Materiały:

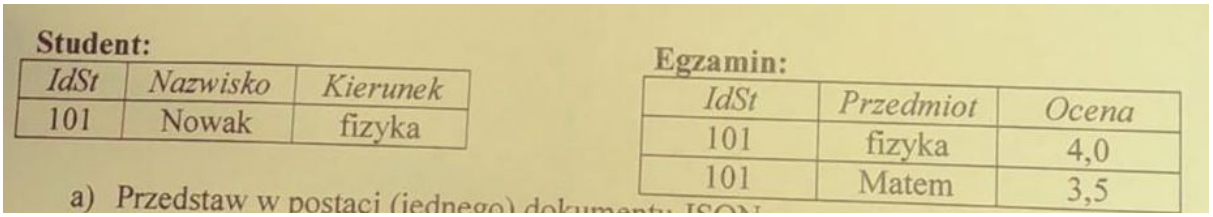
1) https://drive.google.com/drive/folders/1RupfPD8a_5MM2FIjxEfmrM2APX9K21Tr
2) https://drive.google.com/drive/folders/1mVv5asHn2EyDJNwhZ8nfnVustmXrmA41

TODO:

1) To zadanie/zadania z diagramem ER (z triggerami)
2) B-drzewa - metoda kompensacji szczególnie

2019, grupa A, zadanie 2

Dane zawarte w następujących tabelach:



Student:

IdSt	*Nazwisko*	*Kierunek*
101	Nowak	fizyka

Egzamin:

IdSt	*Przedmiot*	*Ocena*
101	fizyka	4,0
101	Matem	3,5

a) Przedstaw w postaci (jednego) dokumentu JSON.

a) Przedstaw w postaci (jednego) dokumentu JSON.

```json
{
    "IdSt": 101,
    "Nazwisko": "Nowak",
    "Kierunek": "fizyka",
    "Egzaminy": [
        {
            "IdSt": 101,
            "Przedmiot": "fizyka",
            "Ocena": 4.0
        },
        {
            "IdSt": 101,
            "Przedmiot": "matem",
            "Ocena": 3.5
        }
    ]
}
```

Analogicznie dla zadania z drugiej grupy:

```json
{
    "IdTow": "T-01",
    "Nazwa": "Drukarka",
    "GrupaTow": "KOM",
    "Sprzedaż": [
        {
            "IdTow": "T-01",
            "Data": "2019-01-13",
            "Wartosc": 4000
        },
        {
            "IdTow": "T-01",
            "Data": "2019-01-13",
            "Wartosc": 4000
        }
    ]
}
```

Zastanowiliśmy się z Martą po raz trzeci, i tak w sumie to chyba to co na początku Michał wrzucił, też jest ok, ogólnie można powiedzieć, że to zależy od kontekstu (co chcemy osiągnąć), jak powinien ten JSON (i te kolejne podpunkty) wyglądać. Więc to chyba też jest równie ok:

```json
{
    "Egzaminy": [
        {
            "Student": {
                "IdSt": 101,
                "Nazwisko": "Nowak",
                "Kierunek": "fizyka"
            },
            "Przedmiot": "fizyka",
            "Ocena": 4.0
        },
        {
            "Student": {
                "IdSt": 101,
                "Nazwisko": "Nowak",
                "Kierunek": "fizyka"
            },
            "Przedmiot": "matem",
            "Ocena": 3.5
        }
    ]
}
```

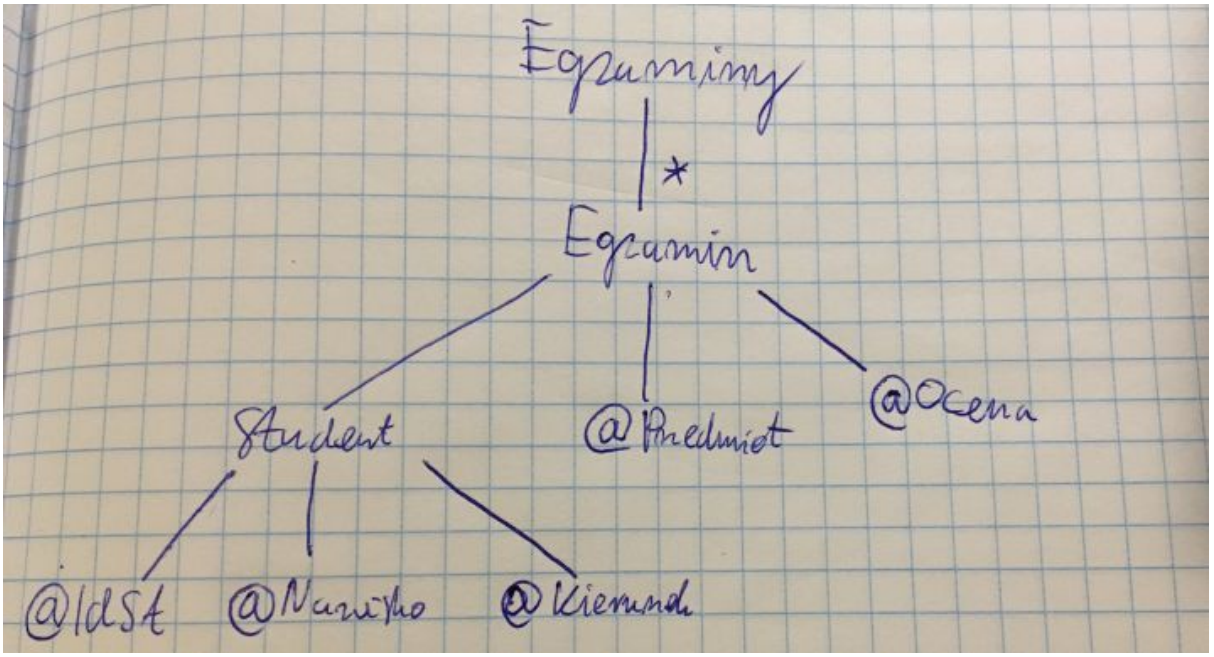
Druga grupa:

```json
{
    "Sprzedaże": [
        {
            "Towar": {
                "IdTow": "T-01",
                "Nazwa": "Drukarka",
                "GrupaTow": "KOM"
            },
            "Data": "2019-01-13",
            "Wartosc": 4000
        },
        {
            "Towar": {
                "IdTow": "T-01",
                "Nazwa": "Drukarka",
                "GrupaTow": "KOM"
            },
            "Data": "2019-01-21",
            "Wartosc": 2000
        }
    ]
}
```

b) Napisz schemat (w postaci grafu - drzewa) XML i przedstaw te dane jako (jeden) dokument DOM. Należy wykorzystać wszystkie cztery klasy wierzchołków.

