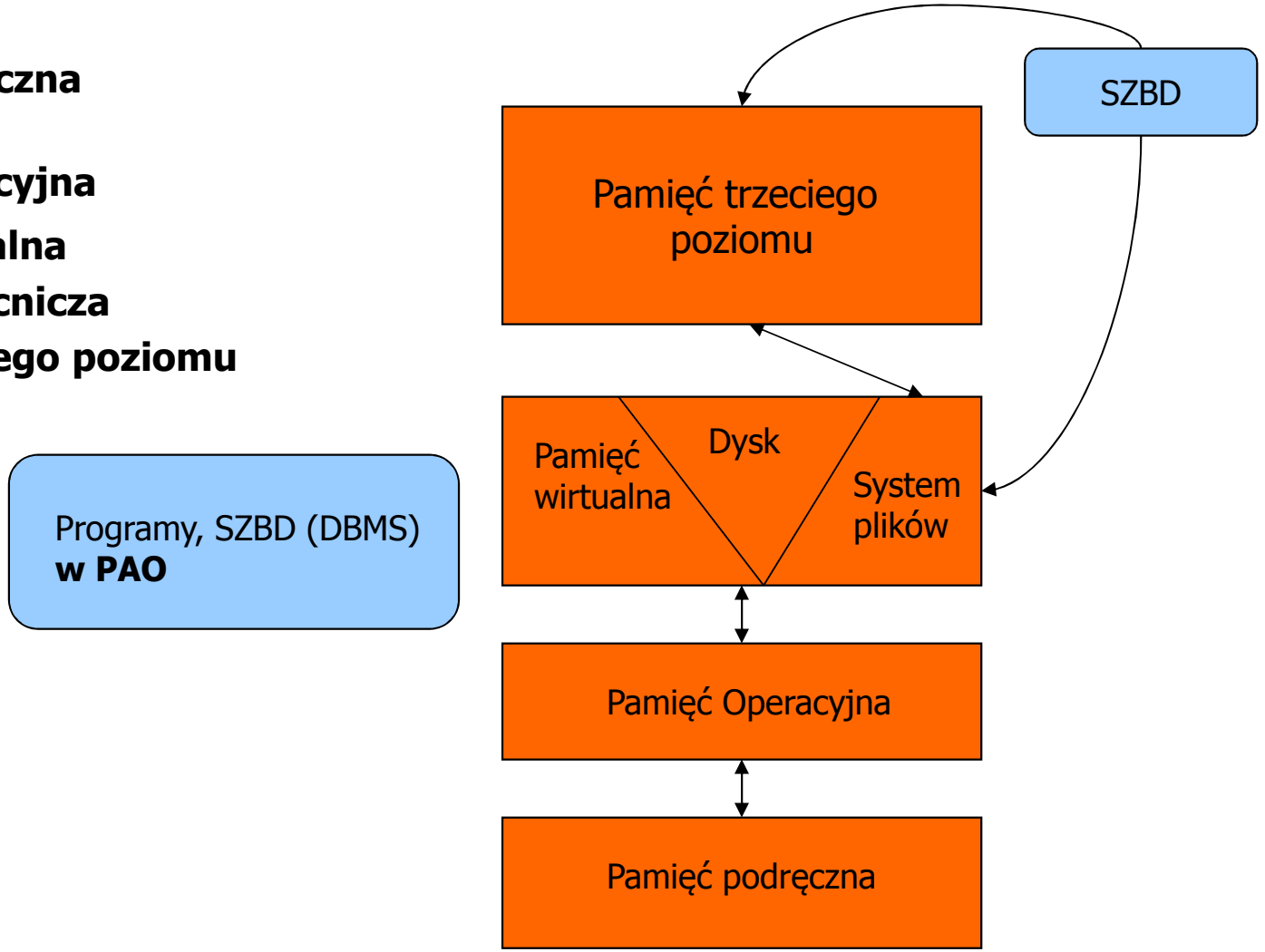
# Przechowywanie danych

- Systemy baz danych mają zdolność efektywnej obsługi bardzo dużych zbiorów danych.
- W jaki sposób System Zarządzania Bazą Danych (SZBD) pamięta i organizuje duże zbiory danych?
- Jak sposoby składowania danych wpływają na efektywność operowania na nich.

