
12	Pantarka	sekretarka
11	Oleks	portier
13	Bamczyk	kierowca
15	Paker	ochroniaż
10	Karczek	ochroniaż

szczelina	
0	10; Karczek; ochroniaż
1	11; Oleks; portier
2	12; Pantarka; sekretarka
3	13; Bamczyk; kierowca
4	14; Misiek; kierownik
5	15; Paker; ochroniaż
6	16; Kociołek; portier

$$H(\text{Id\_Prac}) = \text{Id\_Prac} \bmod 10$$

Jako przykład ilustrujący haszowanie wewnętrzne rozważmy rekordy pracowników z polami: Id\_Prac, Nazwisko, Etat. Przyjmijmy tablicę haszową ze szczelinami o numerach od 0 do 6. Zdefiniujmy funkcję haszową postaci:  $H(\text{Id\_Prac}) = \text{Id\_Prac} \bmod 10$ . Funkcja ta oblicza modulo 10 z wartości atrybutu Id\_Prac. Wynikiem działania funkcji są numery szczelin w tablicy haszowej. W szczelinach tych są następnie umieszczane rekordy.



## Proces haszowania

- Proces haszowania składa się z dwóch kroków:
  - transformacji
    - pola nienumeryczne są transformowane do liczb całkowitych
  - haszowania
- Przykładowy algorytm transformacji

```
temp := 1;  
for i = 1 to 20 do  
    temp := temp * code(K[i])  
hash_address := temp MOD M;
```

Argumentem funkcji modulo musi być wartość całkowita. Oznacza to, że przed zastosowaniem haszowania należy przetransformować wartości nienumeryczne do numerycznych. Na slajdzie przedstawiono prosty algorytm, który transformuje 20 znaków zapisanych w tablicy K do wartości numerycznych i tak przetransformowane wartości transformuje za pomocą funkcji MOD M.



## Kolizja

- Kolizja  $\Rightarrow$  wartość funkcji haszowej dla danej wartości pola haszowego nowego rekordu odpowiada zajętemu już adresowi szczeliny
- Źródło kolizji  $\Rightarrow$  funkcja haszowa generująca te same adresy szczeliny dla różnych wartości atrybutu haszowego
- Rozwiązanie kolizji  $\Rightarrow$  procedura znajdowania innej lokalizacji dla wprowadzanego rekordu

Zasadniczym problemem w przypadku haszowej organizacji pliku jest tzw. problem kolizji. Kolizja występuje wtedy, gdy wartość funkcji haszowej dla danej wartości pola haszowego nowego rekordu odpowiada adresowi szczeliny, w której znajduje się już inny rekord. Kolizja jest spowodowana tym, że żadna funkcja haszowa nie gwarantuje, że różnym wartościom pola haszowego będą odpowiadały różne adresy szczelin.

Rozwiązaniem problemu kolizji jest zastosowanie procedury znajdowania innej lokalizacji dla wprowadzanego rekordu.



## Metody rozwiązywania kolizji

- Adresowanie otwarte (open addressing)
  - następna wolna lokalizacja
- Łańcuchowanie (chaining)
  - wskaźniki do nowych lokalizacji
- Haszowanie wielokrotne (multiple hashing)
  - nowa funkcja haszowa
- Każda metoda rozwiązywania kolizji wymaga własnych algorytmów wstawiania, wyszukiwania i usuwania rekordów

Wyróżnia się trzy podstawowe metody rozwiązywania kolizji, mianowicie: adresowanie otwarte (ang. open addressing), łańcuchowanie (ang. chaining) i haszowanie wielokrotne (ang. multiple hashing). Każda z tych metod wymaga własnych algorytmów wstawiania, wyszukiwania i usuwania rekordów.



## Adresowanie otwarte

```
i := adres_szczeliny --pierwotny adres szczeliny
a := i;
if lokalizacja i zajeta then
  begin
    i := (i + 1) mod M;
    while (i ≠ a) and lokalizacja i zajeta
      do
        i := (i + 1) mod M;
        if (i = a)
          then wszystkie pozycje sa zajete;
          else nowy_adres := i;
  end;
```

Adresowanie otwarte polega na znalezieniu następnej najbliższej wolnej szczeliny. Pseudo-kod algorytmu adresowania otwartego przedstawiono na slajdzie.





## Łącuchowanie

szczelina		wskaźnik do obszaru przepełnienia
0		NULL
1	rekord 1	M
2		NULL
3		M+2
M-2		
M-1		
M	rekord 2 - kolizja	M+1
M+1	rekord 3 - kolizja	NULL
M+2		M+4
M+3		
M+4		
...	.....	
M+O-2		NULL
M+O-1		NULL

obszar adresowy

obszar przepełnienia

BD – wykład 5 (33)

Technika łańcuchowania polega na przechowywaniu w szczelinie dodatkowo wskaźnika do tzw. obszaru przepełnienia (ang. overflow space). Służy on do przechowywania wszystkich rekordów ulegających kolizji w danej szczelinie. Rekordy w obszarze przepełnienia tworzą listę.