Schemat (chyba to jest to, na podstawie wykładu 9 - XML, slajd 16):

* - 0 lub więcej; + - 1 lub więcej; coś jak regex, tak mi się wydaje analizując materiały

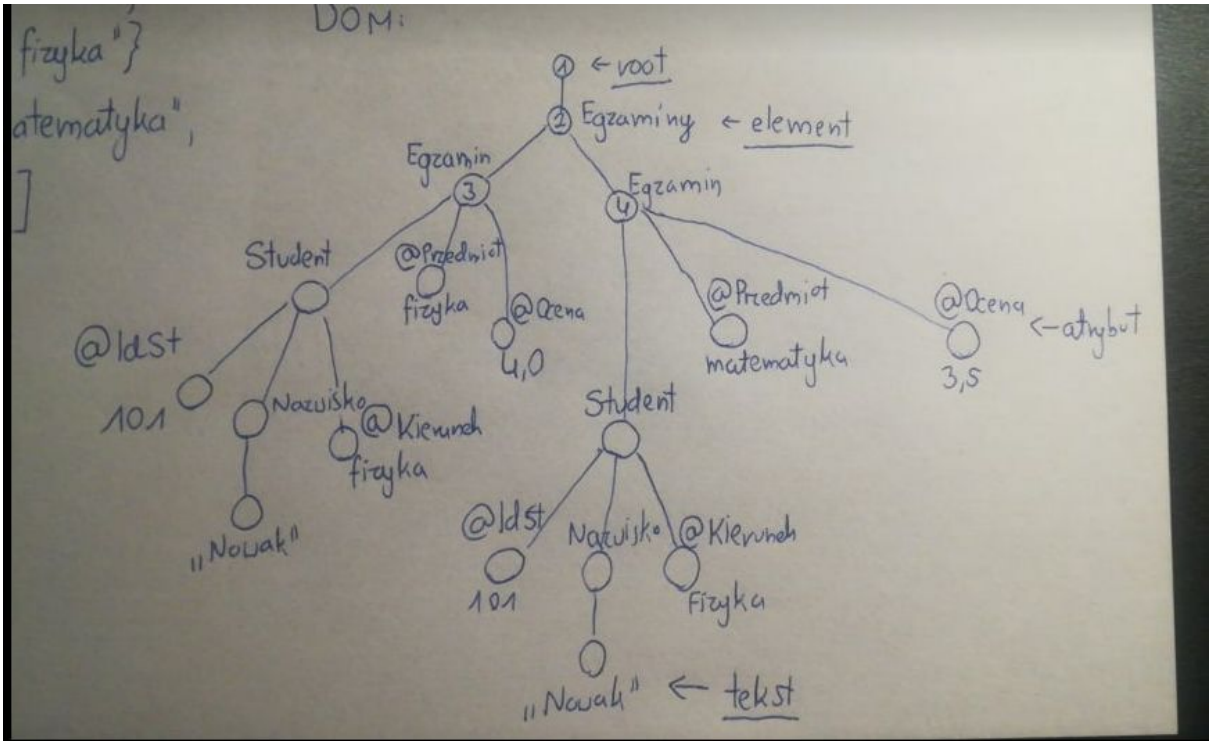


DOM:

4 klasy wierzchołków - korzeń, element, atrybut, tekst

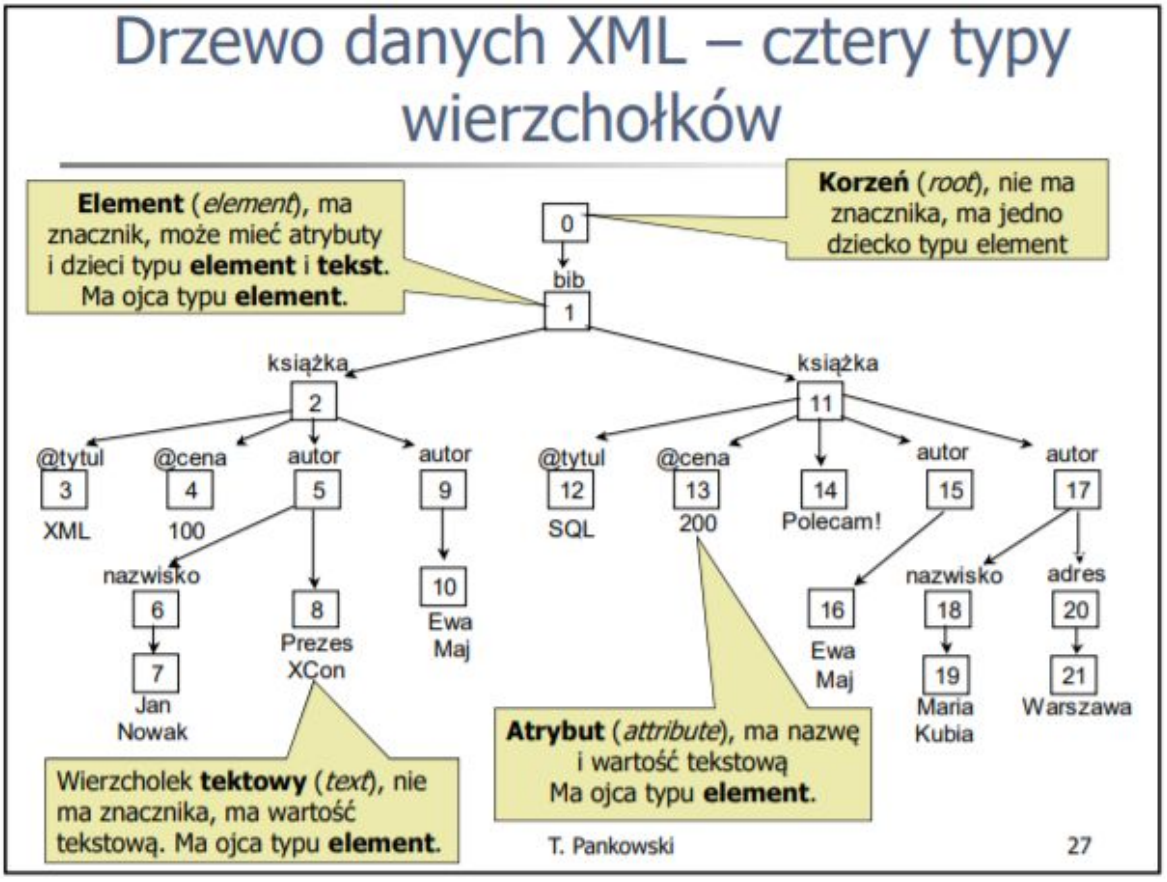
Element EGZAMIN**Y** musi być, bo korzeń może mieć tylko jedno dziecko, a potrzebujemy dwóch egzaminów, bo mamy dwie pozycje w tabeli.

Tu chyba trzeba pamiętać o numeracji tych węzłów, jest to pokazane jak zrobić w zad. 3, 2017.



