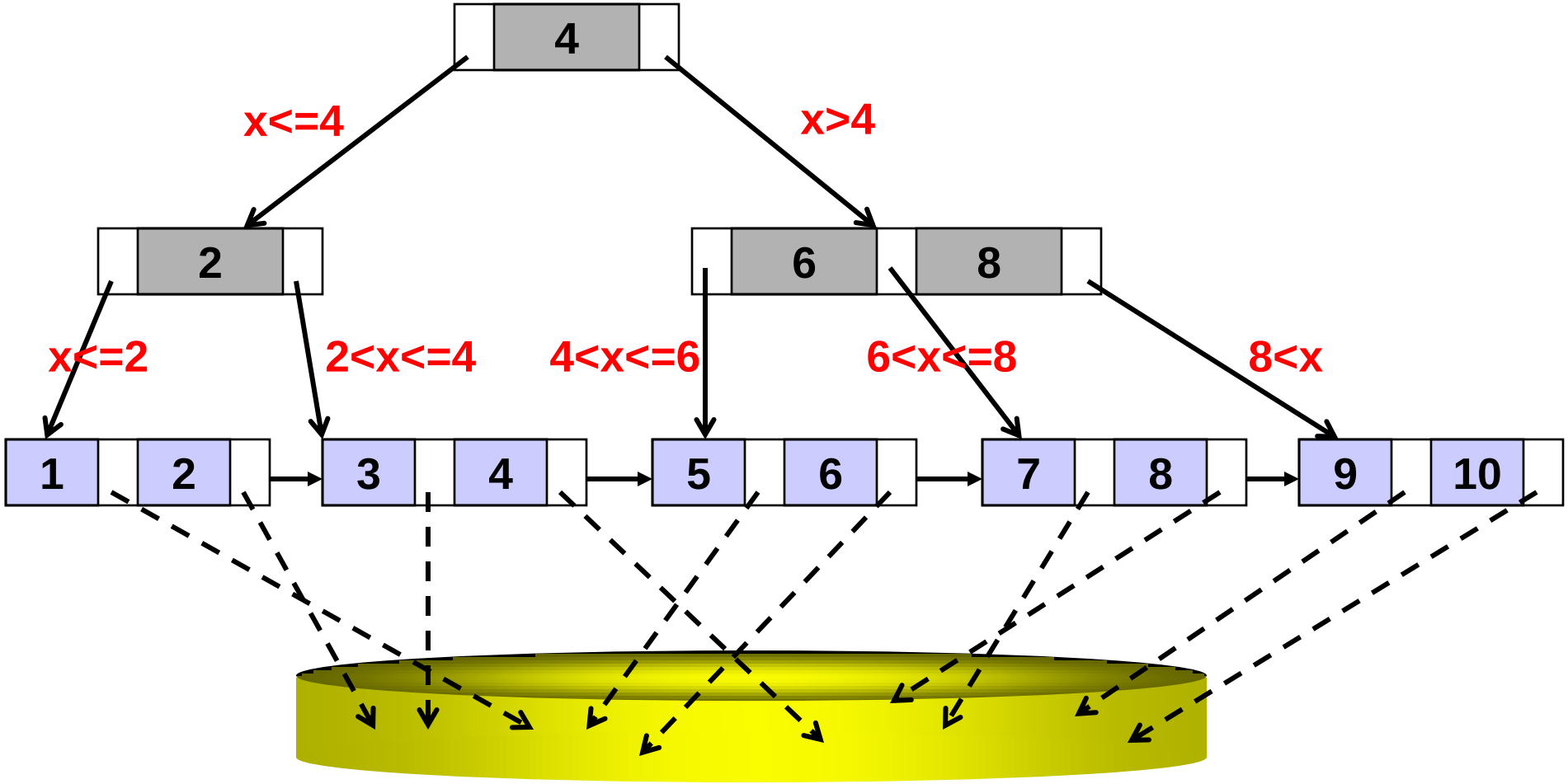
Hurtownie danych - przegląd technologii

Robert Wrembel
Politechnika Poznańska
Instytut Informatyki
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Efektywność przetwarzania OLAP

1. **Indeksowanie**
   - **B-drzewo**
   - **Indeks bitmapowy**
   - **Indeks połączeniowy**
   - **Bitmapowy indeks połączeniowy**
2. **Perspektywa zmaterializowana**
   - **koncepcja**
   - **przepisywanie zapytań**
   - **odświeżanie**
   - **wybór perspektyw do materializacji**
3. **Optymalizacja GROUP BY**
4. **Partycjonowanie**
5. **Kompresja**
6. **Przetwarzanie równoległe**

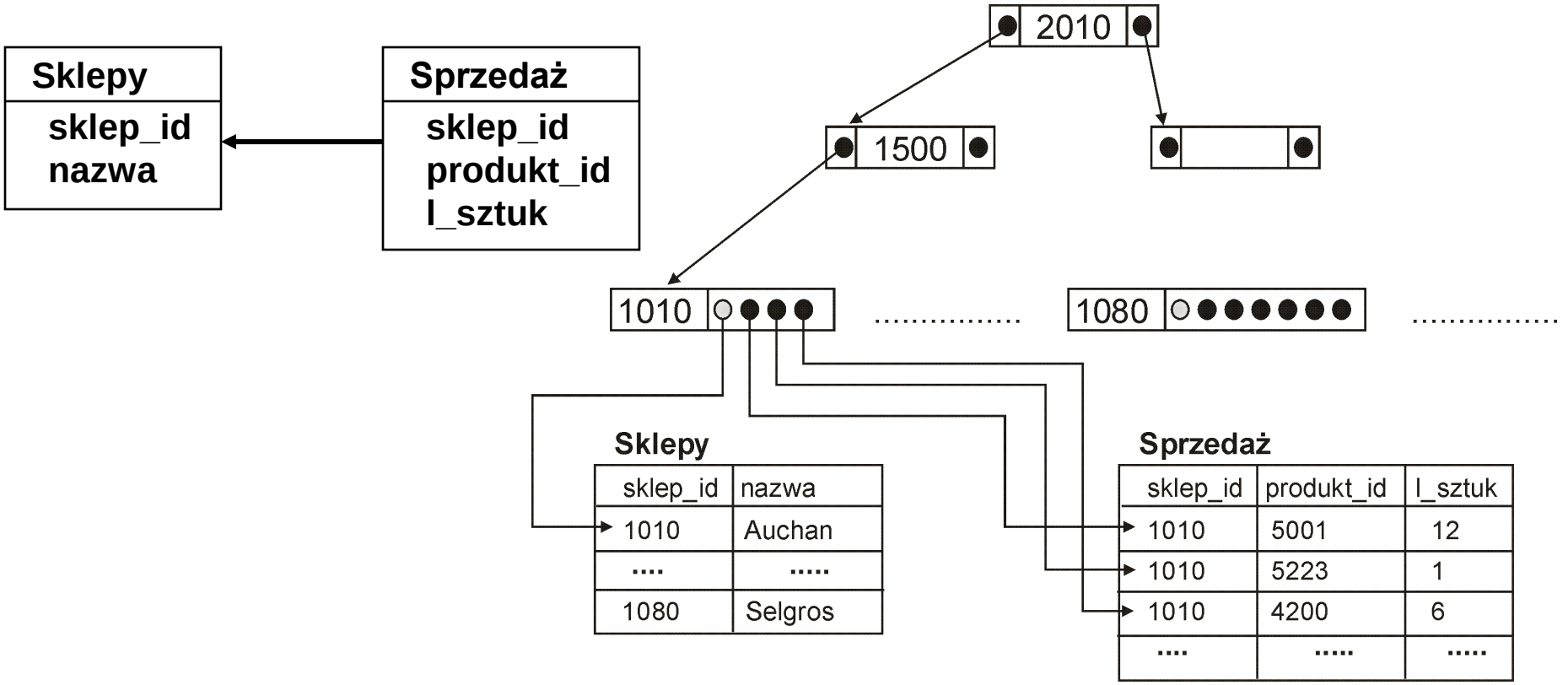
B-drzewo (B-tree)



Indeks połączeniowy (join index)

- **Zmaterializowany wynik połączenia**

np. indeks na atrybucie Sklepy.sklep_id



Wprowadzenie (2)

- Zapytania analityczne często nie są selektywne
  - wybierają kilkadziesiąt % rekordów tabeli
- Indeksy B-drzewo są efektywne dla selektywności max 20%
- Zastosowanie innego rodzaju indeksu do indeksowania danych w HD
  - indeks bitmapowy