Rozważmy rysunek ze slajdu. Przyjmijmy, że pierwszy rekord ("rekord 1") jest umieszczany w szczelinie o adresie 1, zgodnie z pewną funkcją haszową. Kolejny rekord ulega kolizji w szczelinie 1 i jest umieszczany w obszarze przepełnienia o adresie M ("rekord 2 - kolizja"). We wskaźniku do obszaru przepełnienia szczeliny 1 jest umieszczany adres szczeliny przechowującej pierwszy rekord z kolizji, czyli adres M. Przyjmijmy dalej, że kolejny rekord ulega kolizji w szczelinie 1 i jest on umieszczany w kolejnej wolnej szczelinie obszaru przepełnienia. W naszym przypadku jest to "rekord 3 - kolizja" umieszczony w szczelinie M+1. Adres tej szczeliny jest zapisywany w polu wskaźnika do obszaru przepełnienia szczeliny M.

NULL w polu wskaźnika do obszaru przepełnienia oznacza brak wskaźnika.



## Haszowanie wielokrotne

- Jeżeli występuje kolizja z wykorzystaniem funkcji haszowej H1, to stosowana jest druga funkcja haszowa H2
- Jeśli H2 również powoduje kolizję to stosuje się trzecią funkcję haszową lub haszowanie otwarte

Ostatnią omawianą techniką rozwiązywania kolizji jest haszowanie wielokrotne. Jeżeli występuje kolizja z wykorzystaniem funkcji haszowej H1, to w haszowaniu wielokrotnym stosowana jest druga/inna funkcja haszowa H2, której rezultatem jest inny adres. Jeśli funkcja H2 również powoduje kolizję to stosuje się trzecią funkcję haszową lub haszowanie otwarte.



## Haszowanie zewnętrzne (1)

- Wartościami tablicy haszowej są adresy logicznych obszarów dyskowych (LOD) (block buckets)
- Liczba LOD jest stała i równa liczbie szczelin w tablicy haszowej ⇒ jest to tzw. **haszowanie statyczne** (static hashing)
- LOD jest albo pojedynczym blokiem dyskowym albo zbiorem kolejnych (leżących obok siebie) bloków dyskowych
- Funkcja haszowa odwzorowuje wartość atrybutu haszowego w numer LOD
- Plik dyskowy zawiera tablicę konwersji numerów LOD w fizyczne adresy bloków dyskowych

W haszowaniu zewnętrznym wartościami tablicy haszowej są adresy logicznych obszarów dyskowych (LOD) (ang. block buckets). Liczba LOD jest stała i jest równa liczbie szczelin w tablicy haszowej. Ponieważ liczba LOD jest stała, więc technika ta nazywa się haszowaniem statycznym.

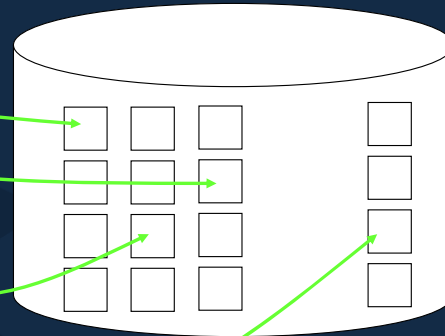
Każdy z LOD jest albo pojedynczym blokiem dyskowym albo zbiorem kolejnych (leżących obok siebie) bloków dyskowych. Funkcja haszowa odwzorowuje wartość atrybutu haszowego w numer LOD. Plik dyskowy zawiera tablicę konwersji numerów LOD w fizyczne adresy bloków dyskowych.