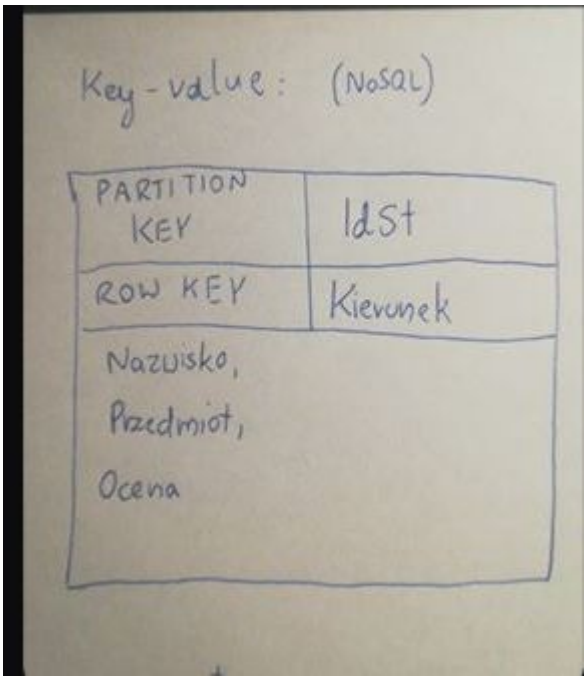
Nie mam pojęcia jak ma wyglądać drzewo XML, bo mam wrażenie, że tak samo jak DOM, więc wrzucam po prostu zwykłego XMLa:

```
<?xml version="1.0" encoding="UTF-8"?>
<Egzaminy>
        <Egzamin>
                <Student>
                        <IdSt>101</IdSt>
                        <Nazwisko>Nowak</Nazwisko>
                        <Kierunek>fizyka</Kierunek>
                </Student>
                <Przedmiot>fizyka</Przedmiot>
                <Ocena>4.0</Ocena>
        </Egzamin>
        <Egzamin>
                <Student>
                        <IdSt>101</IdSt>
                        <Nazwisko>Nowak</Nazwisko>
                        <Kierunek>fizyka</Kierunek>
                </Student>
                <Przedmiot>Matem</Przedmiot>
                <Ocena>3.5</Ocena>
        </Egzamin>
</Egzaminy>
```



W tym zadaniu chodzi właśnie o zrobienie drzewa, tak jak na powyższym zdjęciu, to jest XML DOM

c) Przedstaw w postaci modelu key/value (wg Azure Storage, jako jedną encję w tabeli).



Tu zgodnie z tym co mówił na filmiku (na dysku) w zasadzie jest pełna dowolność, partition key opisuje całą partycję więc najlepiej dać tu “klucz główny”, row key obojętnie co, pod spodem reszta danych
“Występuje duża redundancja danych i zużycie pamięci, ale nie przejmujemy się tym. Liczy się to że jest szybkie wyszukiwanie po kluczu (klucz = partition key + row key)” ~ Pank.

2019, grupa A, zadanie 3
Zarządzanie współbieżnością w aplikacjach baz danych:

a) zasady stosowania blokowania optymistycznego i jego ograniczenia

Blokowanie optymistyczne – blokowanie „lekkie” – stosowane w “zwykłych” programach (bez definiowania transakcji)
EF - Entity Framework

Blokowanie optymistycznie w EF

- Domyślnie EF wspiera współbieżność optymistyczną.
- EF zapamiętuje krotki w bazie danych zakładając, że nie zostały zmienione przez inną transakcję od chwili ich wczytania.
- Jeśli stwierdzi, że dane jednak zostały zmienione, to zgłasza wyjątek i należy wtedy rozwiązać konflikt i ponownie spróbować zapisać.

- Współbieżność optymistyczna wymaga istnienia w tabeli kolumny **RowVersion** typu **Timestamp** (lub lepiej użyć typu **RowVersion**).
- Wartość w RowVersion jest automatycznie generowana jako jednoznaczna wartość binarna, gdy wykonywana jest komenda INSERT lub UPDATE.
- Po modyfikacji struktury tabeli aktualizujemy Entity Data Model:
  designer → Update Model From Database → Refresh Student Table.
- Ustaw model współbieżności na ***fixed***, prawy klawisz na własności RowVersion.

b) uzasadnij poprawność blokowania dwufazowego (B2F)

2) 5.-Teoria.pdf, str. 8; wykład 2 - Transakcje B2F, slajd 12

Reguły działania planisty B2F

Są trzy podstawowe reguły działania planisty:

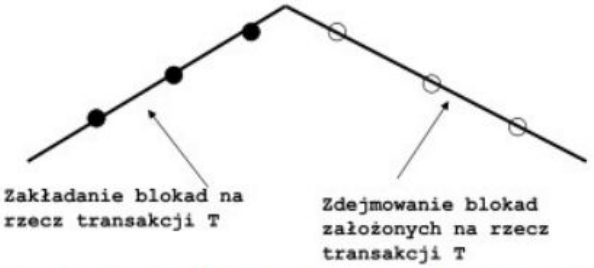
1. Jeśli operacja $p_i[x]$ może być wykonana, to planista zakłada blokadę (odpowiednio dla op. odczytu – SHARED, op. zapisu – EXCLUSIVE) na zasób x dla transakcji T_i i przekazuje tę operację do wykonania przez MD. Jeśli operacja nie może być wykonana, jest dodawana do

---

Opracował Filip Sochal

kolejki.

2. Zdjęcie blokady może nastąpić dopiero po zawiadomieniu planisty przez MD o zakończeniu wykonywanej operacji.
3. Jeśli jakakolwiek blokada założona dla transakcji T została zdjęta, nie można już zakładać na poczet T nowych blokad (reguła jest uzasadnieniem dla nazwy B2F – dana jest faza zakładania blokad i następująca po niej faza zdejmowania blokad). Sytuacja ta jest przedstawiona na poniższym rysunku:



Ilustracja 1. Schemat zakładania i zdejmowania blokad przez planistę blokowania dwufazowego (B2F).

Istotne jest, że sposób zdejmowania blokad jest związany z realizacją przetwarzania odpowiadającego różnym poziomom izolacji.

Cechy B2F

Planista blokowania dwufazowego ma dwie cechy charakterystyczne:

- każda historia przetwarzania transakcji utworzona przez planistę B2F jest poprawna,
- zasada działania planisty B2F może doprowadzać do zakleszczeń.

Każda historia przetwarzania utworzona przez planistę B2F jest poprawna – to znaczy, że dla każdej pary transakcji T i T' zachodzi warunek: **jeśli $T < T'$ to nieprawda, że $T' < T$.**

Dowód:

1. Jeśli $T < T'$ to założenie blokady na rzecz T' musi być poprzedzone zdjęciem blokady założonej na rzecz T.
2. Dana jest sytuacja w historii H: zachodzi zarówno $T < T'$, jak i $T' < T$. Wówczas uwzględniając punkt 1. widać, że założenie blokady na rzecz T musi być poprzedzone zdjęciem blokady założonej wcześniej na T, co jest sprzeczne z trzecią regułą B2F.
3. Z punktu 2. wynika zatem, że rozważana historia przetwarzania H nie mogłaby być utworzona przez planistę B2F, a zatem twierdzenie o poprawności wszystkich historii przetwarzania transakcji utworzonych przez planistę B2F jest prawdziwe.