Indeks bitmapowy - definicje

- Krotność atrybutu - szerokość dziedziny atrybutu
- Mapa bitowa - wektor bitów
  - bit odpowiada rekordowi tabeli
- Indeks bitmapowy
  - zbiór map bitowych - jedna mapa opisuje jedną wartość z dziedziny
- Dostęp do map bitowych
  - 2-wymiarowa tablica
  - indeks B-drzewo
  - ...

typ
sedan
coupe
sport
limuzyna
sport
sport
sport
sport
sport
limuzyna
limuzyna
sport
sedan
sport

typ			
coupe	*limuzyna*	*sedan*	*sport*
0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
0	1	0	0
0	1	0	0
0	0	0	1
0	0	1	0
0	0	0	1

Indeks bitmapowy - wykorzystanie

```
select count(*) from sprzedaż
where  marka in ('Audi', 'Ford')
and    typ = 'sport'
and    plec='K';
```

Audi	OR	*Ford*	=	M_1	AND	*sport*	=	M_2	AND	*K*	=	M_3
0		1		1		0		0		0		0
1		0		1		0		0		1		0
0		1		1		1		1		0		0
0		0		0		0		0		0		0
0		0		0		1		0		1		0
1		0		1		1		1		1		1
0		0		0		1		0		1		0
0		1		1		1		1		1		1
1		0		1		1		1		1		1
0		0		0		0		0		0		0
0		0		0		0		0		1		0
0		0		0		1		0		1		0
0		0		0		0		0		0		0
0		0		0		1		0		1		0

Indeks bitmapowy - charakterystyka (1)

- **Mały rozmiar dla atrybutów o małej krotności**
- **Przykład**
  - **l_rekordów = 1 000 000**
  - **krotność$_A$ = 4**
  - **adres_rekordu = 10B (Oracle)**
  - **indeks bitmapowy na atrybucie A**
    - **4 mapy bitowe: 4 x (1 000 000 / 8) = 4 x 125kB = 500kB**
  - **indeks B$^+$-drzewo na atrybucie A**
    - **1 000 000 x 10B = 10MB**

Indeks bitmapowy - charakterystyka (2)

- **Szybkość przetwarzania**
  - **mały indeks ⇨ mała liczba operacji we/wy**
  - **przetwarzanie w RAM**
  - **przetwarzanie 64 bitów w jednym cyklu zegara**
  - **szybkie operacje AND, OR, NOT, COUNT na mapach bitowych**
  - **nie nadaje się do obsługi operatora LIKE**

Indeks bitmapowy - charakterystyka (3)

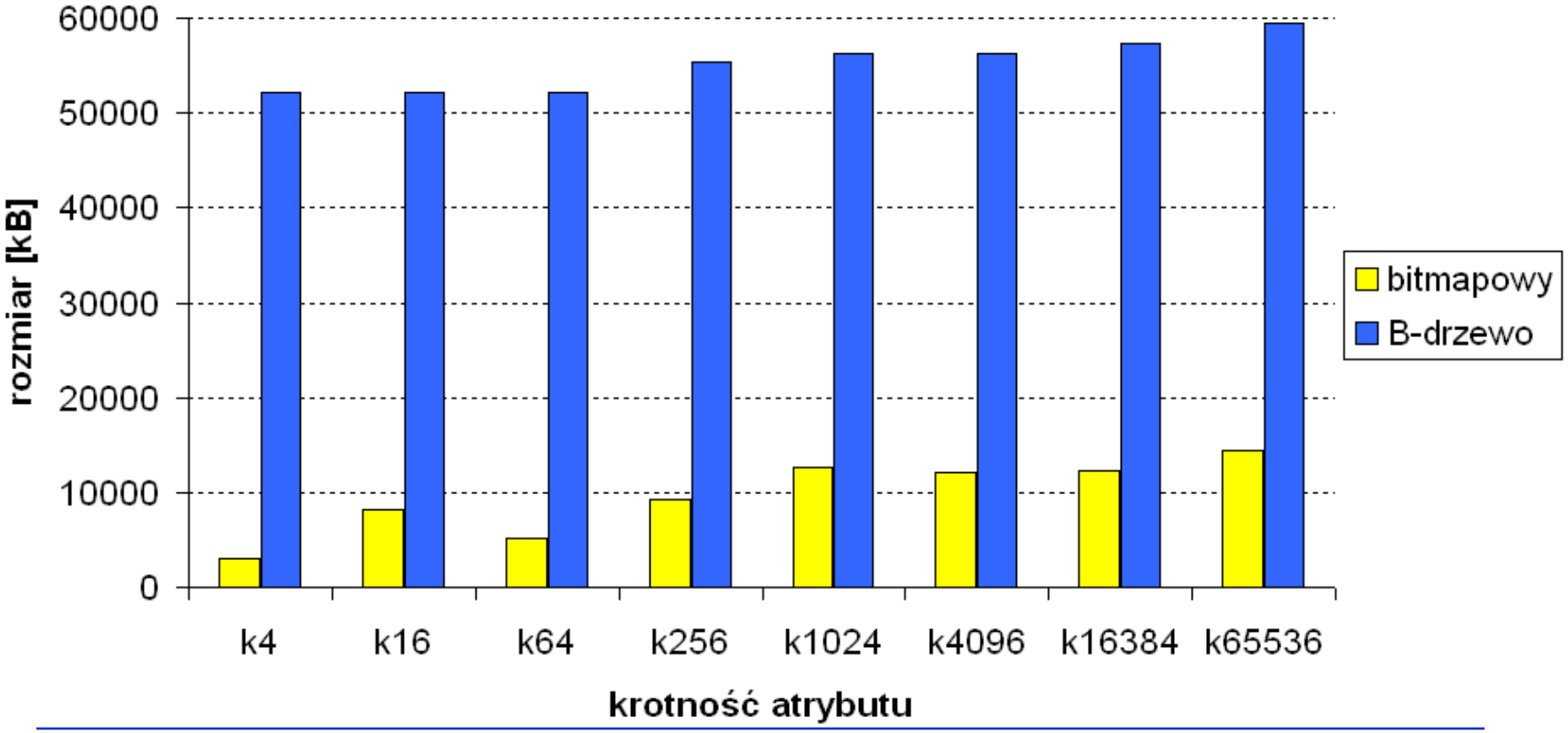
- **Duży rozmiar dla atrybutów o dużej krotności**
- **Przykład**
  - **l_rekordów = 1 000 000**
  - **krotność$_A$ = 1024**
  - **adres_rekordu = 10B**
  - **indeks bitmapowy na atrybucie A**
    - **1024 mapy bitowe: 1024 x (1 000 000 / 8) = 1024 x 125kB = 128MB**
  - **indeks B$^+$-drzewo na atrybucie A**
    - **1 000 000 x 10B = 10MB**

Indeks bitmapowy – charakterystyka (5)

- Koszty uaktualniania indeksów w trakcie pracy systemu
  - wstawianie rekordów – zwiększanie długości map bitowych
  - usuwanie rekordów – zmniejszanie długości map bitowych
  - modyfikowanie rekordów – operacje na 2 mapach
  - współbieżność – blokowane są ciągłe obszary map bitowych

