Reprezentowanie danych

- W jaki sposób organizowane jest miejsce dla informacji zawartych w tabeli ?

```
CREATE TABLE GwiazdaFilmowa
    (
    nazwisko CHAR(30) PRIMARY KEY,
    adres VARCHAR(255),
    plec CHAR(1),
    data_ur DATE
    );
```

- Pola reprezentowane są przez ciągi bajtów stałej lub zmiennej długości.
- Krotki (rekordy) to zbiory pól
- Rekordy przechowywane są w blokach
- Zbiór bloków tworzy plik

Reprezentowanie danych

- Typ **CHAR(n)** - najprostszy sposób: ciąg znaków o długości n bajtów.

  Nawet jeśli łańcuch znaków wypełniających jest krótszy, to pozostała część miejsca przeznaczonego dla wartości tego pola jest wypełniana jakimś znakiem specjalnym
- Typ **VARCHAR(n)** - tu (w początkowych bajtach) oprócz znaków przechowywana jest informacja o długości łańcucha
- Typ **DATE** - może być przechowywany jako napis (o określonym formacie), lub jako liczba (od ustalonego „początku”).

Rekordy

- Dla każdego rodzaju rekordu musi być zapisany jego schemat, który zawiera nazwy i typy pól, a także ich położenie względem początku rekordu.
- Schemat jest przeglądany podczas dostępu do składowych rekordu.
- W przypadku rekordów z polami o stałych długościach dane mogą następować po sobie.

- Początek kolejnego pola jest określany przesunięciem względem początku rekordu.

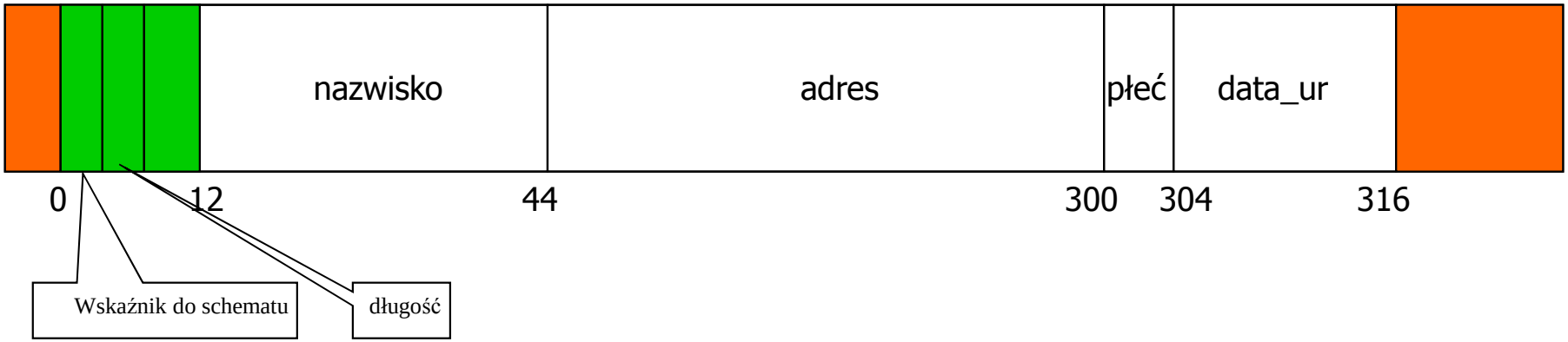
Rekordy

- W niektórych systemach umieszczanie danych pod adresami będącymi wielokrotnością 4 jest bardziej efektywne (dla pewnych typów danych jest to wręcz wymagane).
- Trzeba na to uważać podczas przepisywania do PAO.
- Dla uproszczenia przyjmujemy, że pola zaczynają się od bajtów w PAO, których adresy są wielokrotnością 4.

- Po co przechowywać schemat przesunięć w samym rekordzie?
- Struktura rekordu może się zmieniać, musi więc być dostęp do jej aktualnej postaci.

Rekordy

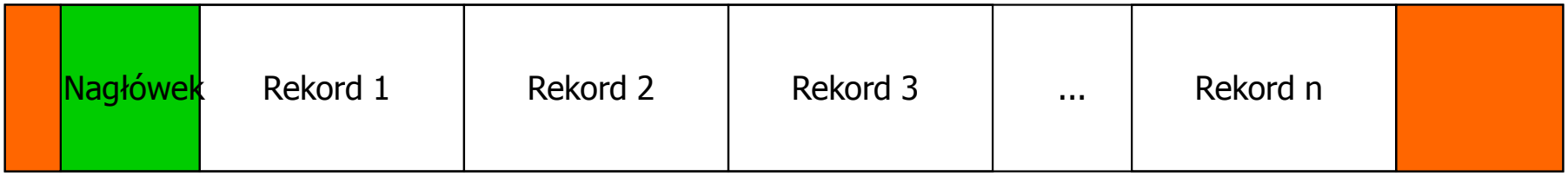
- Co przechowuje nagłówek rekordu:
  - schemat rekordu
  - długość rekordu
  - znaczniki czasowe (np. ostatnia modyfikacja, ostatni odczyt)