2019, grupa B, zadanie 3

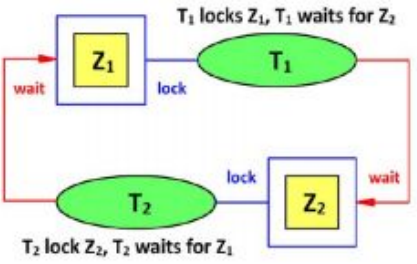
Zasady blokowania dwufazowego (B2F) w zarządzaniu transakcjami:

a) podaj przykład ilustrujący powstawanie zakleszczeń przy stosowaniu B2F i uzasadnij, w jaki sposób wprowadzenie blokady typu U wpływa na zmniejszenie liczby zakleszczeń

Zakleszczeniem (ang. deadlocks) – nazywamy sytuację, w której co najmniej dwie transakcje wzajemnie czekają na zwolnienie zasobów (danych) naprzemiennie przez siebie zablokowanych. Wtedy:

- dostęp do pewnych zasobów jest niemożliwy wskutek ich zablokowania
- transakcja, która ma już pewne zablokowane dla siebie zasoby żąda dostępu do innych, również zablokowanych zasobów (przez inna transakcję)
- zwolnienie zasobów nastąpi dopiero po zakończeniu jednej z transakcji
- dochodzi do zakleszczenia – obie transakcje czekają w nieskończoność na zasoby

Przykład zakleszczenia:



Blokada typu U - Update. Zakładany na dane, które mogą być aktualizowane. Zapobiega zakleszczeniom w sytuacji, gdy wiele transakcji czyta dane z zamiarem ich aktualizacji. Nie można założyć tej blokady, gdy inne operacje wykonują modyfikacje danych wymagające wyłączności (dzięki temu nie ma zakleszczeń).

b) omów implementację poziomów izolacji za pomocą mechanizmów B2F
Wykład 2 - Transakcje B2F, slajdy 7-10

Realizacja poszczególnych poziomów izolacji za pomocą B2F

Założenia

Dane jest 5 podstawowych założeń planisty blokowania dwufazowego:

- blokada *S* (SHARED) jest konieczna do odczytania danych,
- blokada *X* (EXCLUSIVE) jest konieczna do zapisania/modyfikacji/usunięcia danych,
- blokadę *S* dla tych samych danych może uzyskać dowolna liczba transakcji jednocześnie,
- blokadę *X* dla konkretnych danych może uzyskać tylko jedna transakcja jednocześnie,
- blokady *S* i *X* nie mogą być jednocześnie założone na te same dane dla dwóch różnych transakcji.

Poziom 0

Dana zablokowana na czas realizacji operacji czytania (*S*) jest natychmiastowo odblokowywana po zakończeniu tej operacji, natomiast dana zablokowana na czas realizacji operacji zapisu (*X*) zmienia się po wykonaniu operacji na blokadę *S*.

Poziom 1

Dana zablokowana na czas realizacji operacji czytania (*S*) jest natychmiast odblokowywana po zakończeniu tej operacji, zaś dana zablokowana na czas realizacji operacji zapisu (*X*) jest odblokowywana po zatwierdzeniu transakcji.

Poziom 2

Zdejmowanie obu blokad następuje dopiero po zatwierdzeniu transakcji.

Poziom 3

Poza tym, że zdejmowanie obu blokad następuje dopiero po zatwierdzeniu transakcji, są także utrzymywane blokady sterowane przez warunki, np. zablokowana jest możliwość takich aktualizacji, które wpływają na rekordy spełniające określony warunek.

2017, zestaw A, zadanie 2

Algorytmy haszowania danych w bazach danych i ich wpływ na złożoność operacji wyszukiwania i modyfikowania (dołączania i usuwania) danych.

Haszowanie – pośrednie i bezpośrednie

Kubełek może być implementowany na wiele sposobów:

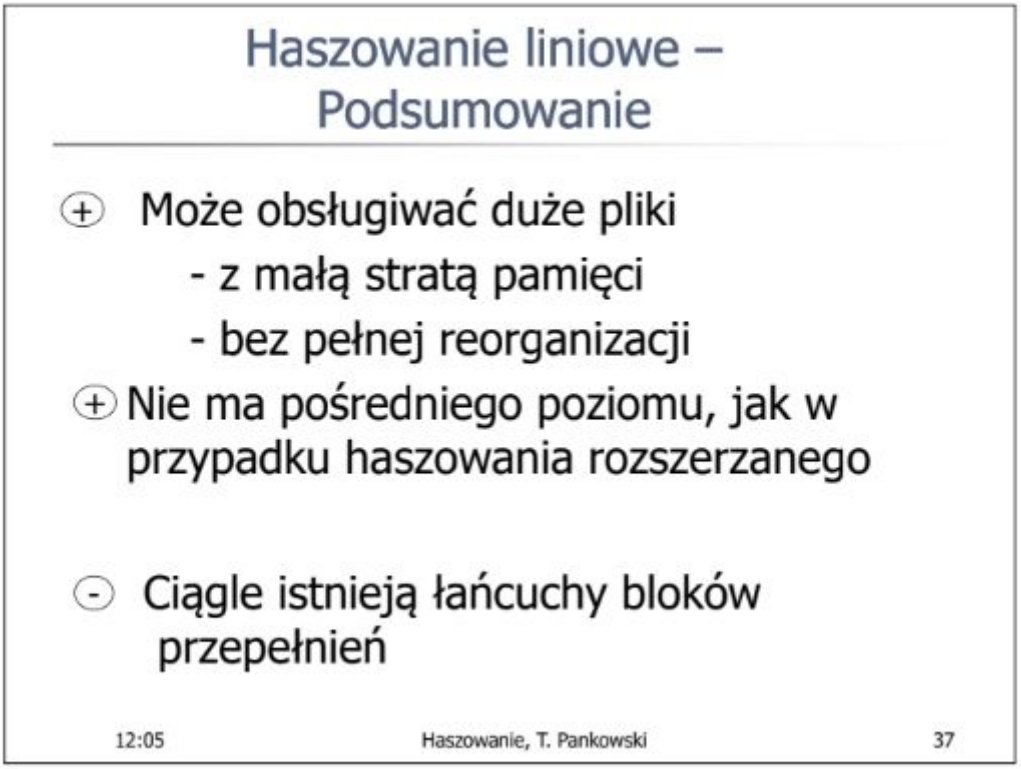
- wartość funkcji haszującej jest traktowana jako pozycja w pewnej *tablicy haszowej*, która zawiera wskaźniki na początek listy bloków rekordów przyporządkowanych do danego kubełka (*hashowanie pośrednie*);
- wartość funkcji haszującej wskazuje bezpośrednio pierwszy blok rekordów przyporządkowanych do danego kubełka (*hashowanie bezpośrednie*);

Haszowanie rozszerzalne – podsumowanie

(+) Obsługuje pliki o wzrastających rozmiarach
- przy małym marnowaniu pamięci
- bez potrzeby pełnej reorganizacji