# Hierarchia pamięci

- **Pamięć podręczna**
- **Pamięć operacyjna**
- **Pamięć wirtualna**
- **Pamięć pomocnicza**
- **Pamięć trzeciego poziomu**



# Hierarchia pamięci

## Pamięć podręczna

- Zintegrowany układ scalony (lub fragment procesora) - może przechowywać dane jak i instrukcje. Jedne i drugie trafiają do pamięci podręcznej z określonego miejsca pamięci operacyjnej (PAO). Zwykle przesyłane są tu niewielkie ilości danych. Można więc w praktyce założyć, że w pamięci podręcznej są przechowywane pojedyncze instrukcje, liczby, czy krótkie łańcuchy.
- Pamięć podręczna jest czasem dzielona na *pamięć podręczną płyty* (w tej samej kostce co mikroprocesor) i *pamięć podręczną drugiego poziomu* (w odrębnej kostce).
- Zasada komunikacji z PAO: Gdy instrukcje są wykonywane, wówczas sam rozkaz i dane są pobierane do pamięci podręcznej. Jeśli nie można ich tam odnaleźć, to są kopiowane z PAO. Często (ze względu na pojemność pamięci podręcznej), dane zawarte w pamięci podręcznej muszą być zamieniane nowymi. Jeśli nie były zmieniane, to zostają po prostu zastąpione następnymi, jeśli były na nich wykonywane zmiany, to muszą wcześniej zostać skopiowane do PAO.
- W systemach wieloprocesorowych, gdzie kilka procesorów korzysta z tej samej PAO, często dane z pamięci podręcznej muszą być natychmiast odwzorowywane w pamięci operacyjnej.
- Pojemność pamięci podręcznej: (około 1MB ),
- Czas wymiany z procesorem lub wykonania rozkazu: ok. 10 nanosekund (10^-8 sekundy )
- Czas wymiany z pamięcią operacyjną: ok. 100 nanosekund (10^-7 sekundy)

# Hierarchia pamięci

## Pamięć operacyjna

- O wszelkich operacjach wykonywanych w komputerze można myśleć jak o operacjach wykonywanych w pamięci operacyjnej, gdyż wymiana z pamięcią podręczną jest tak szybka, że można ją pominąć przy rozważaniach związanych z czasem.
- Pamięć operacyjną charakteryzuje *dostęp swobodny,* tzn. każdy bajt jest dostępny w takim samym czasie.
- Pojemność pamięci operacyjnej: (100 MB – 10 GB i więcej )
- Czas dostępu do danych: 10 – 100 nanosekund.

# Hierarchia pamięci

## Pamięć wirtualna

- Program wykonywany przez komputer znajduje się w *przestrzeni adresowej pamięci wirtualnej.*
- 32 bitowa przestrzeń adresowa pozwala zaadresować 2^32 adresów (ponad 4 miliardy adresów ). Typowa pamięć wirtualna ma więc 4 GB.
- Jest to więcej niż dostępna pamięć operacyjna, zatem większość pamięci wirtualnej jest przechowywana na dysku.
- Bloki na dysku 4KB do 56 KB.
- Pamięć wirtualna jest przekazywana między dyskiem a pamięcią operacyjną w blokach nazywanych stronami.

# Hierarchia pamięci

## Pamięć pomocnicza

- Dyski (zwykle magnetyczne, optyczne i magnetooptyczne).
- Pamięć wolniejsza, ale znacznie pojemniejsza niż pamięć operacyjna.
- Czas dostępu do danych nie jest stały, ale różnice nie są duże.

  Dyski są więc podstawą dla pamięci pomocniczej, ale także dla pamięci wirtualnej.
- DBMS'y często nie korzystają z pośrednictwa menedżera plików systemu operacyjnego, ale same organizują sobie wymianę między dyskiem i pamięcią operacyjną. Zasady pozostają jednak takie same.
- Operacje zapisu i odczytu trwają w przybliżeniu 10-30 milisekund (0,01 – 0,03 sek.)
- W tym czasie przeciętna maszyna może wykonać milion rozkazów, dlatego tak istotna jest gospodarka pamięcią. W miarę możliwości jak najczęściej blok z danymi zapisany na dysku powinien znajdować się już w pamięci, gdy będzie potrzebny.