- Długość krotki wynika z jej struktury, ale czasem informacja jest przydatna bezpośrednio w nagłówku (np. gdy nie sięgamy do zawartości rekordu, tylko szukamy następnego

Bloki

- Blok przechowujący rekordy stałej długości



- Nagłówek jest opcjonalny. Może się w nim znajdować:
  - katalog przesunięć każdego rekordu w bloku
  - identyfikator
  - znacznik czasowy dostępu do bloku

- Przy długości rekordu 316 B w bloku 4096 B (zachowując 12 B na nagłówek) mieści się 12 rekordów (i pozostaje 292 B nieużyte)

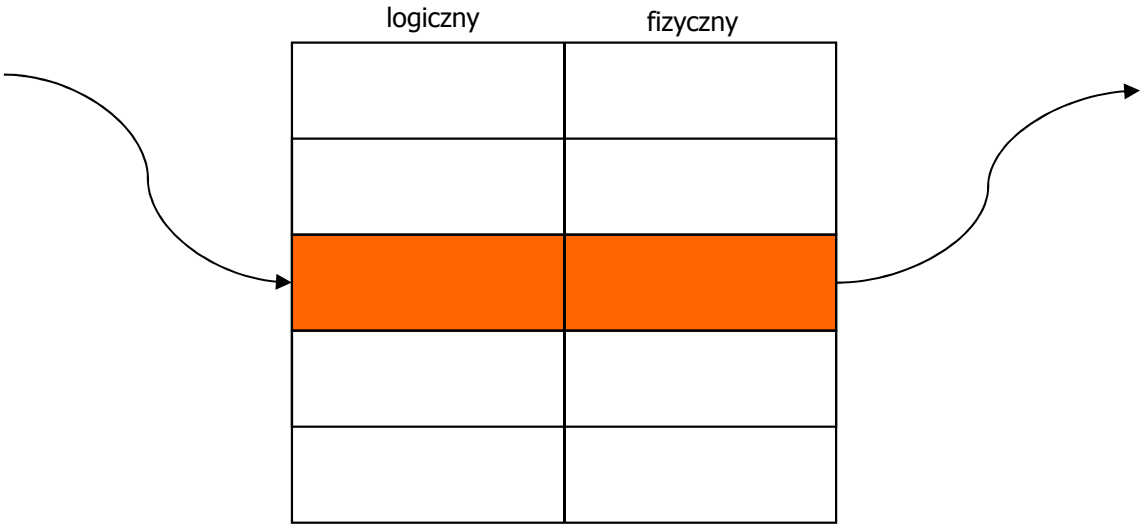
Adresy bloków i rekordów

- W modelu pracy klient-serwer oprogramowanie klienta korzysta z pamięci (dokładniej - z wirtualnej przestrzeni adresowej).
- Proces serwera dostarcza dane z pamięci pomocniczej do jednego lub wielu takich procesów klienta.
- Adres bloku ładowanego do pamięci jest ustalany jako pierwszy adres jego pierwszego bajta w pamięci wirtualnej, a adres rekordu w takim bloku jako adres pierwszego bajta tego rekordu.
- Zanim blok zostanie załadowany do pamięci konieczne jest ustalenie, gdzie taki blok się znajduje. SZBD również posiada swoją **przestrzeń adresową**, która na to pozwala.
- Przestrzeń adresowa może być zorganizowana wprost, jako
  - **Adresy Fizyczne:** ciągi bajtów, które umożliwiają określenie miejsca w pamięci pomocniczej, w którym znajduje się rekord, np.:
    - **Komputer** do którego należy pamięć pomocnicza (w sytuacji, gdy baza jest utrzymywana na kilku komputerach)
    - **ID dysku** lub urządzenia, które przechowuje blok
    - **Numer cylindra** na dysku
    - **Numer ścieżki** w cylindrze
    - **Numer bloku** na ścieżce
    - **Przesunięcie początku rekordu** w bloku ta tego rekordu.

Adresy bloków i rekordów

- Zwykle jednak przestrzeń adresowa zawiera
  - **Adresy Logiczne:** ciągi bajtów ustalonej długości

Potrzebna jest wówczas **tablica odwzorowań** (przechowywana w ustalonym miejscu), która ustala przyporządkowanie adresów logicznych i fizycznych