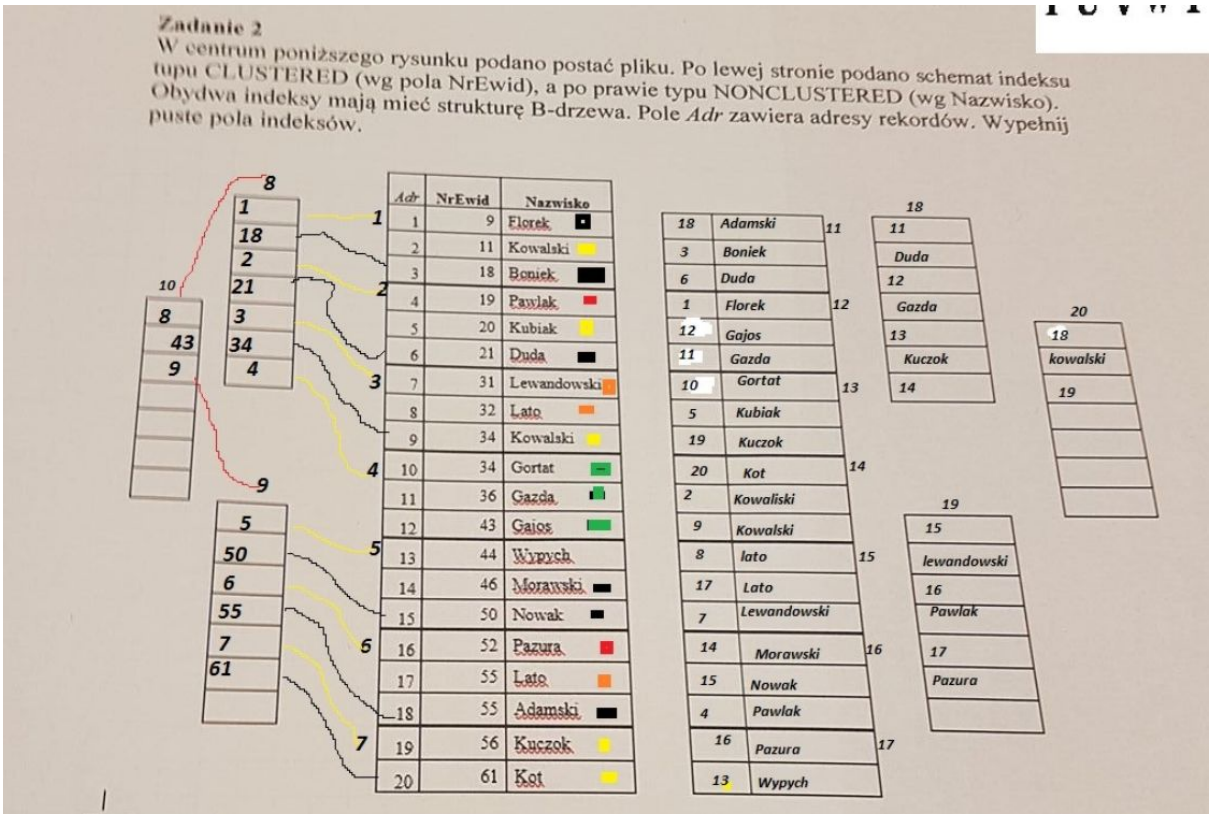
(-) Pośredni dostęp
(Mała wada, gdy tablica haszowa w pamięci)

(-) Podwajający się rozmiar tablicy haszowej
(Tabela haszowa może nie mieścić się w pamięci)



Cały wykład 7 - Haszowanie lub 2) 5.-Teoria.pdf, od str. 12
Nie wrzucam tu żadnych materiałów bezpośrednio, bo mam wrażenie, że wszystko tam jest ważne :/

2017, zestaw B, zadanie 2
Nie wiadomo o co tu chodzi, ale podobno rozwiązane dobrze:



Schemat:
Tabela ta z 8 na górze (wypełniona: 1-18-2-21-3-34-4):
Z każdego z 3 pierwszych bloków bierzemy ostatni wiersz (Boniek, Duda, Kowalski) i ich nrEwid (18,21,34), wpisujemy do tabeli, na górze numery bloków.

Tabela poniżej (5-50-6-55-7-61):
Tak samo jak ta powyżej ale 3 ostatnie bloki.

Tabela pierwsza od lewej:
Bierzemy środkowy blok (nr 4) i ostatni wiersz z tego bloku (Gajos), wpisujemy jego nrEw (43) i jakieś nr bloków z tamtych tabel (chyba).

Po prawej duża tabela:
Alfabetycznie nazwiska + ich Adr

Następne tabele tak samo jak te po lewej, numerowanie bloków kontynuujemy od 10.

2017, zestaw A, zadanie 1
Pojęcie transakcji, postulaty ACID. Poziomy izolacji transakcji i występowanie konfliktów. Przykłady ilustrujące problemy z przetwarzaniem transakcji na różnych poziomach izolacji.

Co to jest transakcja?

Transakcją T_i nazywamy ciąg operacji na wspólnej bazie danych. Do operacji, jakie można wykonywać należą:

- $r_i[x]$ – czytanie zasobu x przez transakcję T_i,
- $w_i[x]$ – zapisywanie zasobu x przez transakcję T_i,
- a_i – anulowanie/cofnięcie transakcji T_i,
- c_i – zatwierdzenie transakcji T_i.

Właściwości ACID

ACID jest zbiorem właściwości, jakie powinny posiadać transakcje i protokoły nimi zarządzające. Składa się z czterech zasad:

- **niepodzielność (*atomicity*)** – transakcję należy traktować, jako niepodzielną całość, jeśli któraś z operacji się nie powiedzie, należy cofnąć/odrzucić całą transakcję,
- **spójność (*consistency*)** – transakcja, która zaczyna się w stanie spójnym bazy danych, powinna pozostawić tę bazę również w spójnym stanie; w przeciwnym wypadku należy taką transakcję odrzucić,
- **odizolowanie (*isolation*)** – zmiany wprowadzane przez niezatwierdzone transakcje nie są widoczne dla innych transakcji. W zależności od poziomu izolacji można uzyskać większą współbieżność wykonywanych operacji, kosztem podatności na błędy i vice versa,
- **trwałość (*duration*)** – zmiany wprowadzone przez zatwierdzone transakcje są trwałe i odporne na awarie (z wyłączeniem awarii fizycznych).

Konfliktowość operacji

- Z punktu widzenia stosowanego protokołów (algorytmów) zarządzania transakcjami istotne jest przyjęcia pojęcia *konfliktowości* operacji, tzn. przyjęcia, jakie operacje są konfliktowe, a jakie nie. Z góry można określić, jakie operacje nigdy nie będą konfliktowe.
- Operacje $o_i[x]$ i $p_j[y]$ są konfliktowe, jeśli:
  - pochodzą z różnych i aktywnych transakcji,
  - co najmniej jedna z nich jest operacją zapisu,
  - dane x i y nie są rozłączne, tzn. $x = y$ lub $x \cap y \neq \emptyset$,
  - druga z operacji, $p_j[y]$, powoduje zmianę zbioru danych x (wyznaczonego przez pewną formułę φ), na których działa pierwsza operacja, $o_i[x]$.