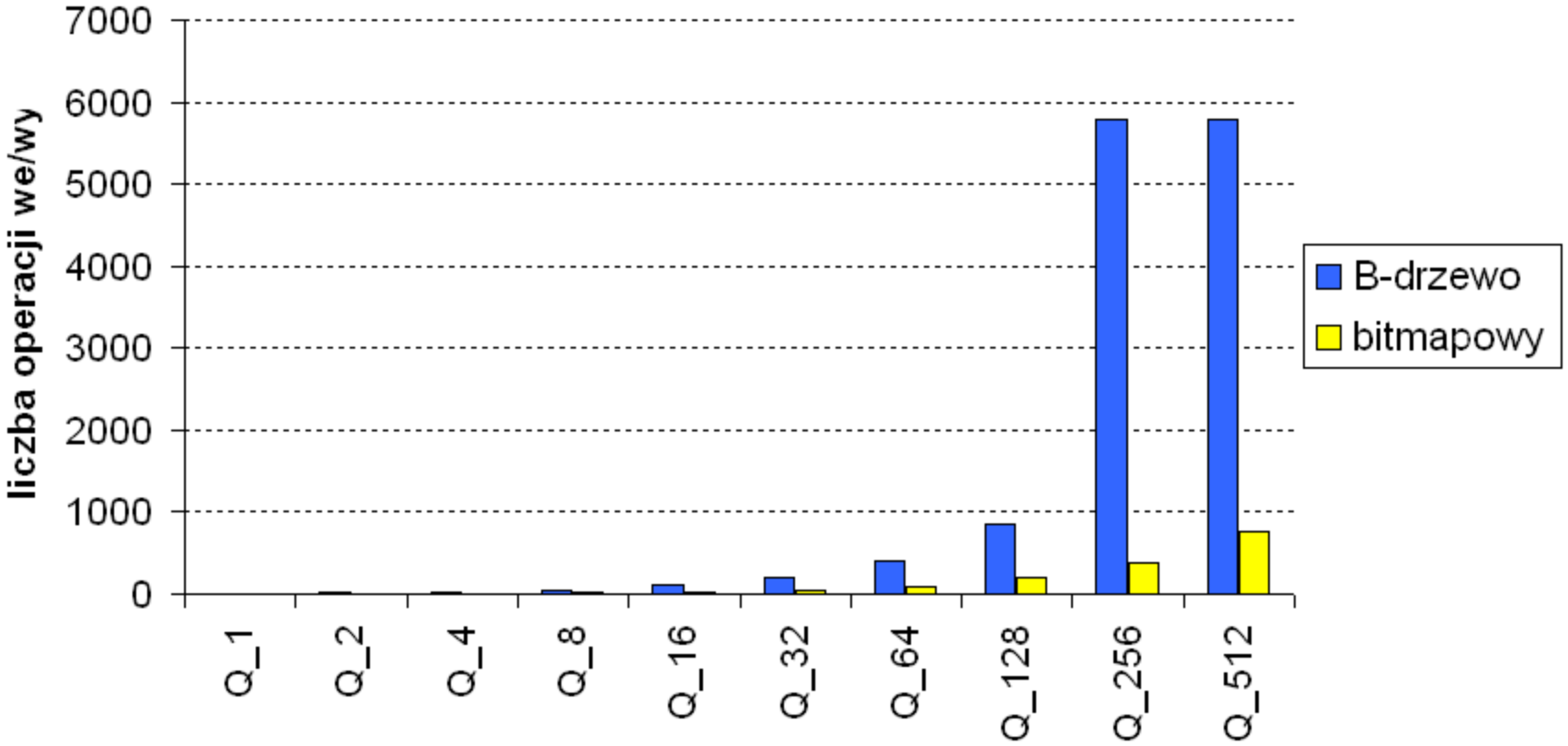
Eksperyment (1)

- Oracle10g R2
  - bufor danych: 73MB; SGA: 570MB
  - liczba rekordów w indeksowanej tabeli: 3,2 mln



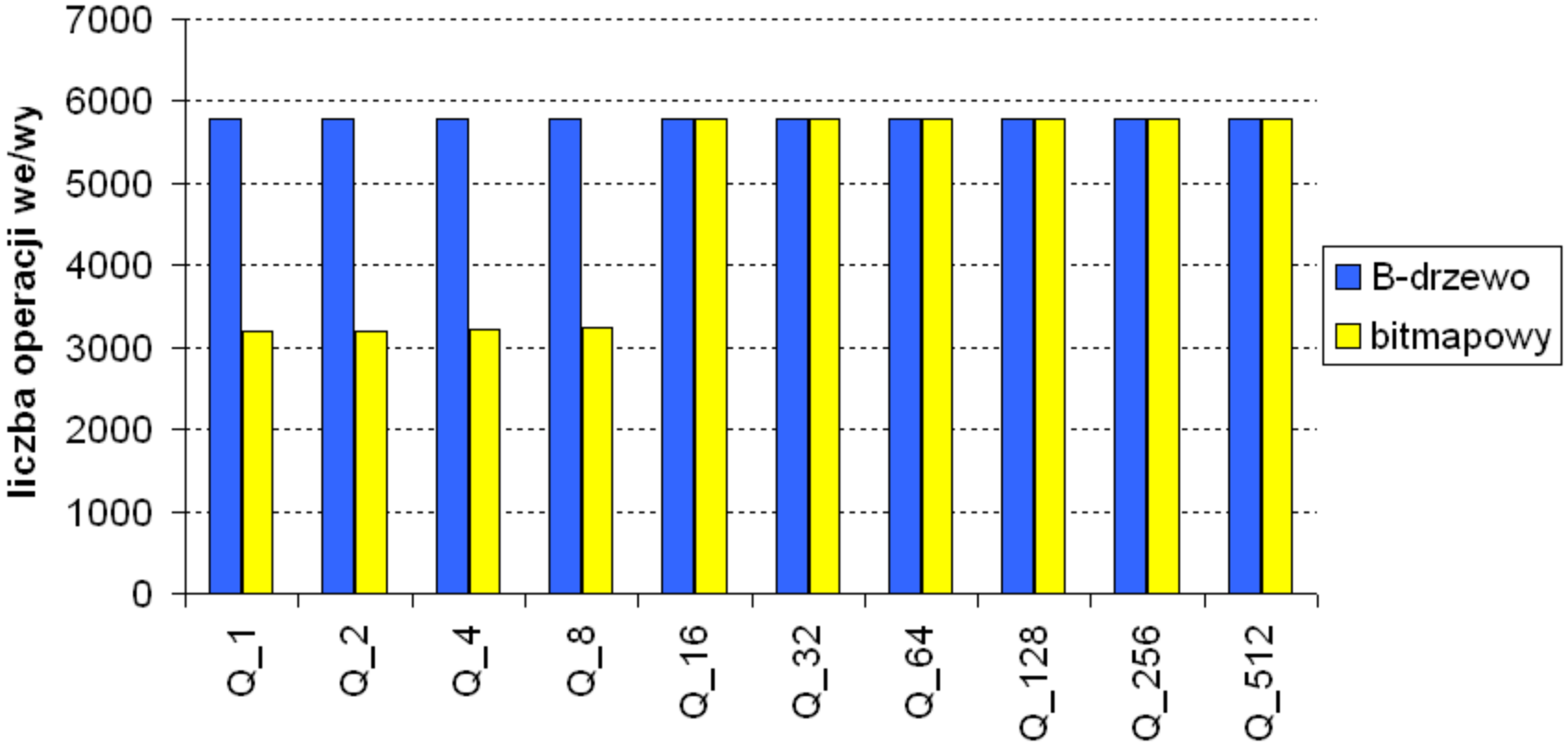
Eksperyment (2)

- `select count(1) from T where ...`
- **liczba rekordów w indeksowanej tabeli: 3,2 mln**
- **krotność indeksowanego atrybutu: 1024**