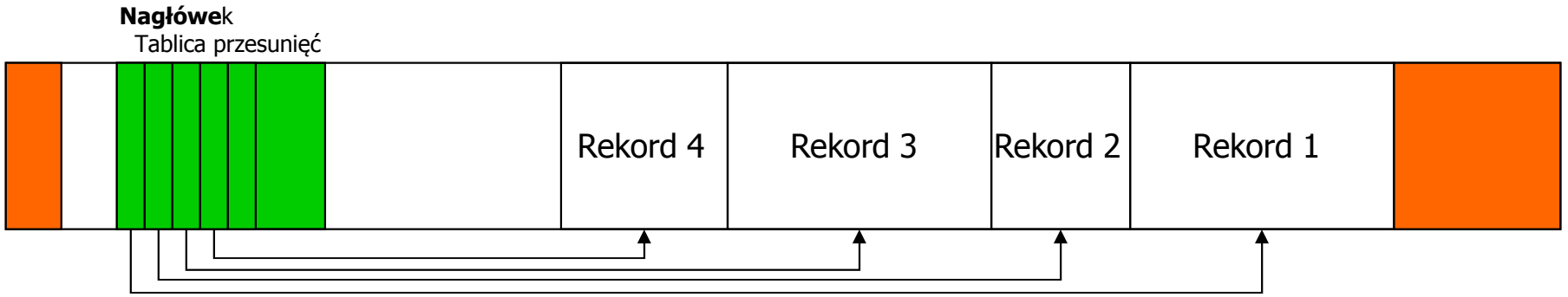
Adresy bloków i rekordów

- Po co stosować adresy logiczne? (rodzi to dodatkową konieczność przeszukiwania tablicy odwzorowań).
  - **Adresy fizyczne są długie** (8~16B potrzebne na zapisanie wszystkich elementów adresu fizycznego).
  - Przemieszczanie danych między blokami (lub w samym bloku) dzięki stosowaniu adresów logicznych jest łatwiejsze - wystarczy zmodyfikować adresy fizyczne tylko w tablicy odwzorowań - Logiczne pozostaną niezmienione.

- **Adresy strukturalne -** połączenie adresów logicznych i fizycznych.
  - Przykład: utrzymujemy adres fizyczny bloku i klucz rekordu zamiast jego przesunięcia względem początku bloku. Wtedy odnalezienie rekordu wymaga przeszukania bloku, ale ponieważ odbywa się to w PAO, to czas tej operacji jest nieistotny.

Adresy bloków i rekordów

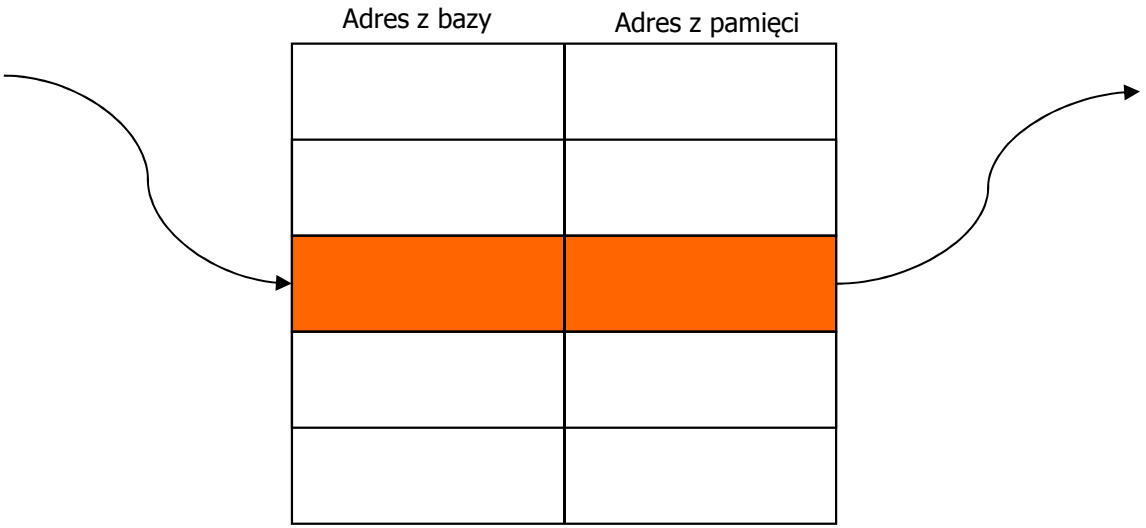
- Użyteczne połączenie adresów logicznych i fizycznych - przechowywanie z każdym blokiem **tablicy przesunięć**.
- Adres rekordu składa się wówczas z fizycznego wskazania na blok i informacji z tablicy przesunięć wewnątrz bloku



- Rekordy przyrastają od końca bloku - nie trzeba rezerwować z góry zadanej ilości bajtów na tablicę przesunięć.
- Przemieszczenie rekordu wewnątrz bloku wymaga jedynie zmiany wartości w tablicy przesunięć
- Usuwanie polega na wstawieniu w tablicy przesunięć ustalonego znacznika (tzw.***klepsydry***)