# Hierarchia pamięci

## Pamięć III-go poziomu

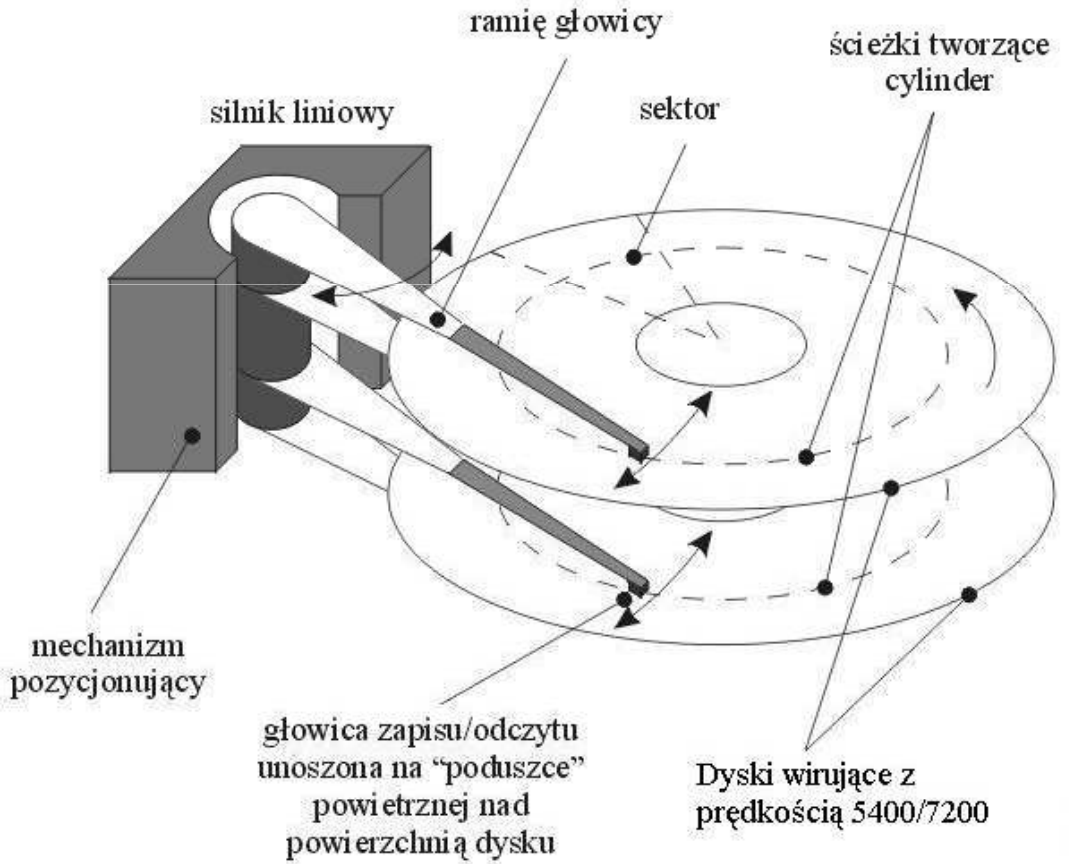
- Stosowane są pamięci taśmowe, urządzenia automatycznie zmieniające dyski optyczne (jukebox), silosy taśmowe (robot przenosi taśmy do czytników).
- Pamięć III-go poziomu charakteryzuje się znacznie dłuższymi czasami dostępu i znacznie większymi pojemnościami.
- Pojemności rzędu terabajtów (jedna taśma potrafi przechowywać 50GB)
- Czas dostępu (kilka sekund do kilku minut).

# Prawo Moore’a

- Możliwości układów scalonych są podwajane zgodnie z krzywą wykładniczą w cyklu osiemnastomiesięcznym, dotyczy to poniższych parametrów:
- Szybkość procesora (liczba instrukcji wykonywanych w ciągu sekundy)
- koszt bitu pamięci operacyjnej i liczba bitów upakowanych w jednej kostce.
- Liczba bitów możliwych do przechowywania na dysku.

- Prawo Moore’a nie stosuje się do wszystkich parametrów - szybkość dostępu do danych w PAO, prędkość obrotowa dysków należą do najwolniej zmieniających się parametrów. Pogłębia się przez to różnica między czasem dostępu do danych, a czasem obliczeń. Już w tej chwili „odległość” pamięci pomocniczej od procesora jest kłopotliwa.