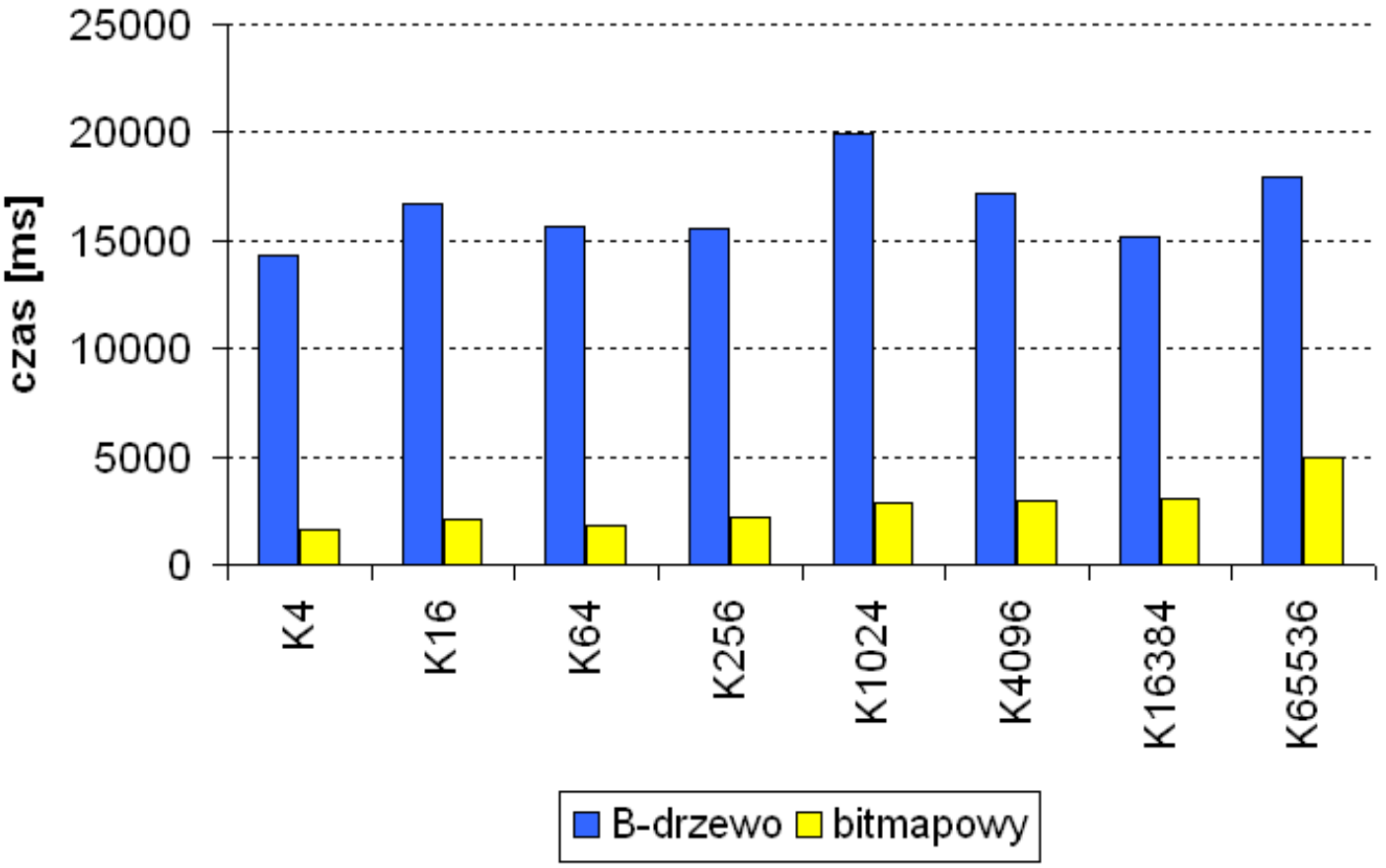
Eksperyment (3)

- `select sum(val) from T where ...`
- **liczba rekordów w indeksowanej tabeli: 3,2 mln**
- **krotność indeksowanego atrybutu: 1024**



Eksperyment (4)

- **Tworzenie indeksów**



Techniki zmniejszania rozmiaru IB

- Podział dziedziny indeksowanego atrybutu na zakresy
- Kodowanie map bitowych
- Kompresja map bitowych

Podział na zakresy (1)

- **Dziedzina indeksowanego atrybutu jest dzielona na zadane zakresy**
  - **np. temperatura <0, 20), <20, 40), <40, 60), <60, 80), <80, 100)**

indeksowany atrybut

temp
21
39,6
51,3
12
98,8
71
68,8
50,4
40

indeks bitmapowy z podziałem na zakresy

B4	B3	B2	B1	B0
(100, 80>	(80, 60>	(60, 40>	(40, 20>	(20, 0>
0	0	0	1	0
0	0	0	1	0
0	0	1	0	0
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	1	0	0

- **zapytanie: policz rekordy dla których 10<=temp<45**

Podział na zakresy (2)

- Możliwe budowanie map bitowych dla wybranych zbiorów wartości
  - np. {żółty, pomarańczowy, czerwony}, {błękitny, niebieski, granatowy}
- Cechy
  - uniezależnienie liczby map bitowych od szerokości dziedziny
  - w zbiorze wskazanym przez mapę bitową znajdują się również rekordy nie spełniające warunku selekcji ⇨ konieczność odfiltrowania

Kodowanie map bitowych (1)

- Zastąpienie wartości indeksowanego atrybutu inną wartością, której reprezentacja za pomocą map bitowych zajmie mniej miejsca
- Przykład
  - krotność indeksowanego atrybutu: 50000
  - podstawowy indeks bitmapowy: 50000 map bitowych
  - zakodowanie 50000 różnych wartości na $\lceil \log_2 50000 \rceil = 16$ bitów
  - konieczność odwzorowania wartości oryginalnych w zakodowane ⇨ tabela odwzorowania

Kodowanie map bitowych (2)

- zapytanie: select * from T where taryfa = 'Kubali'
- maska IB: ¬B2 ¬B1 B0

atrybut

taryfa
Kubali
Plus 20
Plus 40
Kubali
Relaks
Plus 40
Plus 60
Bonus Prestige

zakodowany IB

B2	B1	B0
0	0	1
0	1	0
0	1	1
0	0	1
1	0	1
0	1	1
1	0	0
1	1	0

tabela odwzorowania

wartość	kod
Kubali	001
Plus 20	010
Plus 40	011
Plus 60	100
Relaks	101
Bonus Prestige	110

Kompresja map bitowych

➲ Kodowanie run-length

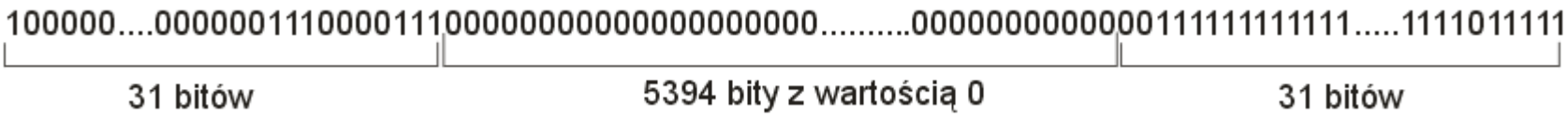
- **zastąpienie jednorodnego ciągu bitów o wartości w (0 lub 1) i o długości m ciągiem: [w m]**
- `0000000 1111111111 000 ⇨ 07 110 03`
- **kompresja BBC (Oracle)**
- **kompresja WAH**
- **kompresja RLH**
- **podział mapy bitowej na słowa o długości n bitów**
  - **BBC ⇨ słowa 8 bitowe**
  - **WAH ⇨ słowa 31 bitowe**
  - **RLH ⇨ słowa 1024 bitowe lub brak podziału**

Kompresja WAH i BBC

- WAH - Word Aligned Hybrid
- BBC - Byte-aligned Bitmap Compression
- Wypełnienie ⇨ słowo złożone z samych zer lub z samych jedynek
  - podlega kompresji
- Dopełnienie ⇨ słowo złożone z zer i jedynek
  - nie podlega kompresji
- Format skompresowanego słowa
  - bit pierwszy
    - wartość 1 oznacza wypełnienie
    - wartość 0 oznacza dopełnienie
  - bit drugi oznacza wartość wypełnienia (0 lub 1)
  - kolejne bity służą do zapisania długości wypełnienia, tj. liczby jednorodnych zer lub jedynek

Kompresja WAH (1)