(c) T. Pankowski, Transakcje 30

Poziomy izolacji: wykład 01-transakcje str. 32-42
W skrócie:

- 0 read uncommited - pozwala na czytanie przez transakcję danych niezatwierdzonych.
- 1 read commited - Wprowadza zakaz czytania danych z transakcji niezatwierdzonych, jednakże nadal jest dopuszczalne zapisywanie danych w transakcjach niezatwierdzonych.

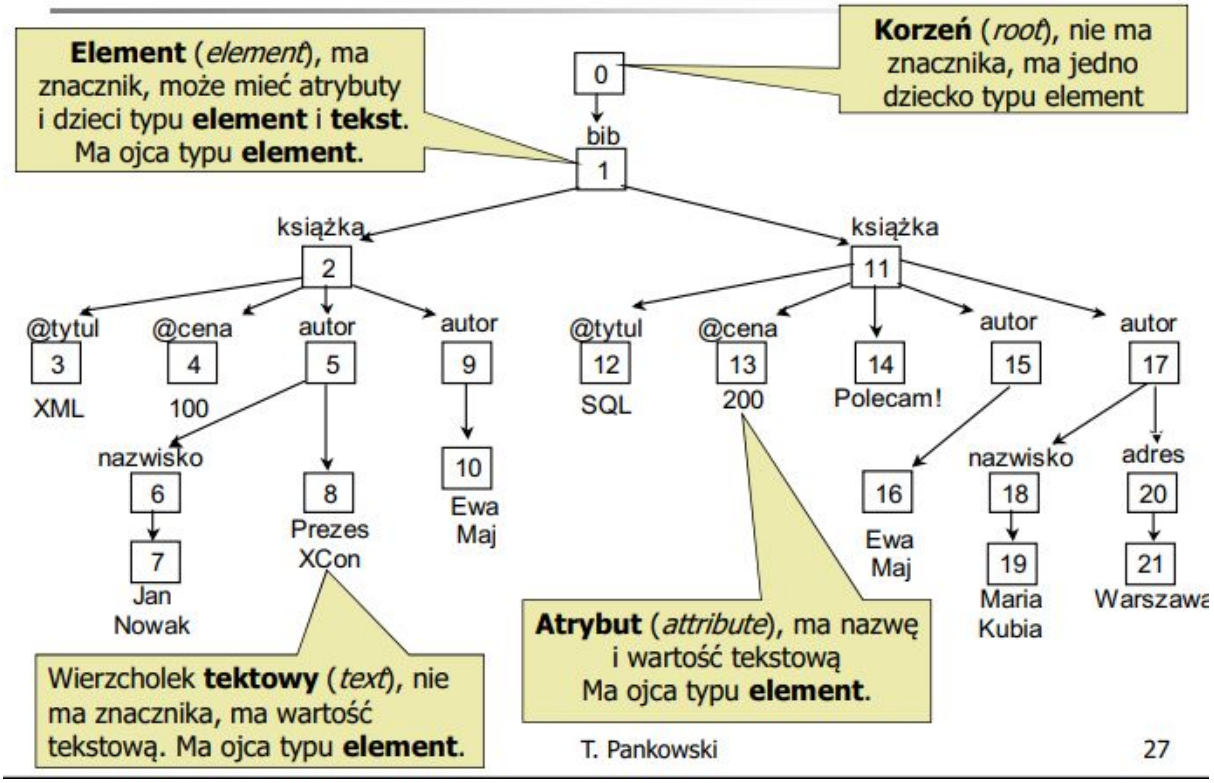
- 2 repeatable read - Wprowadza zakaz zapisywania w transakcjach niezatwierdzonych.
- 3 serializable - historia przetwarzania transakcji jest szeregowalna, a więc jest równoważna historii szeregowej.

2017, zestaw A, zadanie 3
Dany jest następujący dokument XML:

```
<bib>
  <książka tytuł="XML" cena="100">
    <autor>
      <nazwisko>Jan Nowak</name>
      Prezes XCon
    </autor>
    <author>Ewa Maj</author>
  </książka>
<książka tytuł="SQL" cena="200">
    Polecam
    <autor>Ewa Maj</autor>
    <autor>
      <nazwisko>Maria Kubiak</nazwisko>
      <adres>Warszawa</adres>
    </autor>
  </książka>
</bib>
```

1. Przedstaw dokument w postaci modelu DOM.

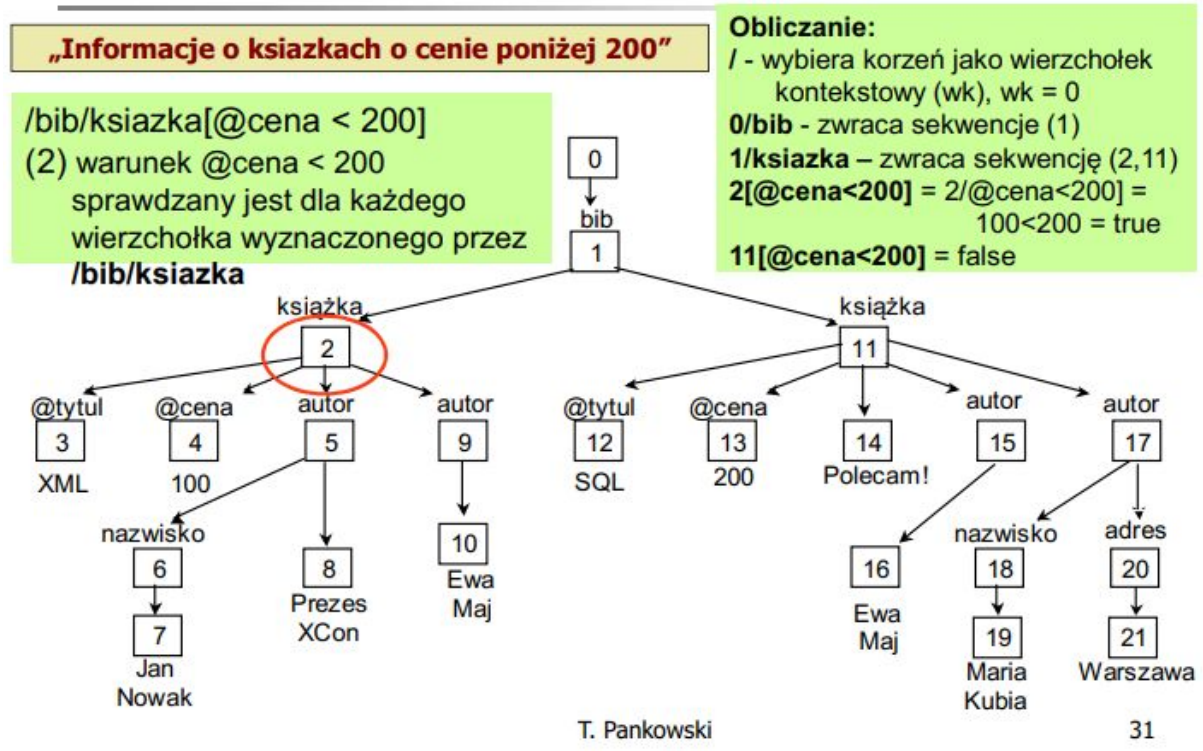
Wykład 9 - XML, slajd 27



2. Omów postać wyrażeń języka XPath i zapisz w XPath następujące polecenie: “Podaj dostępne informacje o autorach książek, których cena jest mniejsza niż 200”