# Dyski


ramię głowicy
ścieżki tworzące cylinder
silnik liniowy
sektor
mechanizm pozycjonujący
głowica zapisu/odczytu unoszona na "poduszce" powietrznej nad powierzchnią dysku
Dyski wirujące z prędkością 5400/7200

# Dyski - budowa

- Na ***powierzchni dysków*** znajdują *się* ***ścieżki***.
- Ścieżki są podzielone na sektory pooddzielane nienamagnesowanymi przerwami (przerwy - ok. 10%, sektory - ok. 90%)
- ***Sektory*** stanowią podział fizyczny, ***Bloki*** - podział logiczny. Rozmiar bloku jest co najmniej taki jak rozmiar sektora, zwykle zawiera jednak więcej sektorów.
- ***Sterownik dysku –*** niewielki procesor, pozwalający na
  - Sterowanie mechanicznym urządzeniem uruchamiającym, które przesuwa układ głowic nad odpowiednią ścieżkę. Zestaw wszystkich ścieżek znajdujących się pod głowicami w pewny ustawieniu to ***cylinder.***
  - Wybór powierzchni, z której następuje odczyt (zapis) i określenie ścieżki nad którą znajdują się głowice. Sterownik ustala także moment, w którym wrzeciono osiąga położenie, w którym żądany sektor zaczyna się przesuwać pod głowicą.
  - Przesyłanie bitów odczytanych z żądanego sektora do pamięci ( i odwrotnie).
- **Charakterystyka dysków:**
- Prędkość obrotowa: np.: 5400 obr./min. (czyli 1 obrót zajmuje 11 milisekund).
- Liczba płyt: np.: 5 płyt (10 powierzchni)
- Ilość ścieżek: np.: 10 tyś ścieżek
- Liczba bajtów na ścieżce: np.: 10^5 na ścieżce.

# Charakterystyka dostępu do dysku

- ***Czas oczekiwania*** - czas między wydaniem rozkazy odczytu bloku z dysku a pojawieniem się zawartości tego bloku w pamięci.

  Na *czas oczekiwania* składa się:

- Czas potrzebny procesorowi i sterownikowi dysku do „***zrozumienia" żądania*** *(pomijalny).*
- Czasy potrzebne procesorowi do uzgodnienia, **czy proces ma prawo** czytać lub pisać dane *(pomijalny).*
- ***czas przeszukiwania*** - czas potrzebny na ustawienie głowic przy właściwym cylindrze *(istotny!)*

  Może zdarzyć się, że jest równy 0 (gdy głowice akurat stoją w dobrym miejscu). Zwykle jednak konieczne jest wykonanie przesunięcia głowic. Są to koszty rzędu 10~40 milisekund przez wszystkie ścieżki. Nie odbywa się to jednak liniowo.

- ***opóźnienie obrotowe*** - czas potrzebny na ustawienie odpowiednich sektorów pod głowicą (*istotny!*)

  Przy prędkości obrotowej, w której pełny obrót wykonywany jest w czasie 10 milisekund, średnio potrzeba ok. 5 milisekund na ustawienie się w odpowiednim miejscu.

- ***czas przesyłania -*** Przy transferach rzędu 10 MB/s jest to czas *pomijalny*.
- Aktualizacja bloków na dysku to suma czasów potrzebnych na wykonanie operacji: wczytania bloku do PAO, wprowadzenia zmian w kopii bloku umieszczonej w PAO, zapisania bloku na dysk, ewentualnego sprawdzenia poprawności zapisu.

# Efektywne korzystanie z Pamięci Pomocniczej

- Przy analizie algorytmów używa się zazwyczaj **modelu RAM** czyli takiego modelu obliczeń, gdzie zakłada się swobodny dostęp do danych**. Na użytek DBMS takie założenie nie jest dobre**. Nie można zakładać, że dane potrzebne podczas wykonywania algorytmu znajdują się w Pamięci o dostępie swobodnym, ponieważ w przypadku systemów baz danych zazwyczaj dane te nie mieszczą się w całości w pamięci operacyjnej. Dla ustalenia efektywnych algorytmów trzeba zatem brać pod uwagę konieczność stosowania pamięci pomocniczych (a nawet pamięci trzeciego poziomu). Dlatego algorytmy związane z DBMS będą się często mocno różniły od klasycznych, mimo rozwiązywania identycznego zadania.

