




# Hurtownie danych - przegląd technologii

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




R.Wrembel - Politechnika Poznańska, Instytut Informatyki



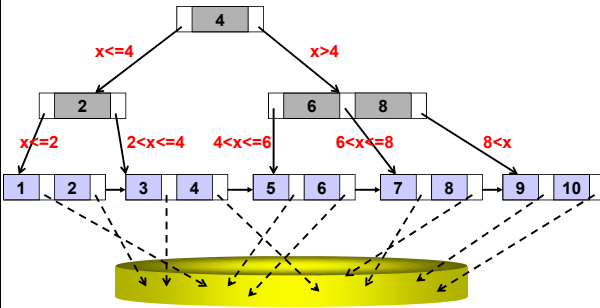
# Efektywność przetwarzania OLAP

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


R.Wrembel - Politechnika Poznańska, Instytut Informatyki



# B-drzewo (B-tree)



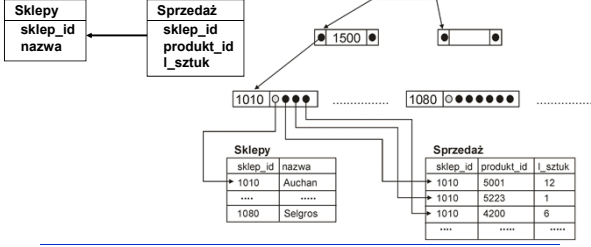
R.Wrembel - Politechnika Poznańska, Instytut Informatyki




# Indeks połączeniowy (join index)

⇒ Zmaterializowany wynik połączenia

np. indeks na atrybucie Sklepy.sklep\_id




R.Wrembel - Politechnika Poznańska, Instytut Informatyki



# 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**

R.Wrembel - Politechnika Poznańska, Instytut Informatyki



# Indeks bitmapowy - definicje

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

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

R.Wrembel - Politechnika Poznańska, Instytut Informatyki



## Indeks bitmapowy - wykorzystanie

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

Diagram illustrating the bit map construction process:

Audi OR Ford = M<sub>1</sub> AND sport = M<sub>2</sub> AND K = M<sub>3</sub>

Audi	Ford	M <sub>1</sub>	sport	M <sub>2</sub>	K	M <sub>3</sub>
0	1	1	0	0	0	0
1	1	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	0	1	1	1
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	0	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

- I\_rekordów = 1 000 000
- krotność<sub>A</sub> = 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<sup>+</sup>-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

- I\_rekordów = 1 000 000
- krotność<sub>A</sub> = 1024
- adres\_rekordu = 10B
- indeks bitmapowy na atrybucie A
  - 1024 mapy bitowe: 1024 x (1 000 000 / 8) = 1024 x 125kB = 128MB
- indeks B<sup>+</sup>-drzewo na atrybucie A
  - 1 000 000 x 10B = 10MB



## Indeks bitmapowy – charakterystyka (5)

⇒ Koszty uaktualniania indeksów w trakcie pracy systemu

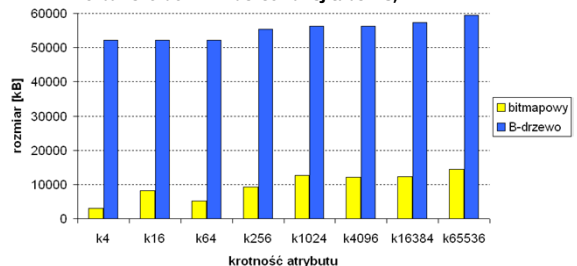
- wstawianie rekordów – zwiększanie długości map bitowych
- usuwanie rekordów – zmniejszanie długości map bitowych
- modyfikowanie rekordów – operacje na 2 mapach
- współbieżność – blokowane są ciągłe obszary map bitowych