## Haszowanie zewnętrzne (2)

tablica konwersji

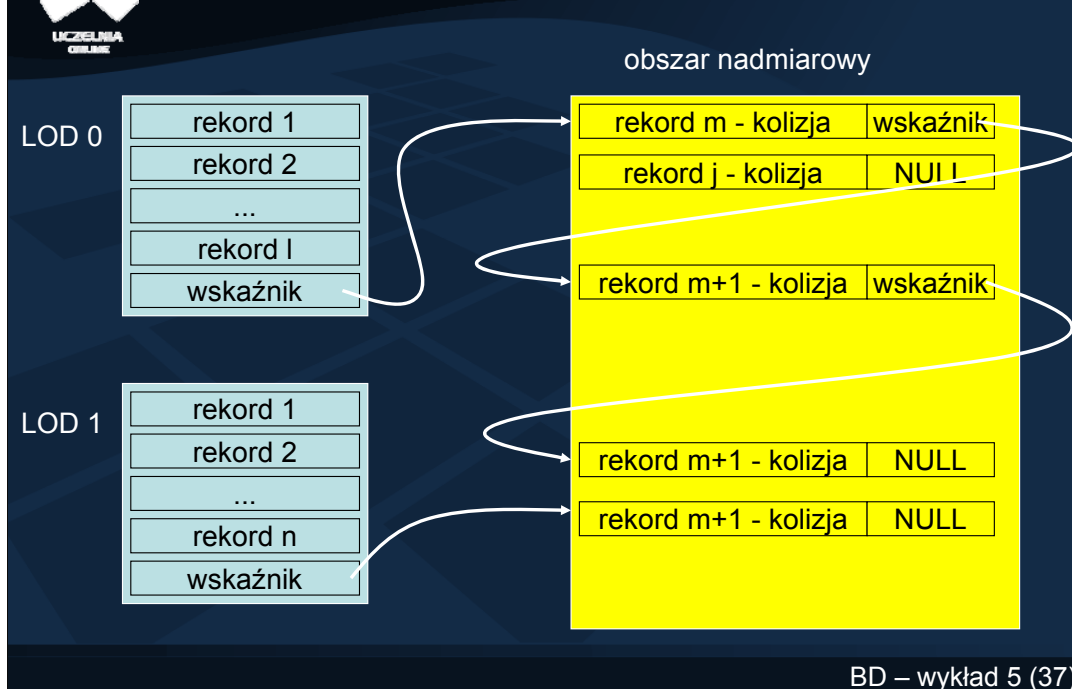
numer LOD	adres bloku na dysku
0	
1	
2	
M-2	
M-1	



Omówioną koncepcję haszowania zewnętrznego ilustruje slajd.



## Haszowanie zewnętrzne - kolizja



W haszowaniu zewnętrznym kolizje występują rzadziej niż w omawianej wcześniej technice haszowania wewnętrznego. Dzieje się tak dla tego, że ten sam LOD, którego numer jest wynikiem działania funkcji haszowej, może pomieścić wiele rekordów. Kolizja może jednak wystąpić po wypełnieniu się wszystkich bloków dyskowych wchodzących w skład danego LOD. W takiej sytuacji, alokowany jest obszar nadmiarowy. LOD zawiera wskaźnik do pierwszego rekordu tego obszaru. Kolejny rekord ulegający kolizji w tym samym LOD będzie zapisany w obszarze nadmiarowym, a wskaźnik do tego rekordu znajdzie się w poprzednim rekordzie obszaru nadmiarowego. Koncepcję tę ilustruje slajd.

LOD0 został w całości wypełniony rekordami 1 do l. Kolejny rekord m powinien trafić do LOD0. Z braku miejsca jest on umieszczany w obszarze nadmiarowym. W LOD0 jest umieszczany wskaźnik do tego rekordu. Następnie rekord m+1 również powinien trafić do LOD0 i jest umieszczany w obszarze nadmiarowym. Do rekordu m+1 prowadzi wskaźnik z rekordu m.



## Haszowanie zewnętrzne - operacje (1)

- Poszukiwanie rekordu z warunkiem nałożonym na pole inne niż haszowe
  - przeszukanie całego pliku i obszaru nadmiarowego
- Poszukiwanie rekordu z warunkiem nałożonym na pole haszowe
  - funkcja haszowa
  - w przypadku kolizji przeszukanie obszaru nadmiarowego

Poszukiwanie w pliku haszowym rekordu z warunkiem nałożonym na pole inne niż haszowe wymaga przeszukanie całego pliku i obszaru nadmiarowego. Natomiast poszukiwanie rekordu z warunkiem nałożonym na pole haszowe jest realizowane z wykorzystaniem funkcji haszowej, która generuje adres szczeliny z poszukiwanym rekordem. W przypadku kolizji, wymagane jest dodatkowo przeszukanie obszaru nadmiarowego z wykorzystaniem listy łączonej wskaźników do rekordów w tym obszarze.



## Haszowanie zewnętrzne - operacje (2)

- Usunięcie rekordu z LOD
  - wyszukanie rekordu (funkcja haszowa + tablica konwersji) i zwolnienie szczeliny
  - przesunięcie pierwszego rekordu z obszaru nadmiarowego do LOD
- Usunięcie rekordu z obszaru nadmiarowego
  - wyszukanie rekordu (funkcja haszowa) + przeszukiwanie listy rekordów w obszarze nadmiarowym
  - zwolnienie szczeliny
  - utrzymywanie listy wolnych szczelin

Usunięcie rekordu z pliku haszowego przebiega w dwóch wariantach. W wariacie pierwszym, usuwany rekord znajduje się w logicznym obszarze dyskowym. Jego usunięcie polega na znalezieniu szczeliny (z wykorzystaniem funkcji haszowej i tablicy konwersji) i jej zwolnieniu. Dodatkowo, pierwszy rekord z obszaru nadmiarowego może zostać przesunięty do zwolnionej szczeliny.