**Model:**

- Komputer, w którym korzystamy z danych nie mieszczących się w PAO.
- Dane będą w PAO buforowane, ale każda porcja, do której będziemy sięgać, musi wcześniej zostać odczytana z dysku.
- Zakładamy, że dysponujemy dyskiem, dla którego czas odczytu lub zapisu jednego bloku 4 KB wynosi 15 milisekund.
- Z systemu korzysta wielu użytkowników, więc powstaje kolejka żądań, która jest zorganizowana w formie FIFO. To założenie powoduje, że nawet jeśli użytkownik czyta bloki leżące obok siebie, to można przyjąć losowość ustawienia głowicy (bo będzie ona w międzyczasie przestawiana w inne położenia, zgodnie z żądaniami pozostałych użytkowników).

# Zasada kosztu Wejścia-Wyjścia

Jeżeli zachodzi konieczność przemieszczenia bloku danych między dyskiem a pamięcią operacyjną, to czas potrzebny do odczytu bądź zapisu jest **znacznie** większy, niż czas potrzebny na przetworzenie danych w pamięci operacyjnej. Dlatego ***liczba bloków, do których należy zapewnić dostęp (odczyty lub zapisy) jest dobrym przybliżeniem czasu wykonania algorytmu*** i tę liczbę należy minimalizować.

**Przykład:**

Przeanalizujemy sytuację zapytania na relacji R, którego wynikiem jest rekord o wartości klucza K. Okaże się, że potrzebny jest indeks pozwalający zidentyfikować na dysku te bloki, w których znajdują się krotki, które posiadają wartości klucza równe K. Indeks nie będzie musial już wskazywać położenia poszukiwanej krotki w bloku, ponieważ dostęp do 4KB bloku trwa 15 milisekund. W takim czasie procesor potrafi wykonać miliony operacji, a na odnalezienie wartości K w 4KB bloku wystarczy ich tylko tysiące (nawet jeśli przeszukiwanie jest tylko w czasie liniowym). Tak więc czas przeszukiwania wewnątrz samego bloku jest znikomo mały w porównaniu z czasem dostępu do dysku i można go zaniedbać w rozważaniach.

# Sortowanie w Pamięci Pomocniczej

Jako przykład sposobu modyfikacji zmiany algorytmu na użytek *modelu we-wy* rozważymy sortowanie.

**Przykład:**

- Relacja R złożona jest z 10 milionów krotek. Każda krotka jest rekordem pewnych pól, a jednym z nich jest pole, będące ***kluczem sortowania (pole kluczowe).*** Celem algorytmu sortowania jest uporządkowania rekordów wg klucza sortowania.
- Zakładamy, że klucz jest unikatowy (chociaż w ogólnym przypadku tak być nie musi - gdy wartości mogą się powtarzać wówczas ich kolejność w wyniku jest dowolna).
- Zakładamy dodatkowo, że wszystkie rekordy mają długość 100 bajtów. Zatem mamy 1 GB danych.
- Dane zapisane są na dysku o wcześniej opisanych parametrach.
- Pamięć operacyjna ma wielkość 64 MB, ale 14 MB zajmuje system. Pozostaje więc 50 MB na buforowanie danych. Blok dysku ma 4KB (4096 B = 2^12)

Mamy więc w bloku 40 krotek (+ 96B na zapisanie dodatkowych danych, lub pozostające pustymi).

Relacja zajmuje więc 250 tyś bloków.

Pamięć ma 50 * 2^20 B więc jest to wielkość, która da się zapisać w 12 800 bloków.

(bo 50*2^20 / 4096 = 50 * 2^20/2^12 = 50 * 2^8 = 50 * 256 = 12 800.)

# Sortowanie w Pamięci Pomocniczej

- W przypadkach, gdy dane mieszczą się w całości w pamięci operacyjnej, stosuje się sprawdzone algorytmy do sortowania. Na przykład Quicksort.
- Dodatkowo stosuje się strategię sortowania tylko pól klucza, z dołączonymi wskaźnikami pełnych rekordów. Dopiero po ustawieniu kluczy i wskaźników w odpowiedniej kolejności na podstawie wskaźników zostają ustawione w odpowiedniej kolejności same rekordy.
- Taki pomysł nie sprawdza się w przypadku dużych ilości danych (czyli w opisywanym przypadku). Tu lepsza jest strategia ograniczająca przesyłanie danego bloku między pamięcią i dyskiem do jak najmniejszej ilości razy. Często tego typu algorytmy działają w kilku przebiegach.