/bib/ksiazka[@cena < 200]/autor

Inne przykłady w wykładzie 9 - XML, od slajdu 30, np.:



- Język XPath
  - pozwala wybierać fragment dokumentu XML-owego,
  - wyrażenie XPath definiuje scieżkę w drzewie danych podając etykiety (znaczniki elementów lub nazwy atrybutów) wierzchołków, kierunki przechodzenia drzewa i warunki, jakie mają spełniać wybierane wierzchołki,
  - każdy wybrany wierzchołek określa poddrzewo, którego jest korzeniem,
  - bieżący wierzchołek nazywamy *wierzchołkiem kontekstowym*,
  - jeśli wyrażenie E wyznaczyło sekwencje S i x $\in$ S, to parę (S,x) nazywamy *kontekstem* wykonywania kolejnego wyrażenia.

Wyrażenia XPath

Path ::= */Step*$_1$ */ Step*$_2$ */ . . . / Step*$_n$

Step ::= *Axis* :: *Node-test* [*Predicate*]*

Axis ::= child | attribute | parent | self | descendant-or-self | . . .

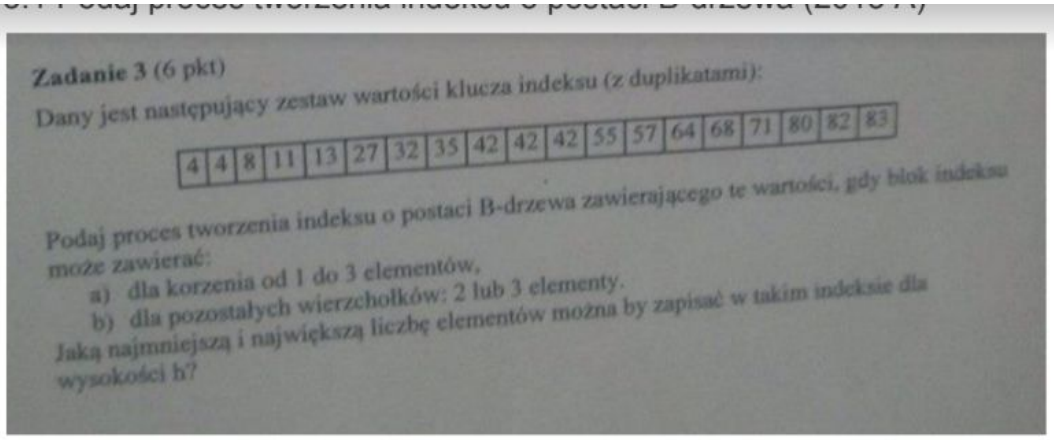
Predicate ::= *Path* | position() Θ *n* | last() Θ *n* | position() Θ *last*() | . . .

Przykład:

/bib//∗[position() = last()]

postać rozwinięta:

/child::bib/descendant-or-self::node()/child:: ∗[position() = last()]



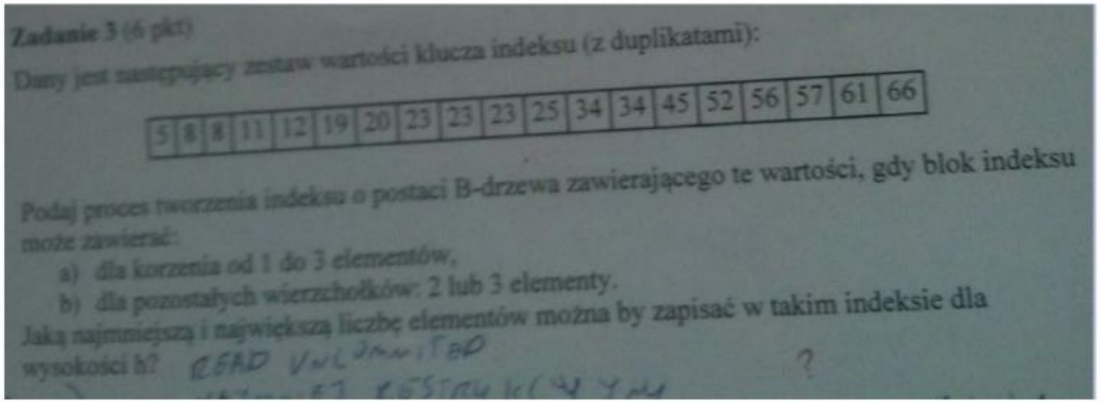
ODPOWIEDŹ 3.1

Ktoś to musi rozpisać, ja nwm dlaczego tak się dzieje, to jest metoda kompensacji a nie podziału jak coś .bo mie chuj szczeli z tym ej, ale na dół z tym, tam jest mój screen poniżej XD

to jest ważne, spędziłem nad tym za dużo czasu
ogólnie drzewa są zrozumiałe, jak robisz metodą podziału i nie ma założeń typu wierzchołek musi mieć minimum 2 klucze(elementy)
To zara spróbuję poszukać

Metoda podziału przy tworzeniu b-drzew
https://www.cs.usfca.edu/~galles/visualization/BTree.html
https://www.youtube.com/watch?v=C_q5ccN84C8&t=508s

3.2 Podaj proces tworzenia indeksu o postaci B-drzewa (2015 B)



Zadanie 3 (6 pkt)

Dany jest następujący zestaw wartości klucza indeksu (z duplikatami):