W wariacie drugim, usuwany rekord znajduje się w obszarze nadmiarowym. Jego usunięcie polega na znalezieniu szczeliny (z wykorzystaniem funkcji haszowej i tablicy konwersji) w LOD. Następnie za pomocą wskaźnika do pierwszego rekordu w obszarze nadmiarowym przeszukuje się listę rekordów w tym obszarze. Znaleziony rekord jest usuwany, a jego szczelina zwalniana. Dokonywana jest też zmiana wartości wskaźnika w rekordzie, który adresował usunięty rekord. Nowa wartość wskazuje na następny rekord wskazywany z rekordu usuniętego. W celu zoptymalizowania alokowania rekordów w obszarze nadmiarowym jest utrzymywana lista wolnych szczelin.



## Haszowanie zewnętrzne - operacje (3)

- Wstawienie rekordu
  - odczyt adresu szczeliny (funkcja haszowa)
  - w przypadku kolizji zaalokowanie szczeliny w obszarze nadmiarowym
- Zmodyfikowanie wartości pola haszowego
  - odczytanie rekordu (funkcja haszowa)
  - rekord zmienia szczelinę
    - usunięcie rekordu + wstawienie rekordu z nową wartością
- Zmodyfikowanie wartości pola nie-haszowego
  - odczytanie rekordu (funkcja haszowa)
  - zmodyfikowanie wartości
  - rekord nie zmienia szczeliny

Wstawienie rekordu do pliku haszowego polega na odczytaniu adresu szczeliny z wykorzystaniem funkcji haszowej i zapisaniu rekordu do tej szczeliny. W przypadku kolizji, alokowana jest szczelina w obszarze nadmiarowym i uaktualniany jest wskaźnik prowadzący do tej szczeliny.

Zmodyfikowanie wartości pola haszowego polega na odczytaniu rekordu z wykorzystaniem funkcji haszowej. Ponieważ zmiana wartości pola haszowego wymaga przeniesienia rekordu do innej szczeliny, fizycznie rekord stary jest usuwany i wstawiany jest nowy rekord do właściwej szczeliny.

Zmodyfikowanie wartości pola nie-haszowego polega na odczytaniu rekordu, zmodyfikowaniu wartości pola i zapisaniu rekordu do tej samej szczeliny.





## Funkcja haszowa

- Cechą dobrej funkcji haszowej jest zapewne równomiernego rozkładu rekordów w obrębie przestrzeni adresowej tablicy haszowej
- Zalecany rozmiar tablicy haszowej

$$r/M \in (0,7 \div 0,9)$$

$r$   $\Rightarrow$  liczba rekordów

$M$   $\Rightarrow$  liczba bloków

Dobra funkcja haszowa to taka, która odwzorowuje wartości kluczy rozłożone nierównomiernie w obrębie dużej dziedziny, w przestrzeń adresową rekordów o znacznie mniejszym rozmiarze w taki sposób, że wynikowe adresy są równomiernie rozłożone w obrębie przestrzeni adresowej tablicy haszowej.

Praktyka pokazuje, że tablica haszowa powinna być zajęta w 70 do 90%, tak aby zapewnić niewielką liczbę kolizji i nie powodować zbyt dużego marnowania obszaru dyskowego. Stąd zalecany rozmiar tablicy haszowej jest wyrażony wzorem  $r/M$  between 0.7 and 0.9, gdzie  $r$  jest liczbą rekordów, a  $M$  jest liczbą szczelin adresowych tablicy haszowej.



## Charakterystyka plików haszowych

- Problem porządkowania pliku oraz wyszukiwania rekordów w porządku wartości pola haszowego
- Problem stałego rozmiaru przestrzeni adresowej przydzielonej plikowi
  - częste kolizje

Pliki haszowe są nieefektywne dla operacji odczytu danych z ich porządkiem zgodnie z wartością pola haszowego. Jest to konsekwencją działania funkcji haszowej, która burzy porządek wartości pola haszowego rozmieszczając rekordy sąsiednie w różnych blokach. Ponadto, haszowanie statyczne zakłada stały rozmiar przestrzeni adresowej przydzielonej plikowi. Konsekwencją tego ograniczenia jest duże prawdopodobieństwo występowania kolizji.