# Merge-Sort (klasyczny)

Idea algorytmy polega na **scalaniu uprzednio posortowanych list w większe listy uporządkowane.**

Aby tego dokonać, cyklicznie porównuje się najmniejsze klucze pozostałe na resztach list wejściowych, przesuwa rekord z mniejszym kluczem na wyjście i powtarza aż do wyczerpania wszystkich elementów z jednej z list wejściowych. Wówczas po dopisaniu do wyjścia reszty z listy wejściowej, która pozostała niepusta, otrzymywany jest wynik posortowany.

**Krok podstawowy:**

- Jeśli jest lista z jednym elementem nie rób nic (bo jest posortowana)

**Krok indukcyjny:**

- Jeśli na liście do sortowania jest więcej niż jeden element, to podziel ją arbitralnie na dwie listy (takiej samej, lub prawie takiej samej długości – w przypadku nieparzystej liczby elementów na liście wejściowej).
- Sortuj rekurencyjnie te podlisty.
- ***Wynikowe listy scal w jedną listę posortowaną.***

# Merge-Sort (klasyczny) c.d.

**Przykład scalania:**

| Krok | Lista 1 | Lista 2 | Wyjście |
|---|---|---|---|
| Start | 1,3,4,9 | 2,5,7,8 | BRAK |
| 1 | 3,4,9 | 2,5,7,8 | 1 |
| 2 | 3,4,9 | 5,7,8 | 1,2 |
| 3 | 4,9 | 5,7,8 | 1,2,3 |
| 4 | 9 | 5,7,8 | 1,2,3,4 |
| 5 | 9 | 7,8 | 1,2,3,4,5 |
| 6 | 9 | 8 | 1,2,3,4,5,7 |
| 7 | 9 | BRAK | 1,2,3,4,5,7,8 |
| 8 | BRAK | BRAK | 1,2,3,4,5,7,8,9 |

- **Czas scalania w pamięci operacyjnej jest liniowy ze względu na sumę długości list wejściowych.**

# Merge-Sort (zmodyfikowany)

Zmodyfikujemy algorytm Merge-Sort tak, aby nadawał się do użytku w omawianym przypadku dużej ilości danych niemieszczących się jednorazowo w pamięci operacyjnej

***Dwufazowe wielowejściowe sortowanie przez scalanie.***

- ***Faza 1:*** Posortuj fragmenty danych, które mieszczą się w pamięci operacyjnej, tak aby każdy rekord znalazł się na dokładnie jednej posortowanej liści mieszczącej się w pamięci operacyjnej. (takich podlist może być wiele).
- ***Faza 2:*** Wszystkie posortowane listy z fazy pierwszej scal w jedną listę.

W fazie pierwszej

- cała dostępna Pamięć Operacyjna jest zapełniana blokami danych z relacji, której rekordy maja być posortowana.
- Rekordy wczytane do pamięci są sortowane
- Rekordy posortowane zapisywane są z pamięci do nowych bloków pamięci pomocniczej, tworząc jedną posortowaną podlistę.

Po zakończeniu pierwszej fazy każdy rekord oryginalnej relacji był wczytany **jeden raz do pamięci**, stał się częścią jednej posortowanej listy mieszczącej się w pamięci, która to lista została zapisana na dysk.

# Dwufazowe wielowejść. sortowanie przez scalanie

**Przykład c.d.**

Ponieważ 12 800 bloków spośród 250 000 bloków całej relacji mieści się w pamięci operacyjnej, więc potrzebne jest 20-krotne wykonanie operacji odczytu, sortowania i zapisu. Ostatnia lista będzie krótsza (tylko 6800 bloków).

***Jak długo to potrwa?***

Każdy blok jest jeden raz czytany i zapisuje się 250 tyś. **nowych** bloków. Zatem mamy 0.5 miliona operacji. Przy losowym rozmieszczeniu bloków na dysku każdy dostęp do bloku to 15 milisekund. Zatem czas we-wy Fazy 1 wynosi 7500 sekund (125 minut) (widać, że samo sortowanie listy zaczytanej do PAO nie będzie tu miało znaczenia, bo procesor z posortowaniem upora się w czasie znacznie krótszym.