Adresy z bazy i z pamięci

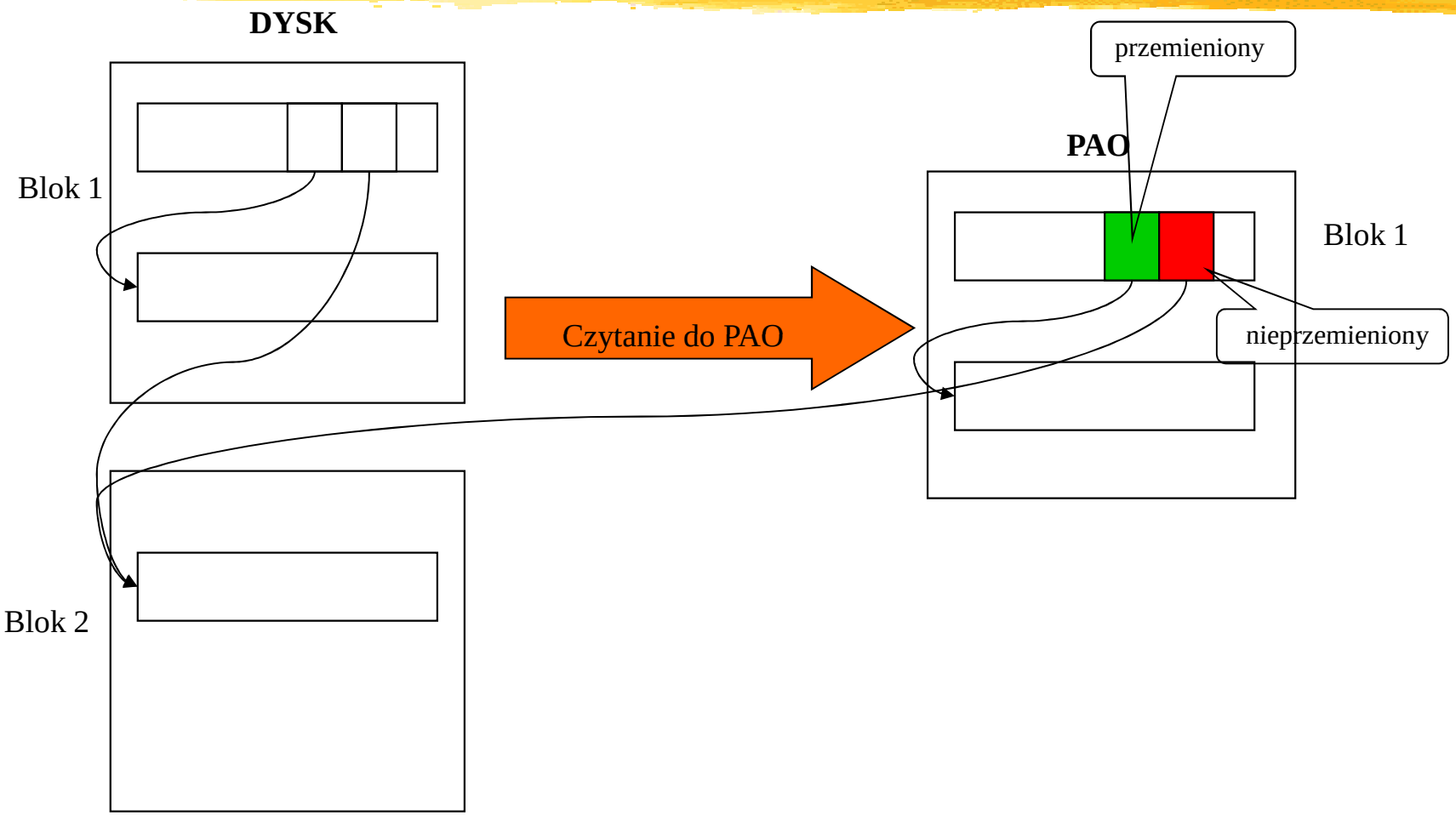
- Dotychczasowe adresy (fizyczne, logiczne, strukturalne) były to tzw.***adresy z bazy,***
- Po przeczytaniu bloku do pamięci operacyjnej dostaje on oprócz posiadanego adresu z bazy również **adres z pamięci** wirtualnej.
- Odwoływanie się do adresów z bazy w tej sytuacji staje się niewygodne (skoro można używać zwykłych adresów PAO)
- Potrzebna kolejna tablica odwzorowań adresów z bazy na adresy z pamięci.



Przemiana wskaźników

- Adresy z pamięci odnoszą się do kopii bloków w PAO.
- W tablicy translacji nie ma zapisów dotyczących danych, które były, ale obecnie nie istnieją w PAO. Znajdują się tu wyłącznie adresy obiektów istniejących aktualnie w PAO.
- Podczas przemieszczania danych z pamięci pomocniczej do PAO wykonywana jest ***przemiana wskaźników*** (czyli tłumaczone są adresy z bazy na adresy z pamięci wirtualnej).
- Wskaźnik posiada więc oprócz właściwego adresu również dodatkowy bit wskazujący z jakim adresem mamy do czynienia.