## Eksperyment (1)

⇒ Oracle10g R2

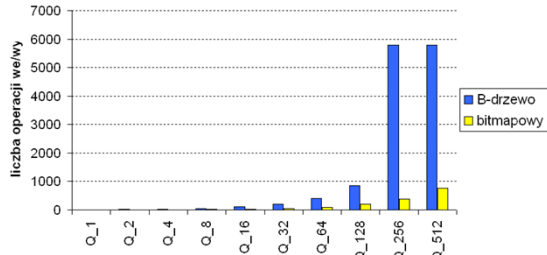
- bufor danych: 73MB; SGA: 570MB
- liczba rekordów w indeksowanej tabeli: 3,2 mln





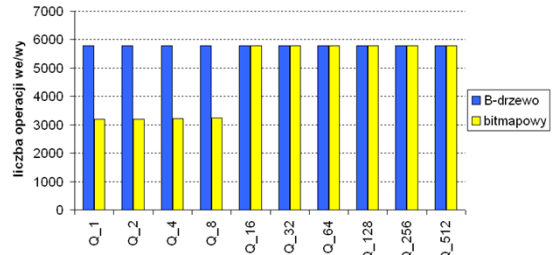
## Ekspertyment (2)

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



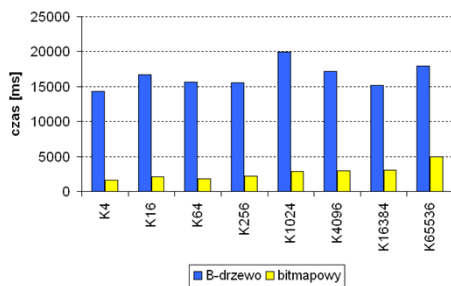
## Ekspertyment (3)

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



## Ekspertyment (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	B4	B3	B2	B1	B0
21	0	0	0	1	0
39,6	0	0	0	1	0
51,3	0	0	1	0	0
12	0	0	0	0	1
98,8	1	0	0	0	0
71	0	1	0	0	0
68,8	0	1	0	0	0
50,4	0	0	1	0	0
40	0	0	1	0	0

- ↻ zapytanie: policz rekordy dla których  $10 \leq temp < 45$



## Podział na zakresy (2)

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







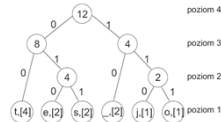
## Kodowanie Huffmana (3)

### ⇒ 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



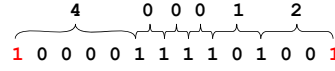
- oryginalny tekst: 12B
- skompresowany tekst: 4B



## 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)

### ⇒ Przykładowe działanie

- Krok 1: kodowanie map bitowych

ID	płeć	indeks bitmapowy	
		kobieta	mężczyzna
1	mężczyzna	0	1
2	kobieta	1	0
3	kobieta	1	0
4	kobieta	1	0
5	mężczyzna	0	1
6	mężczyzna	0	1
7	mężczyzna	0	1
8	kobieta	1	0
9	kobieta	1	0
10	mężczyzna	0	1
11	mężczyzna	0	1
12	mężczyzna	0	1
13	kobieta	1	0
14	kobieta	1	0
15	kobieta	1	0
16	mężczyzna	0	1
17	kobieta	1	0
18	kobieta	1	0
19	kobieta	1	0

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

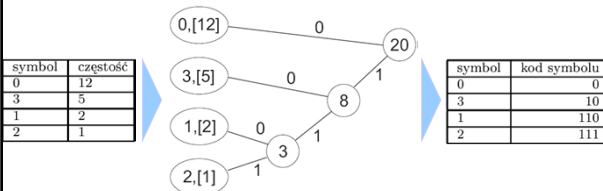
symbol	częstość
0	12
3	5
1	2
2	1



## Kompresja RLH (4)

### ⇒ Przykładowe działanie

- Krok 2: zbudowanie drzewa kodów Huffmana



## Kompresja RLH (5)

### ⇒ Przykładowe działanie

- Krok 3: kompresja

oryginalna mapa bitowa płeć="kobieta"