***Jak zakończyć sortowanie scalaniem posortowanych podlist?***

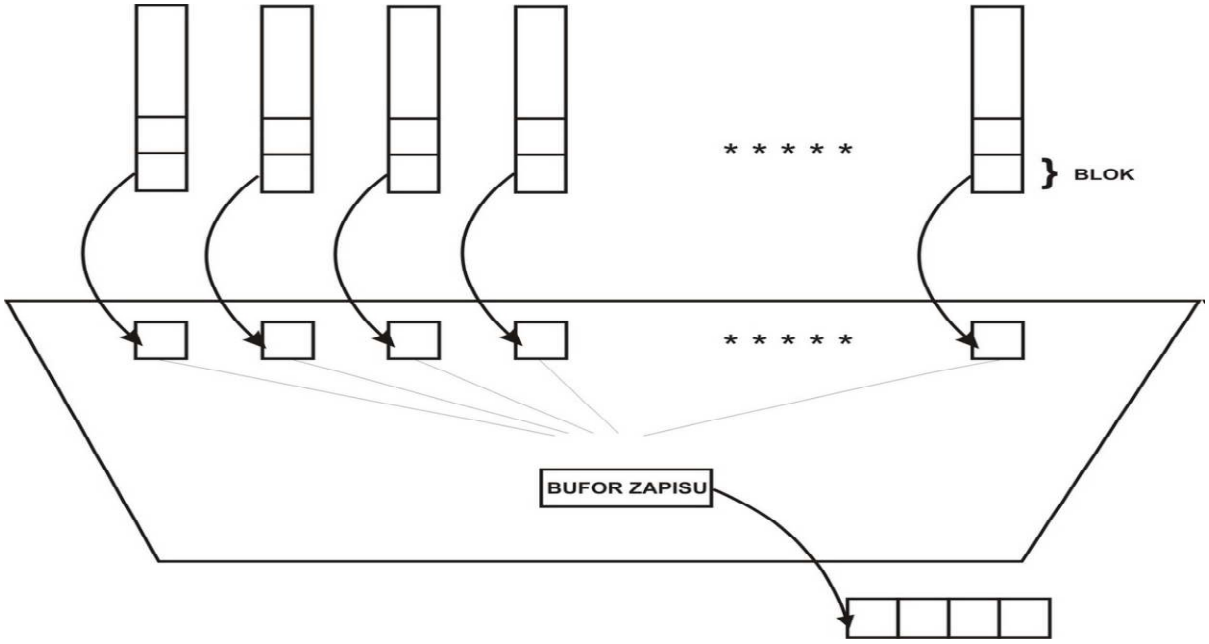
Można parami (tak jak w klasycznym algorytmie). Wtedy konieczne byłoby scalenie 20 podlist w 10, a tych ponownie w 5, itd... Korzystniej jest czytać bloki wszystkich posortowanych podlist do bufora pamięci operacyjnej:

# Dwufazowe wielowejść. sortowanie przez scalanie

- Znajdź najmniejszą wartość klucza w pierwszych elementach pozostałych na listach.

(ponieważ porównania są wykonywane w pamięci operacyjnej więc wystarcza algorytm liniowy)

- Przesuń najmniejszy element na pierwszą dostępną pozycję bloku wyjściowego (BUFORA ZAPISU).
- Jeśli blok wyjściowy jest pełny to zapisz go na dysku i zainicjuj w PAO ten sam bufor dla nowego bloku wyjściowego.
- Jeśli blok z którego pobieraliśmy najmniejszy element, nie zawiera już żadnych rekordów to czytaj następny blok z tej samej podlisty do tego samego bufora, w którym był ów blok zapisany. Jeśli na tej podliście nie ma już bloków, to pozostaw bufor pusty i nie rozważaj już elementów z tej listy do porównywania.



# Dwufazowe wielowejść. sortowanie przez scalanie

**UWAGI:**

- W drugiej fazie **nie da się przewidzieć kolejności wczytywania bloków**, bo to, kiedy elementy listy zostaną wyczerpane zależy od danych.
- Jednakże każdy blok przechowuje rekordy z jednej podlisty i jest wczytywany do pamięci operacyjnej tylko raz!
- liczba odczytów ponownie wynosi 250 tyś.
- Zapis także odbywa się wyłącznie jeden raz do bloku bufora, a blok ten jeden raz na dysk.
- Faza druga trwa więc tak jak faza pierwsza 125 minut (250 minut na posortowanie całości).

- Jaka jest optymalna wielkość bloków?
- Co w sytuacji, gdy danych jest jeszcze więcej?

# Poprawianie czasu dostępu do p.pomocniczej

C.D.N...