Przemiana wskaźników



Przemiana wskaźników

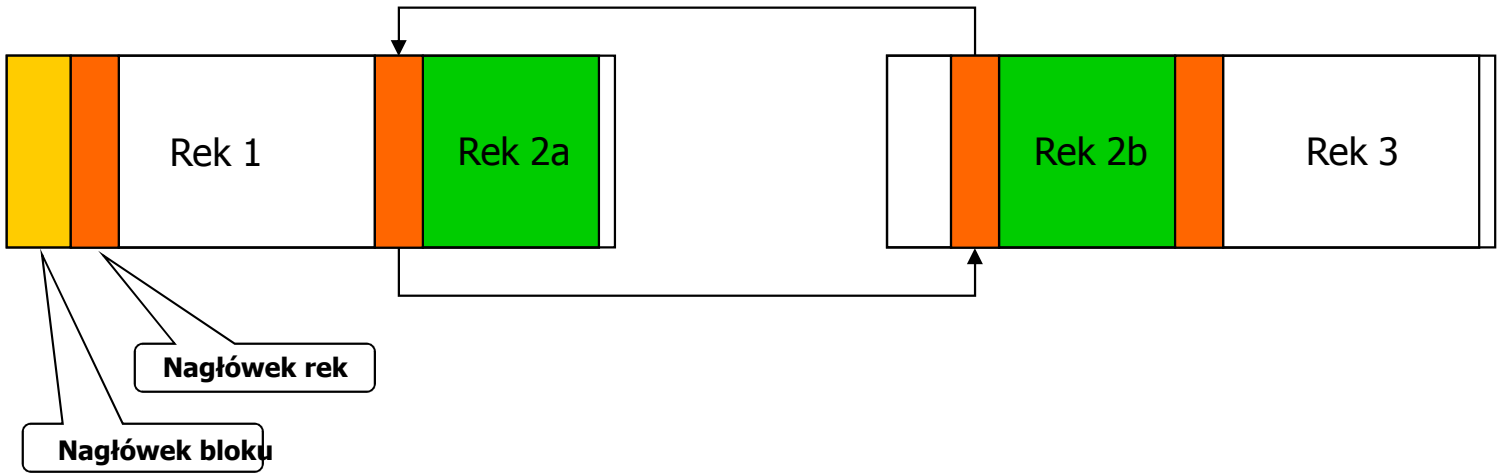
- Kiedy przemieniać wskaźniki? (przemiana automatyczna vs przemiana „na żądanie”)
  - przy przemianie automatycznej po zaczytaniu do PAO **wszystkie** adresy ulegają zamianie.
  - Przy zwracaniu na dysk bloku z PAO konieczne jest zamienienie ich ponownie na adresy z bazy (czyli trzeba przeszukać tablicę translacji w przeciwnym kierunku)
  - mogą się pojawić **bloki unieruchomione** czyli takie, których nie można bezpiecznie zapisać na dysk. W poprzednim przykładzie blok 2, gdyby został zaczytany do PAO stałby się blokiem unieruchomionym, ponieważ blok 1 (który zawierałby wówczas przemienione odwołania do bloku 2) utraciłby adresy wskazujące na blok 2 w momencie zapisania na dysku i usunięcia z PAO. Blok 2 pozostaje unieruchomiony do czasu, gdy wszystkie adresy wskazujące na niego nie zostaną przemienione.

Dane i rekordy zmiennej długości

- Pola w rekordzie dopuszczają zmienną długość
- Pola są zbyt długie, by pomieścić się w bloku (Binary Large Objects) (np. JPEG, MPEG)
- Pole nie mieści się w bloku, bo inne pola zajmują większość miejsca.

- W takich sytuacjach stosuje się **„rozpinanie rekordów”**

- Przykład: Rekord ma długość nieco większą od połowy długości bloku.

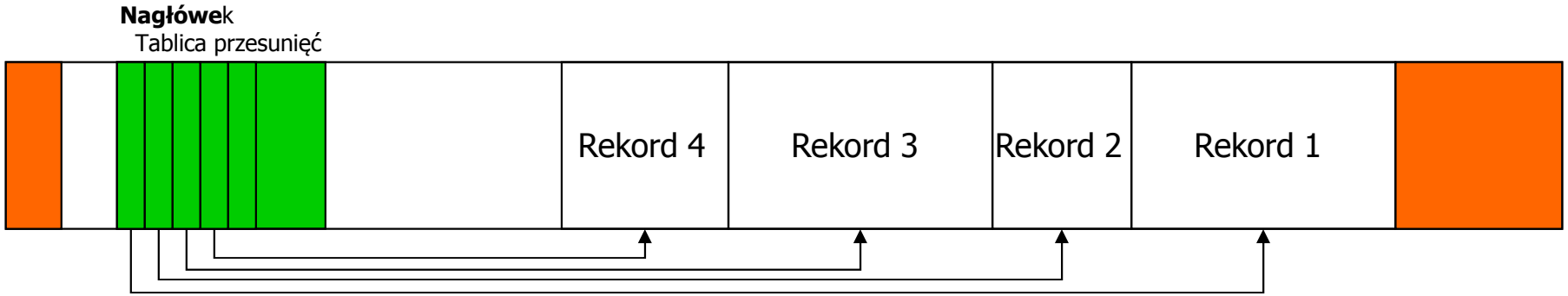


Dane i rekordy zmiennej długości

- Każdy nagłówek rekordu, lub fragmentu rekordu musi zawierać bit informacji o tym, czy jest fragmentem, czy całym rekordem
- fragmenty rekordów zawierają dodatkowe bity (czy pierwszy, czy ostatni)
- jeśli istnieje fragment poprzedni/następny, to potrzebne są do nich wskaźniki