0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1

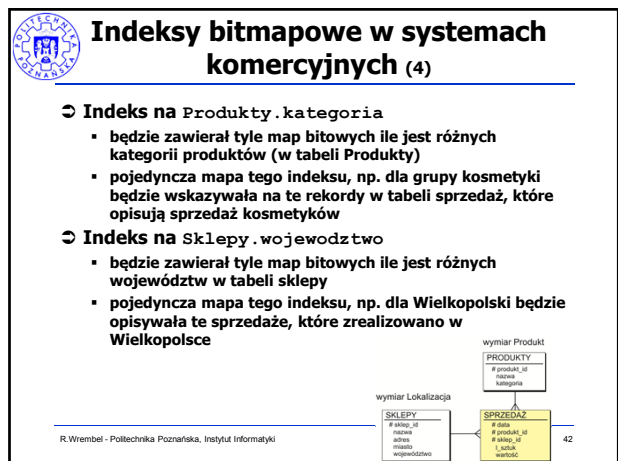
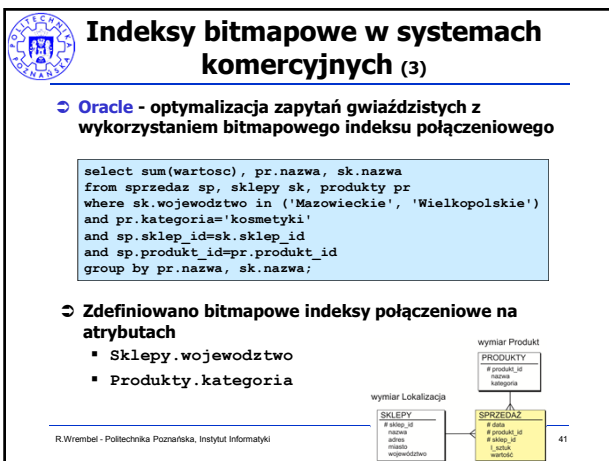
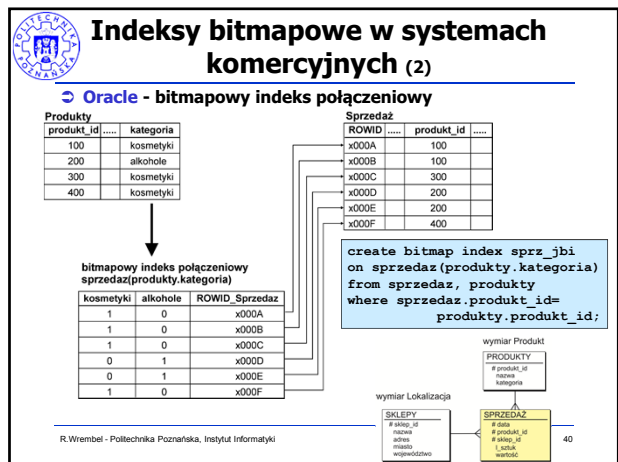
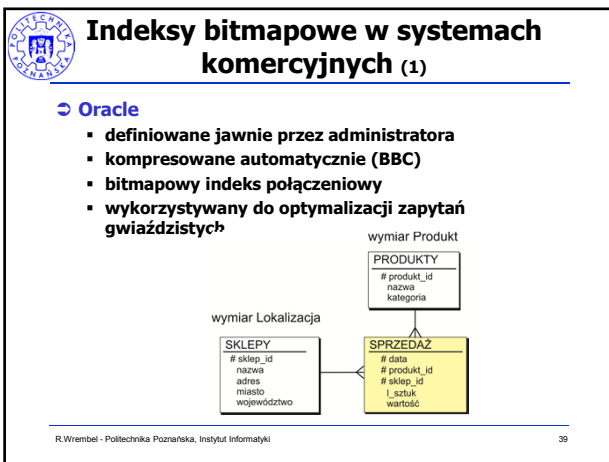
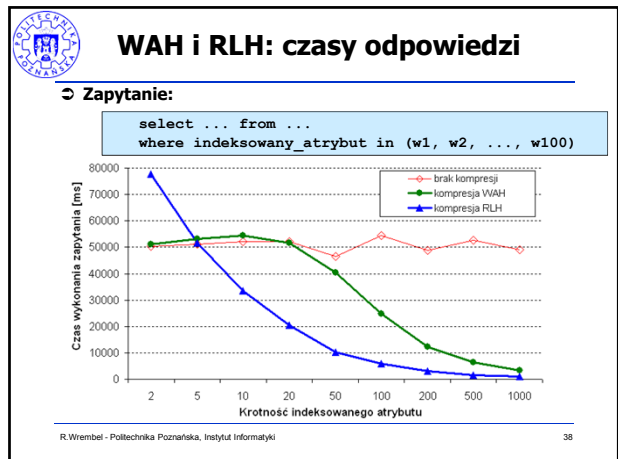
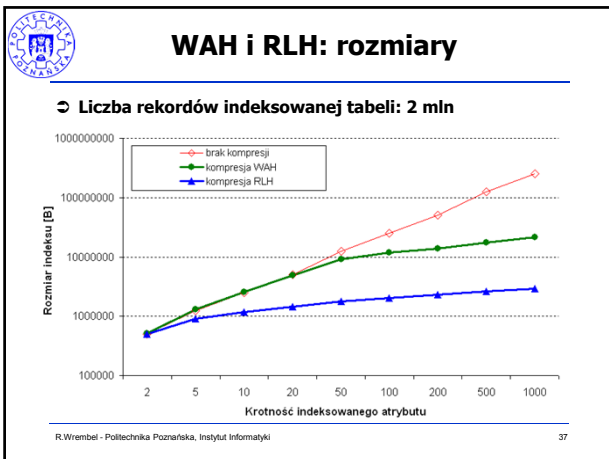
reprezentacja mapy po zmodyfikowanym kodowaniu run-length