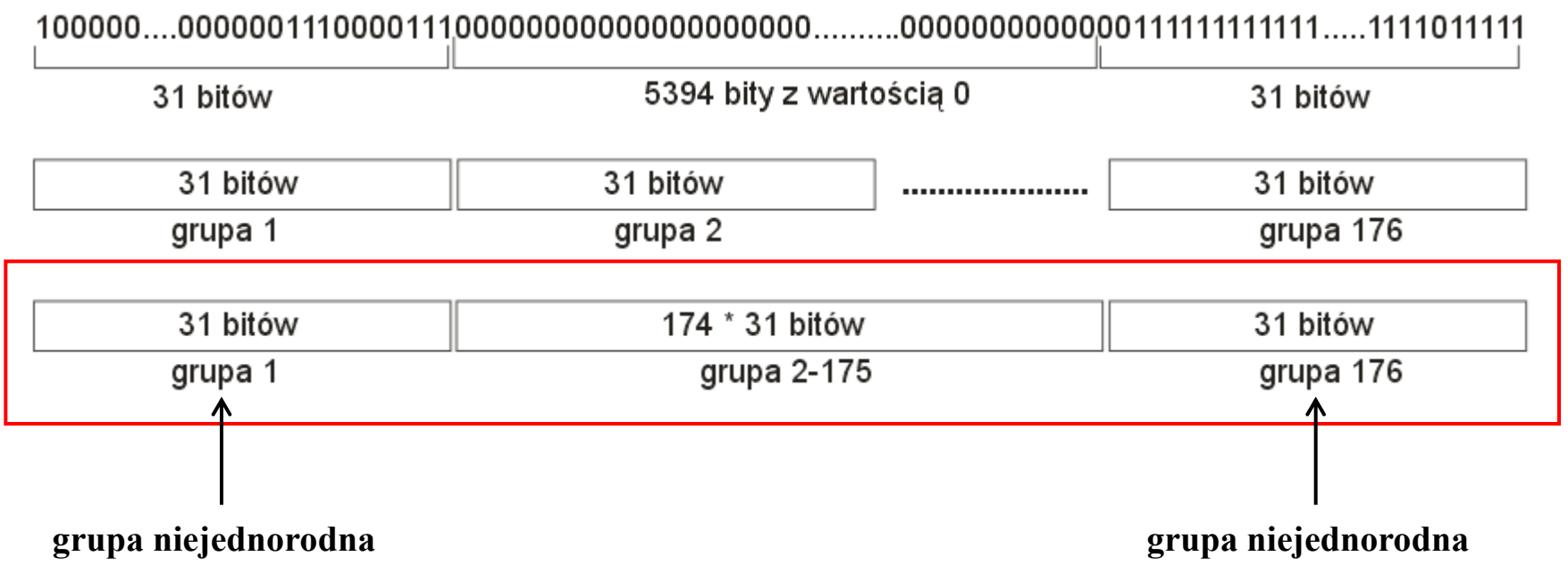
- Przykład
- Założenia
  - 32 bitowy procesor
  - mapa bitowa złożona z 5456 bitów



- Działanie kompresji WAH
  - krok 1: podział mapy bitowej na grupy złożone z 31 bitów

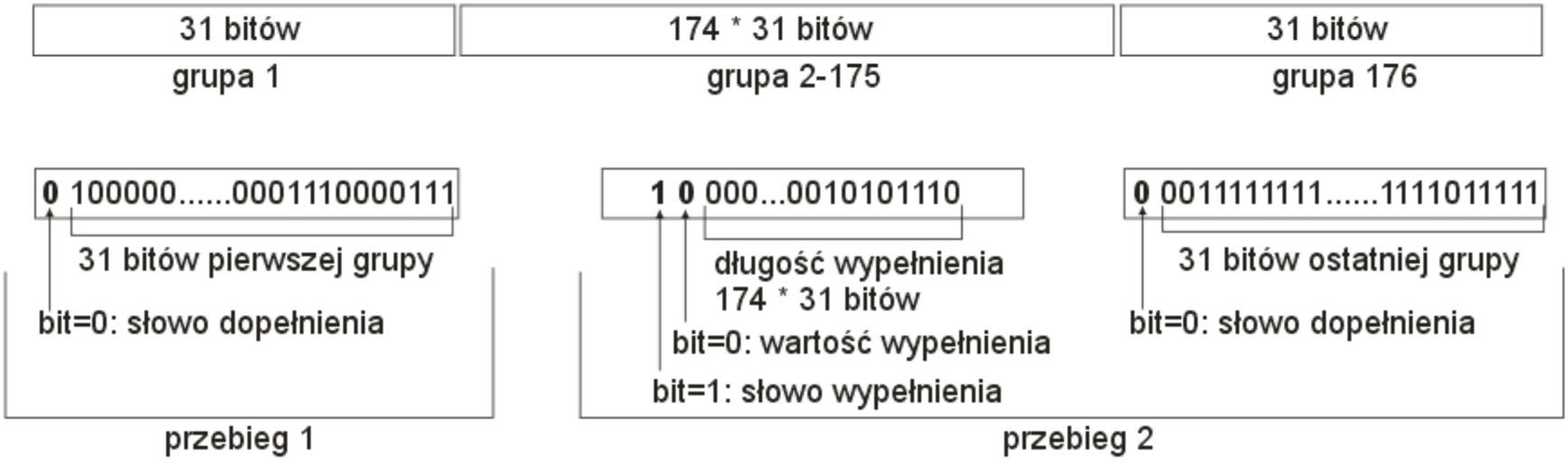
Kompresja WAH (2)

- **Działanie kompresji WAH**
  - **krok 2: scalenie grup z identycznymi bitami w jedną grupę**
    - **scalane są grupy 2 do 175**



Kompresja WAH (3)

- **Działanie kompresji WAH**
  - **kodowanie 3 grup wynikowych**
    - **grupa pierwsza ⇨ dopełnienie przebiegu 1**
    - **grupa 2-175 ⇨ wypełnienie przebiegu 2**
    - **grupa 176 ⇨ dopełnienie przebiegu 2**



Kompresja WAH i BBC - wady

- Podział mapy bitowej na słowa pogarsza jakość jej kompresji ⇨ jednorodne ciągi bitów nie zawsze pasują do długości słowa i w takim przypadku nie mogą być kompresowane
- Ze wzrostem krotności indeksowanego atrybutu maleje liczba jednorodnych ciągów bitów (przy nieuporządkowaniu danych)

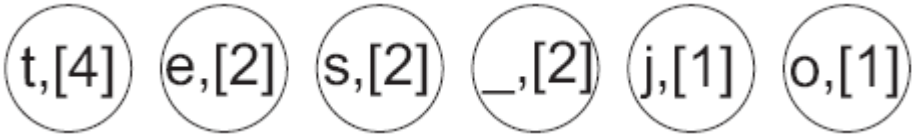
Kompresja RLH

- RLH - Run-Length Huffman
- Bazuje na
  - kodowaniu run-length (zmodyfikowane)
  - kodowaniu Huffmana

Kodowanie Huffmana (1)

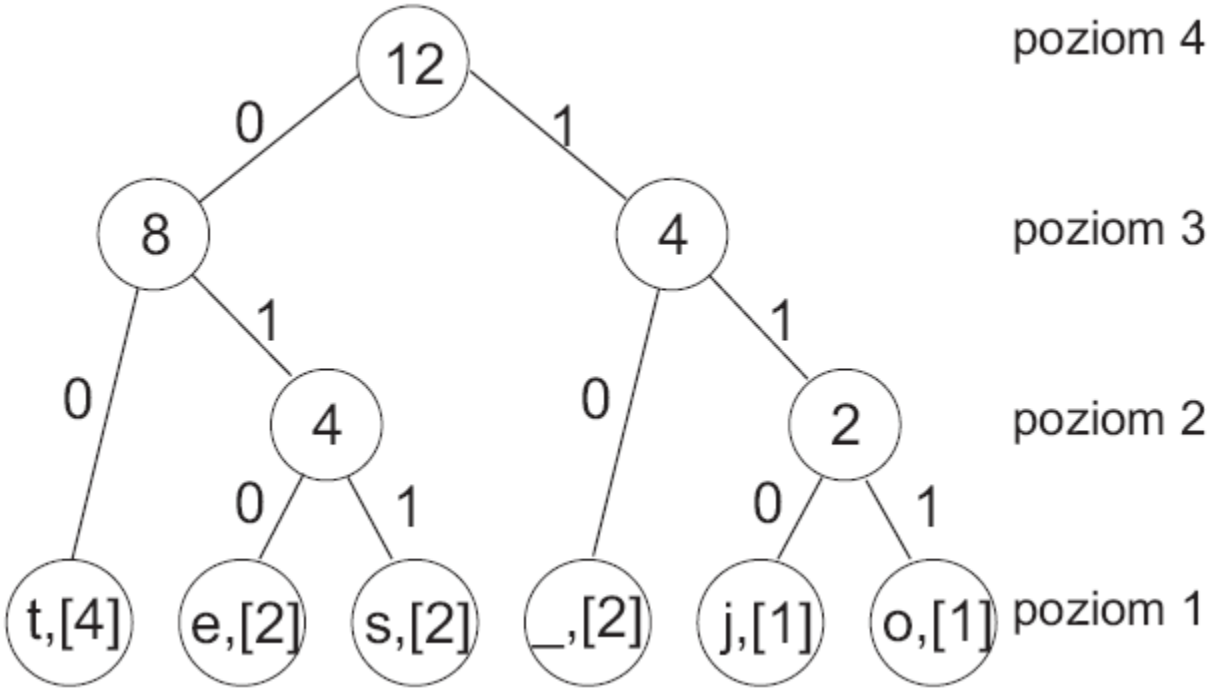
- Koncepcja: oryginalne symbole są zastępowane ciągami bitów, przy czym symbole występujące częściej są zastępowane krótszymi ciągami bitów
- Przykład
  - zakodować ciąg znaków: `to_jest_test`
- Działanie kodowania Huffmana
  - Krok 1: obliczenie częstości występowania symboli w kodowanym ciągu

symbol	t	e	s	_	j	o
częstość	4	2	2	2	1	1



Kodowanie Huffmana (2)