Zmiany w bazie

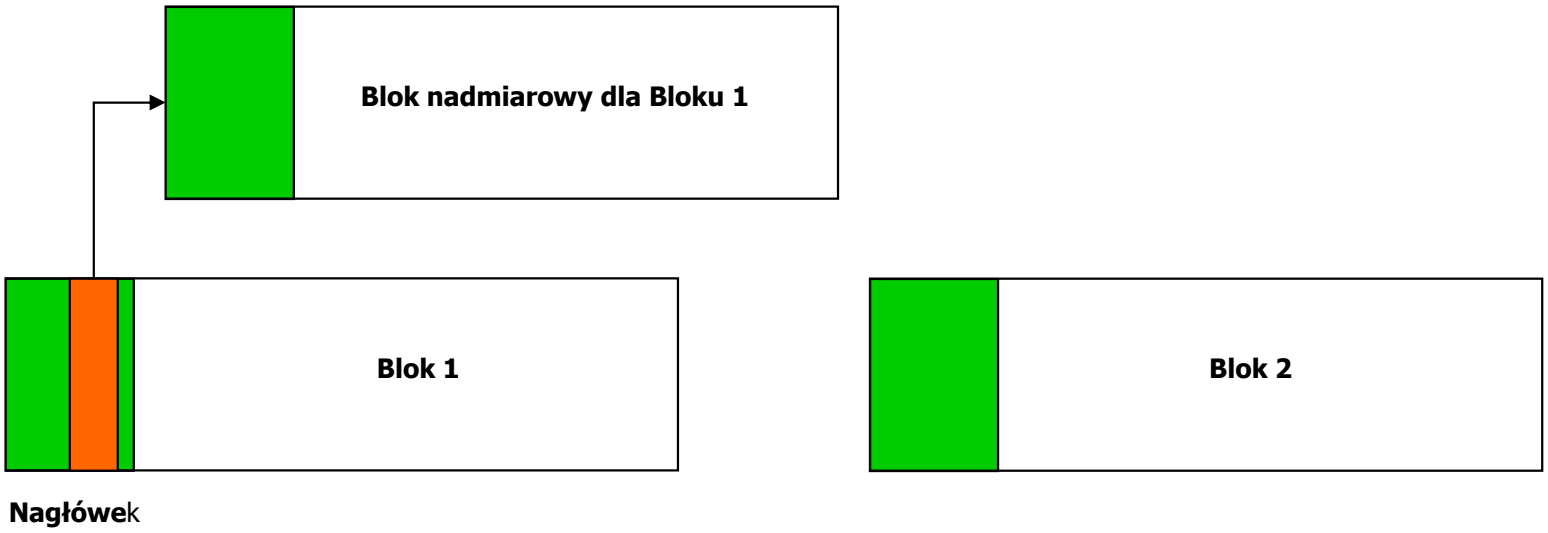
- Podczas **wstawiania** rekordu do tabeli:
  - Jeśli nie ma potrzeby przechowywania danych w określonym porządku, to wystarczy znaleźć blok z wolną przestrzenią i umieścić tam rekord.
  - Jeśli konieczne jest utrzymanie kolejności rekordów, wówczas powstaje problem umieszczenia danych pomiędzy już istniejącymi.



- jeśli wstawianie między istniejące dane odbywa się w ramach jednego bloku, to dzięki użyciu tablicy przesunięć wystarczy jedynie przeorganizować dane i zaktualizować zapisy w tablicy przesunięć.

Zmiany w bazie

- Jeśli jednak nowe dane nie mieszczą się w bloku z istniejącymi danymi, to:
  - albo znajdowane jest miejsce w sąsiednim bloku i rekordy są przesuwane (z uwzględnieniem zmian w adresach).
  - albo tworzony jest tzw. **blok nadmiarowy.** W nagłówku bloku jest umieszczane wskazanie na blok nadmiarowy



Zmiany w bazie

- Podczas **usuwania** rekordu z tabeli:
  - Można skonsolidować miejsce w bloku, a także usuwać bloki nadmiarowe, jeśli zostały utworzone.
  - Do takich bloków mogły się jednak pojawić odwołania w innych blokach, więc istotne jest zadbanie o to, by nie stały się one „pustymi” wskazaniami.
  - We wszystkich wskazaniach na usuwany rekord stosuje się klepsydrę (np. w tablicy odwzorowań adresów fizycznych na logiczne, lub w tablicy przesunięć wewnątrz bloku
- Podczas **aktualizacji** rekordu z tabeli:
  - zmiana wielkości rekordu może wywołać efekty podobne jak przy wstawianiu/usuwaniu rekordu. Może więc pojawić się konieczność dodania bloku nadmiarowego, lub skonsolidowania miejsca odzyskanego po zmianie.

Relacje i indeksy

- Jak są reprezentowane całe relacje w bazie?
  - Czy wystarczy podzielić rekordy między kolejne bloki?