5	8	8	11	12	19	20	23	23	23	25	34	34	45	52	56	57	61	66

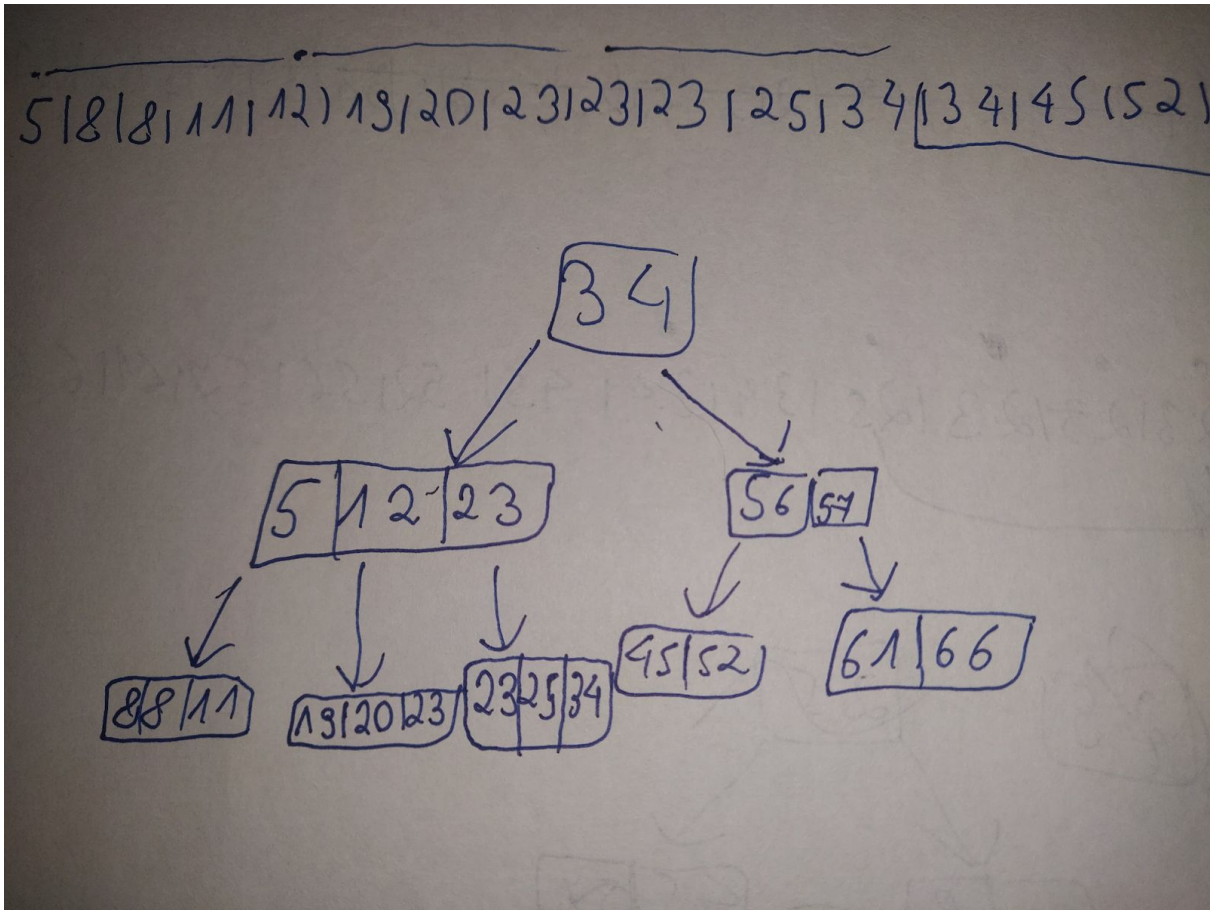
Podaj proces tworzenia indeksu o postaci B-drzewa zawierającego te wartości, gdy blok indeksu może zawierać:

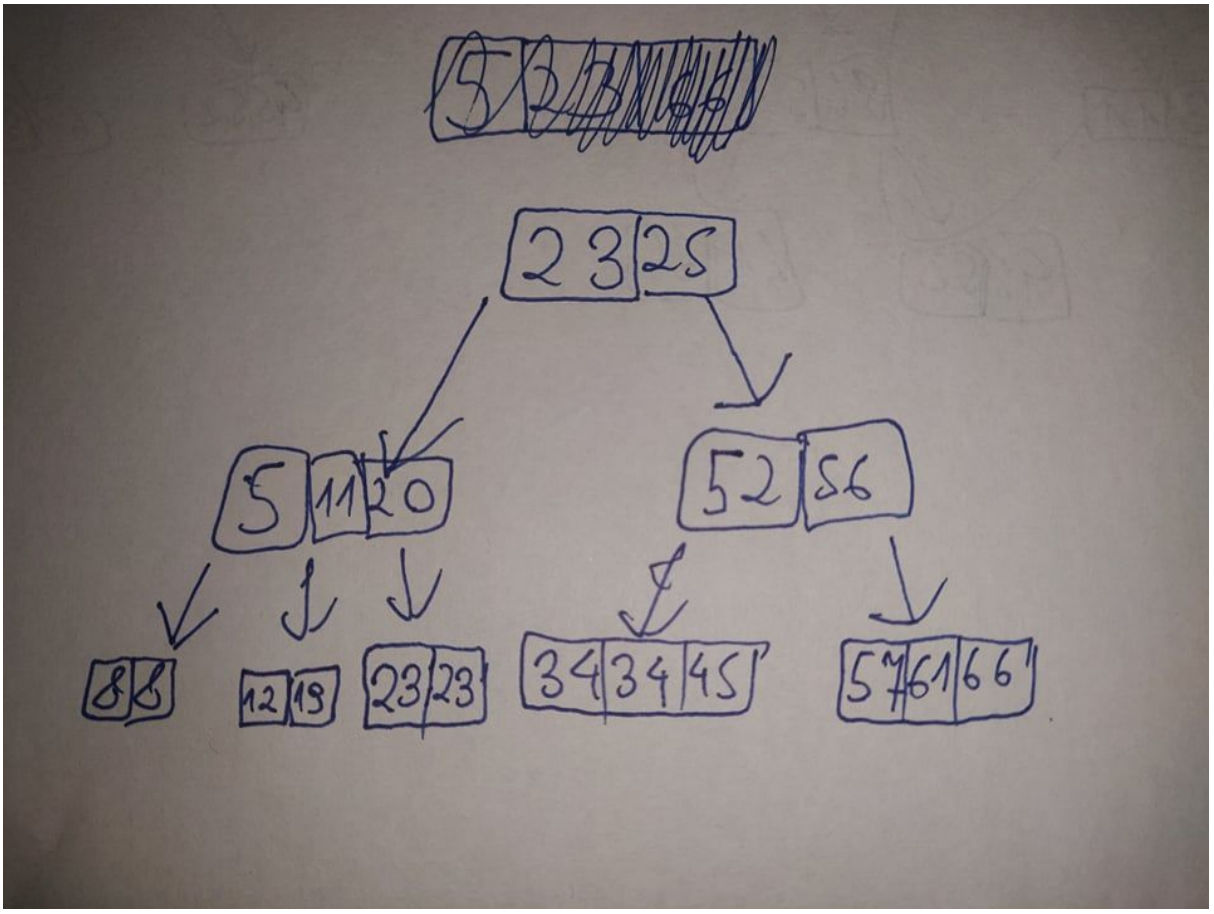
a) dla korzenia od 1 do 3 elementów,

b) dla pozostałych wierzchołków: 2 lub 3 elementy.

Jaką najmniejszą i największą liczbę elementów można by zapisać w takim indeksie dla wysokości h?

Wydaje mi się, że w tego typu zadaniu jest dowolność, żeby zorganizować to drzewo, byle tylko zgadzały się założenia, zrobię tutaj dwa przykłady tego zadania, jeden z jednym elementem w korzeniu, drugi z dwoma/trzema, ale totalnie nie wiem czy to jest dobrze~MK





Poziom i	Liczba wierzchołków przy min wypełnieniu W_{min}	Liczba wierzchołków przy max wypełnieniu W_{max}
1	1	1
2	2	$2m+1$
3	$2(m+1)$	$(2m+1)^2$
...	...	...
h	$2(m+1)^{h-2}$	$(2m+1)^{h-1}$

Nmin = 2(m+1)^(h-1) -1
Nmax = 2(m+1)^h - 1
Wierzchołek - blok

- N – liczba elementów w pliku (i w indeksie)