- **Działanie kodowania Huffmana**
  - **Krok 2: zbudowanie drzewa kodów Huffmana**

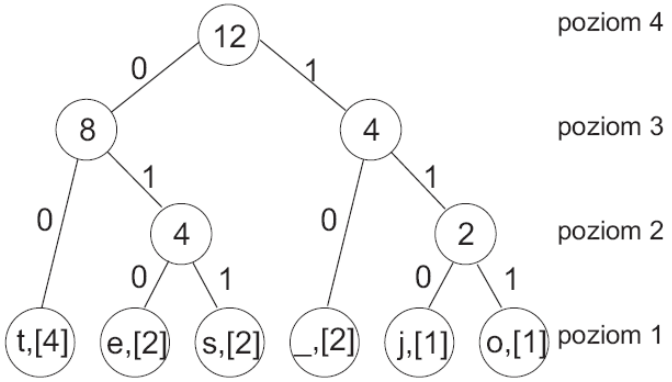


Kodowanie Huffmana (3)

- **Działanie kodowania Huffmana**
  - **Krok 3: wyznaczenie kodów Huffmana i zastąpienie nimi oryginalnych symboli**

kodowany symbol	t	e	s	_	j	o
kod Huffmana	00	010	011	10	110	111

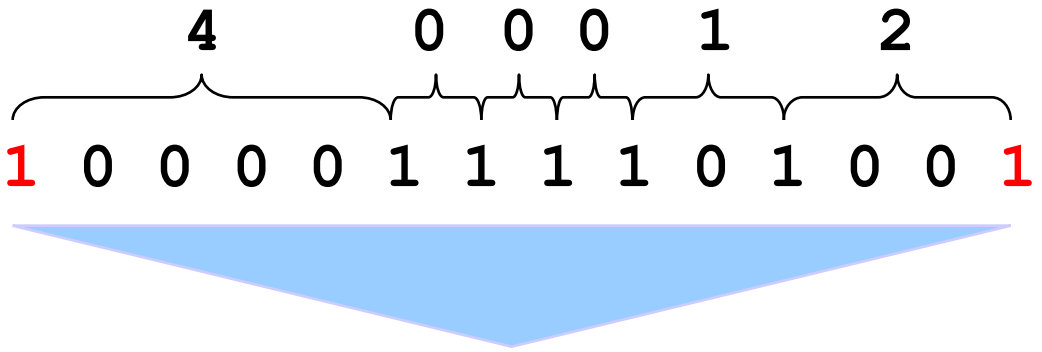
t	o	_	j	e	s	t	_	t	e	s	t
00	111	10	110	010	011	00	10	00	010	011	00



- **oryginalny tekst: 12B**
- **skompresowany tekst: 4B**

Kompresja RLH (1)

- Zmodyfikowane kodowanie run-length
  - kodowaniu podlegają odległości pomiędzy kolejnymi bitami o wartości jeden



400012

- Wzrost krotności atrybutu ⇨ zmniejszanie się gęstości map bitowych ⇨ zmniejszanie się liczby symboli wykorzystywanych do kodowania map bitowych

Kompresja RLH (2)

- Mapy bitowe, zakodowane z wykorzystaniem zmodyfikowanego kodowania run-length są kompresowane z wykorzystaniem kodowania Huffmana
- Częstości odległości we wszystkich mapach bitowych są reprezentowane w drzewie kodów Huffmana
- Rozmiar drzewa kodów Huffmana jest niewielki ⇨ przechowywane w RAM, zwiększając efektywność kompresowania i dekompresowania map bitowych

Kompresja RLH (3)

Klienci

ID	płeć
1	mężczyzna
2	kobieta
3	kobieta
4	kobieta
5	mężczyzna
6	mężczyzna
7	mężczyzna
8	kobieta
9	kobieta
10	mężczyzna
11	mężczyzna
12	mężczyzna
13	kobieta
14	kobieta
15	kobieta
16	mężczyzna
17	kobieta
18	kobieta
19	kobieta

indeks bitmapowy

kobieta	mężczyzna
0	1
1	0
1	0
1	0
0	1
0	1
0	1
1	0
1	0
0	1
0	1
0	1
1	0
1	0
1	0
0	1
1	0
1	0
1	0

➲ Przykładowe działanie

- **Krok 1:**
  - **kodowanie map bitowych**

kobieta	mężczyzna
1	0
0	3
0	0
3	0
0	2
3	0
0	0
0	3
1	3
0	
0	

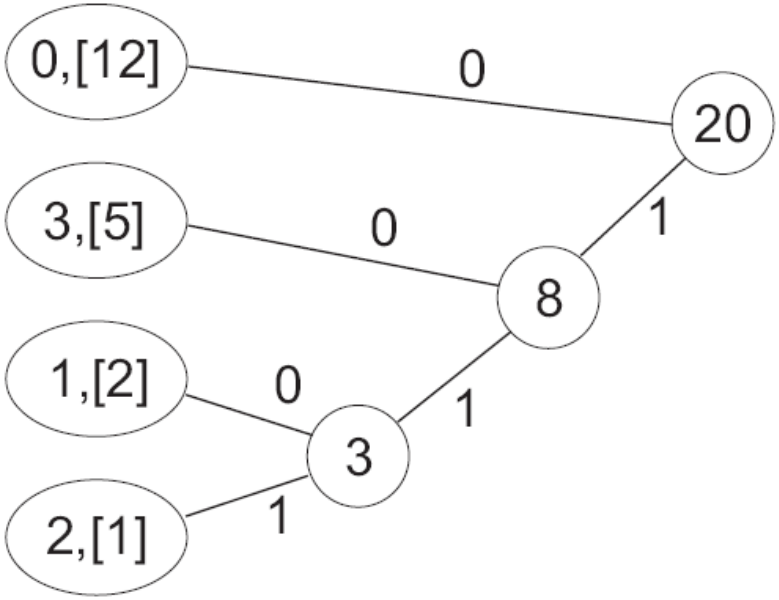
  - **obliczenie częstości symboli**

symbol	częstość
0	12
3	5
1	2
2	1

Kompresja RLH (4)

- **Przykładowe działanie**
  - **Krok 2: zbudowanie drzewa kodów Huffmana**

symbol	częstość
0	12
3	5
1	2
2	1



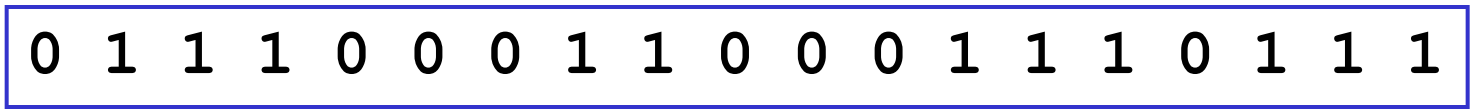
symbol	kod symbolu
0	0
3	10
1	110
2	111

Kompresja RLH (5)