    SELECT * FROM TABELA;

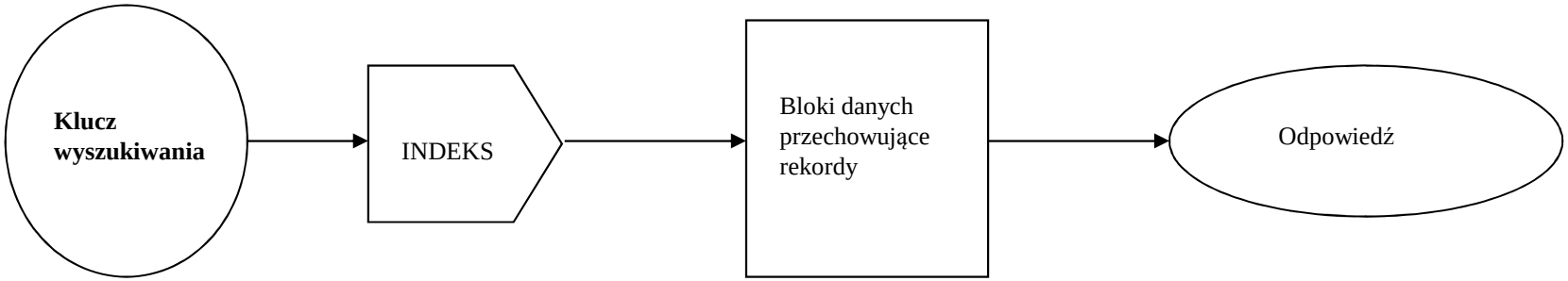
  - Aby wykonać takie zapytanie konieczne byłoby przejrzenie wszystkich bloków i dodatkowa informacja, że dane z bloku należą do danej relacji.

  - Nawet gdyby na relację KONTRAHENCI były zarezerwowane określone bloki to w przypadku poniższego zapytania i tak konieczne by było ponownie przejrzenie każdego z nich.

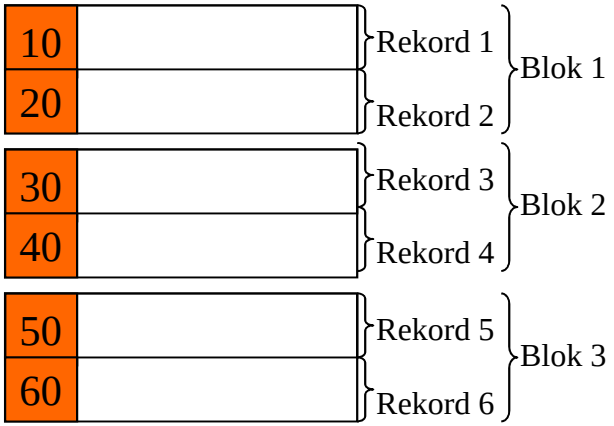
    SELECT * FROM KONTRAHENCI WHERE nazwisko = 'Kowalski';

- **Indeks** to struktura związana z relacją, pozwalająca na podstawie pewnej cechy rekordu (wartości jednego lub kilku pól) dotrzeć do tego/tych rekordu/ów w możliwie szybki sposób.

Relacje i indeksy



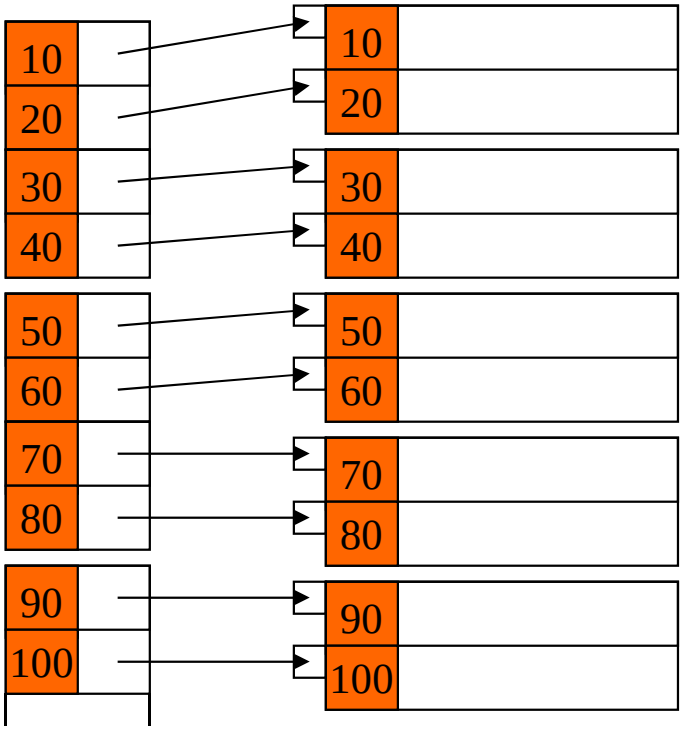
- Przypadek 1:
  - Plik z danymi jest posortowany wg klucza wyszukiwania (***plik sekwencyjny***).
  - Dla uproszczenia zakładamy, że klucz dotyczy pola numerycznego o wartościach całkowitych (liczby co 10)
  - utworzony indeks będzie **Indeksem gęstym** - przechowywane są w nim klucze każdego rekordu i wskaźniki do tych rekordów



Indeks gęsty

- po co tworzyć indeks, skoro plik już jest posortowany?

  Gdy indeks w całości mieści się PAO, wówczas dostęp do szukanego rekordu wymaga tylko jednego dostępu do dysku.
- W przykładzie blok z danymi zawiera tylko 2 rekordy.
- Blok z danymi indeksu zawiera po 4 pary „klucz-wskaźnik” (zwykle takich par w bloku jest znacznie więcej (setki)



- Nawet w tak uproszczonym przypadku występują oszczędności w odczytach (bloków indeksu jest o połowę mniej niż bloków z danymi).
- Plik indeksu również jest sekwencyjny, więc posortowany wg klucza wyszukiwania. Można więc do wyszukiwania zastosować np. wyszukiwanie binarne, przeglądając log(n) bloków przy n wszystkich bloków indeksu.
- W praktyce jest duża szansa, że cały indeks zmieści się w PAO, więc proces ustalania adresu rekordu nie będzie wymagał dostępów do dysku.

Indeks gęsty

- Przykład (realny):
  - Relacja zawiera 1 milion rekordów
  - w bloku mieści się 10 rekordów (blok po 4096 B)
  - Relacja zajmuje więc ok. 400MB
  - niech pole klucza będzie długości 30 B
  - niech wskaźnik do rekordu zawiera 8 B
  - W bloku mieści się 100 par „klucz-wskaźnik” (i zostanie 300 B na nagłówek bloku)

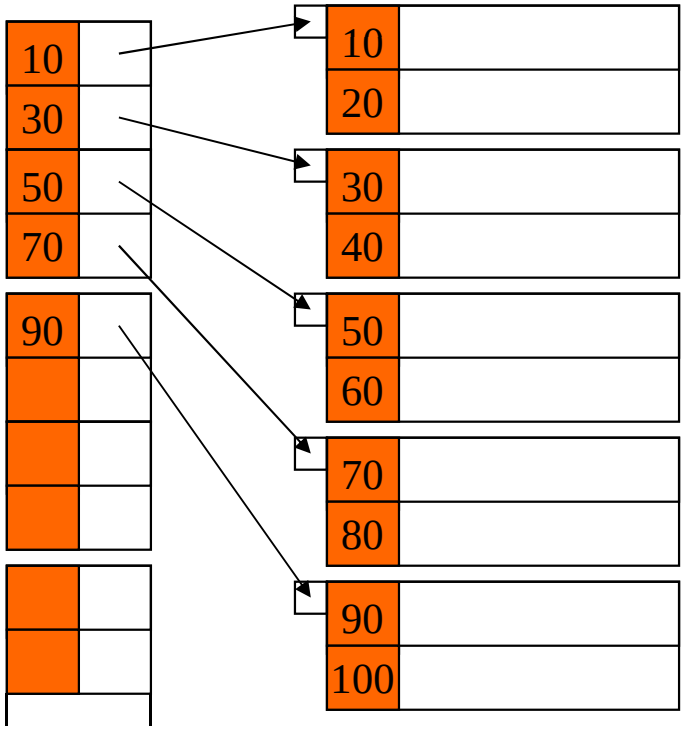
Cały indeks mieści się w 10 tyś bloków (czyli ok. 40 MB). To nadal dosyć duży indeks, ale ma już pewne szanse, by zmieścić się w PAO w całości. Nawet gdyby się nie udało, to i tak: $13 < \log(10000) < 14$

Wystarczy więc 13~14 odczytów bloków, by znaleźć ten, w którym znajduje się szukany klucz i tym samym adres do szukanego rekordu.

Można też rozważać utrzymywanie pewnej części bloków w PAO nawet, jeśli w całości indeks się w niej nie mieści.

Indeks rzadki

- Indeks można zorganizować inaczej, wykorzystując fakt, że operacja wyszukiwania w pamięci operacyjnej jest mniej kosztowna, niż odczytywanie bloków z dysku.
- **Indeks rzadki** utrzymuje tylko jedną parę „klucz-wskaźnik” z bloku z danymi.
- Tak skonstruowany indeks jest mniejszy od gęstego.
- Nadal obowiązuje założenie sekwencyjności pliku.



Indeks rzadki

- Przy założeniach poprzedniego przykładu indeks rzadki mieści się w 1000 bloków (4 MB).
- W przypadku indeksu gęstego można było od razu (bez konieczności odczytywania bloków z danymi) odpowiedzieć na zapytanie typu:
  - Czy istnieje rekord o wartości klucza K ?
- W przypadku indeksu rzadkiego nie ma możliwości stwierdzenia tego na podstawie informacji zawartych w samym indeksie.
- Zasada odnajdywania rekordu na podstawie indeksu rzadkiego:
  - Szukamy największego klucza, który jest mniejszy lub równy K i wg niego odczytujemy blok z danymi
  - w odczytanym bloku (mając go już w pamięci) odnajdujemy właściwy rekord.

- Indeksy rzadkie można „składać”
- dla indeksów gęstych taka czynność nie ma sensu (brak oszczędności).
- Zamiast składania indeksów stosuje się raczej inne struktury (B-drzewa)

Składanie indeksów