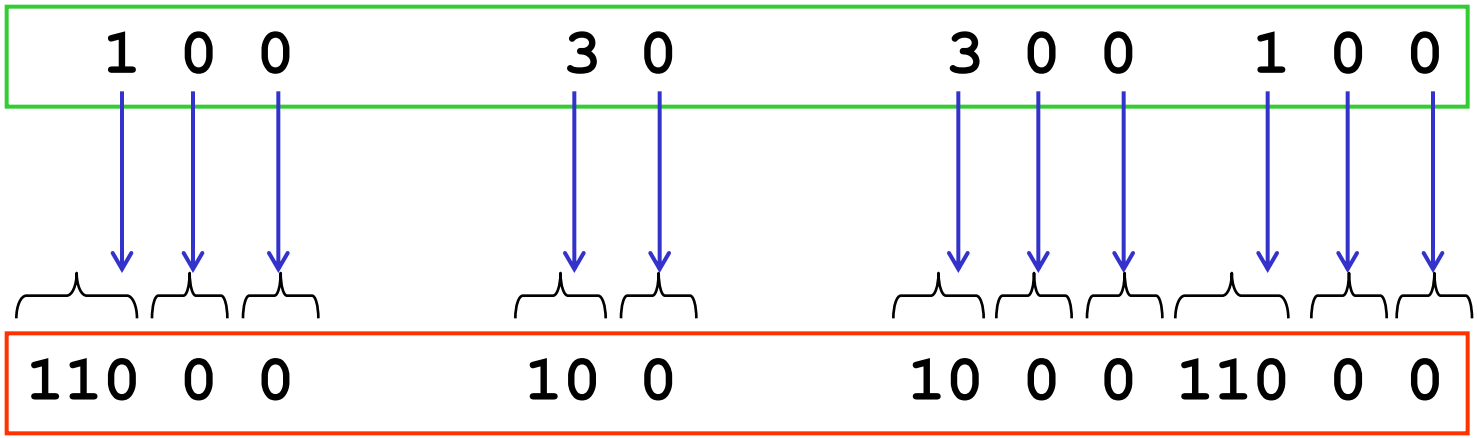
➲ Przykładowe działanie

- **Krok 3: kompresja**

oryginalna mapa bitowa płeć='kobieta'



reprezentacja mapy po zmodyfikowanym kodowaniu run-length

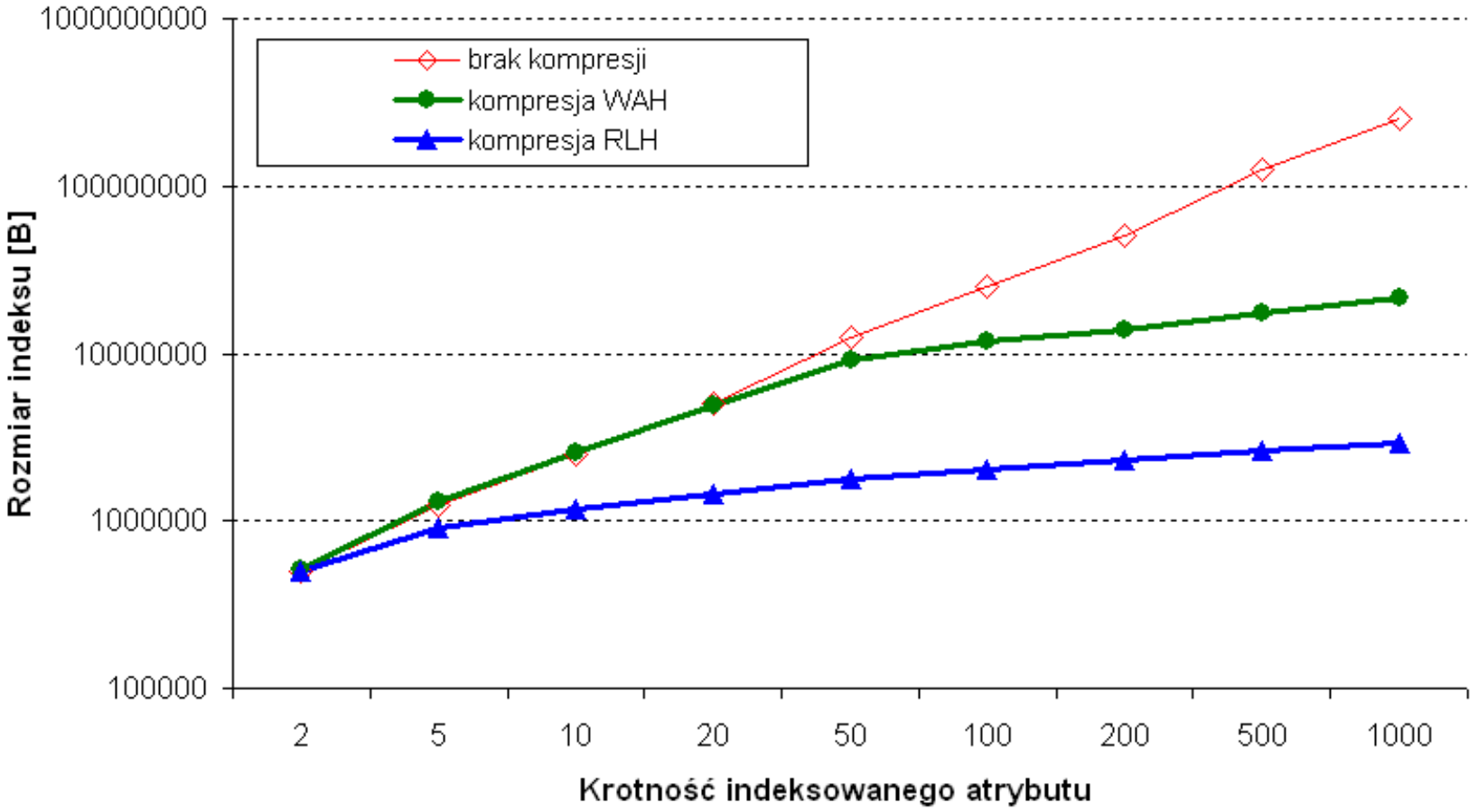


symbol	kod symbolu
0	0
3	10
1	110
2	111

skompresowana mapa bitowa RLH

WAH i RLH: rozmiary

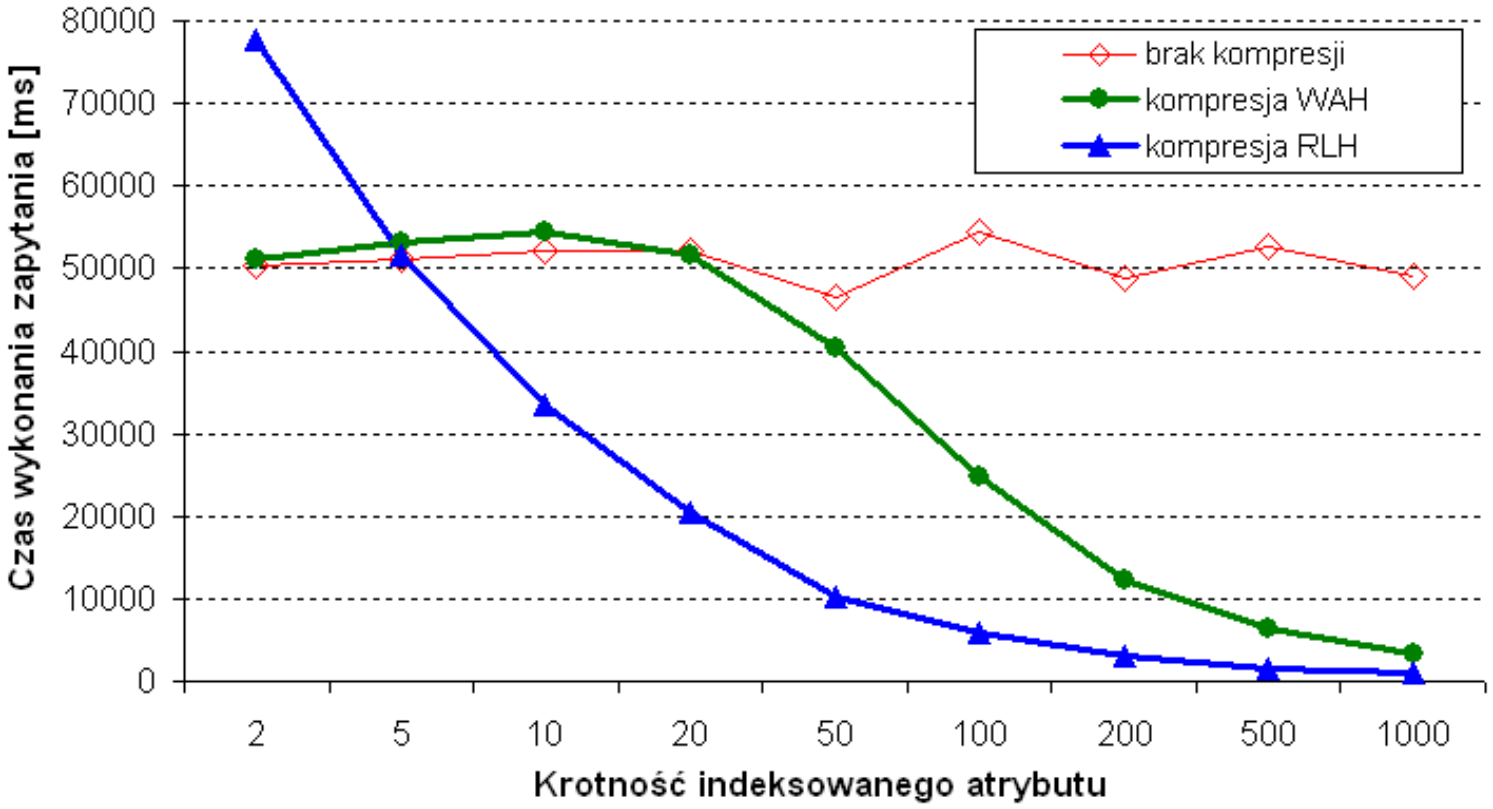
- **Liczba rekordów indeksowanej tabeli: 2 mln**



WAH i RLH: czasy odpowiedzi

- **Zapytanie:**

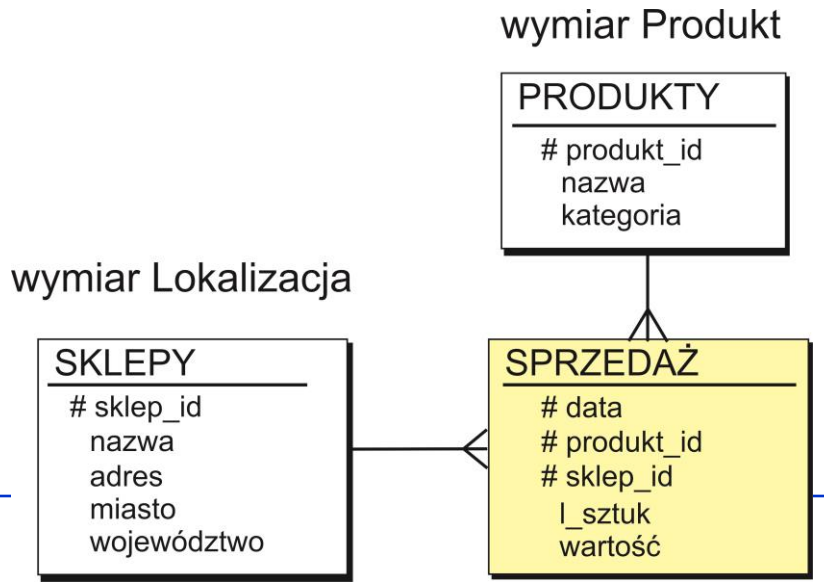
```
select ... from ...
where indeksowany_atrybut in (w1, w2, ..., w100)
```



Indeksy bitmapowe w systemach komercyjnych (1)

➲ Oracle

- **definiowane jawnie przez administratora**
- **kompresowane automatycznie (BBC)**
- **wykorzystywane do optymalizacji zapytań gwiaździstych ⇨ HD_wykl04_indeks-STAR-KJ**
- **bitmapowy indeks połączeniowy**



Indeksy bitmapowe w systemach komercyjnych (2)

➲ Oracle - bitmapowy indeks połączeniowy

Produkty

produkt_id	.....	kategoria
100		kosmetyki
200		alkohole
300		kosmetyki
400		kosmetyki

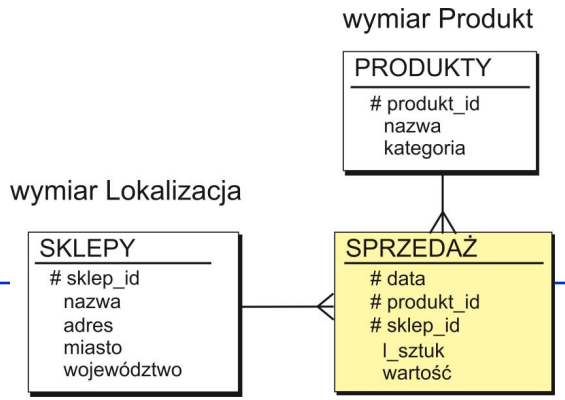
Sprzedaż

ROWID	.....	produkt_id	.....
x000A		100	
x000B		100	
x000C		300	
x000D		200	
x000E		200	
x000F		400	

bitmapowy indeks połączeniowy sprzedaz(produkty.kategoria)

kosmetyki	alkohole	ROWID_Sprzedaz
1	0	x000A
1	0	x000B
1	0	x000C
0	1	x000D
0	1	x000E
1	0	x000F

```
create bitmap index sprz_jbi
on sprzedaz(produkty.kategoria)
from sprzedaz, produkty
where sprzedaz.produkt_id=
                produkty.produkt_id;
```

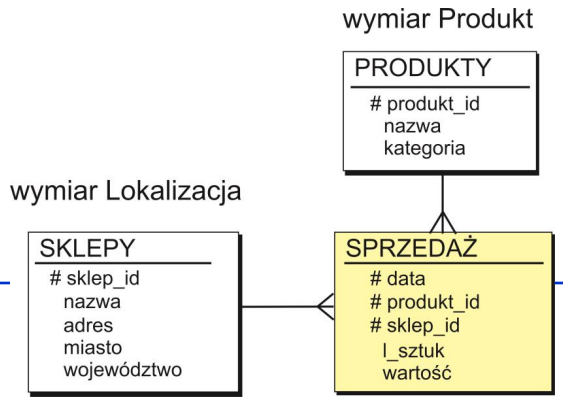


Indeksy bitmapowe w systemach komercyjnych (3)

- **Oracle - optymalizacja zapytań gwiaździstych z wykorzystaniem bitmapowego indeksu połączeniowego**

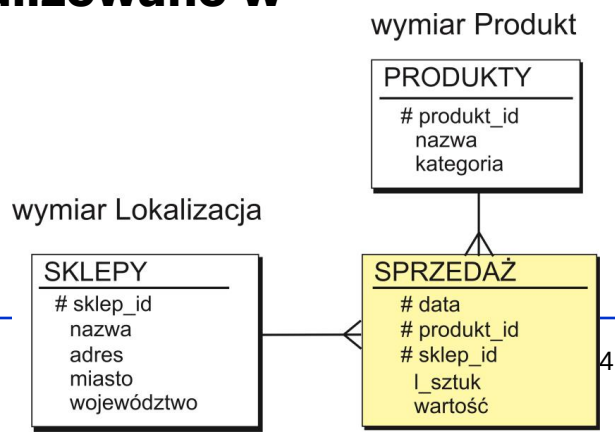
```
select sum(wartosc), pr.nazwa, sk.nazwa
from sprzedaz sp, sklepy sk, produkty pr
where sk.wojewodztwo in ('Mazowieckie', 'Wielkopolskie')
and pr.kategoria='kosmetyki'
and sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
group by pr.nazwa, sk.nazwa;
```

- **Zdefiniowano bitmapowe indeksy połączeniowe na atrybutach**
  - `Sklepy.wojewodztwo`
  - `Produkty.kategoria`



Indeksy bitmapowe w systemach komercyjnych (4)

- **Indeks na `Produkty.kategoria`**
  - **będzie zawierał tyle map bitowych ile jest różnych kategorii produktów (w tabeli Produkty)**
  - **pojedyncza mapa tego indeksu, np. dla grupy kosmetyki będzie wskazywała na te rekordy w tabeli sprzedaż, które opisują sprzedaż kosmetyków**
- **Indeks na `Sklepy.wojewodztwo`**
  - **będzie zawierał tyle map bitowych ile jest różnych województw w tabeli sklepy**
  - **pojedyncza mapa tego indeksu, np. dla Wielkopolski będzie opisywała te sprzedaże, które zrealizowano w Wielkopolsce**



Indeksy bitmapowe w systemach komercyjnych (5)

- Nastąpi przepisanie oryginalnego zapytania do zapytania z wykorzystaniem połączeniowych indeksów bitmapowych
- Krok 1
  - odczytanie mapy opisującej sprzedaż produktów kategorii kosmetyki ($MB^{kosmetyki}$)
  - odczytanie mapy opisującej sprzedaż w województwie wielkopolskim ($MB^{Wielkopolska}$)
  - odczytanie mapy opisującej sprzedaż w województwie mazowieckim ($MB^{Mazowsze}$)
- Krok2
  - wyznaczenie wynikowej mapy (MB^{wynik})

$$MB^{wynik} = MB^{kosmetyki} \text{ and } (MB^{Wielkopolska} \text{ or } MB^{Mazowsze})$$

- Krok3
  - odczytanie rekordów z tabeli z wykorzystaniem mapy MB^{wynik}