1 0 0 3 0 3 0 0 1 0 0

110 0 0 10 0 10 0 0 110 0 0

symbol	kod symbolu
0	0
3	10
1	110
2	111

skompresowana mapa bitowa RLH





## 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  $MB^{wynik}$



## Indeksy bitmapowe w systemach komercyjnych (6)

### IBM DB2

- tworzone i zarządzane niejawnie przez system
- wykorzystywane do optymalizacji zapytań gwiazdzystych

⇒ Każda z tabel wymiaru jest niezależnie łączona operatorem pół-połączenia z tabelą faktów

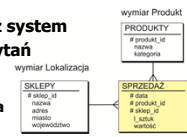
⇒ Do łączenia są wykorzystane indeksy B-drzewo na kluczach obcych tabeli faktów

⇒ Fizyczne adresy rekordów (ROWID) będących wynikiem każdego pół-połączenia są reprezentowane przez oddzielną mapę bitową

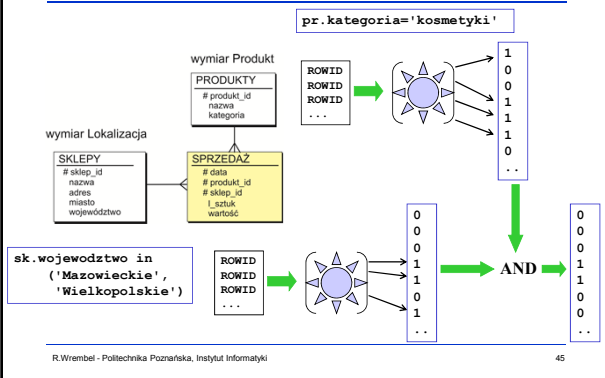
⇒ Mapy bitowe  $B_i$  są konstruowane z wykorzystaniem funkcji haszowej na ROWID

- wartość funkcji haszowej odpowiada określonemu bitowi mapy

⇒ Mapa wynikowa (opisująca rekordy będące wynikiem całego zapytania) jest iloczynem logicznym map  $B_i$



## Indeksy bitmapowe w systemach komercyjnych (7)



## Indeksy bitmapowe w systemach komercyjnych (8)

### SQL Server

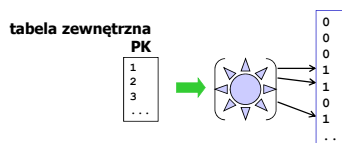
- tworzone i zarządzane niejawnie przez system
- wykorzystywane do optymalizacji połączeń tabeli wymiaru z tabelą faktów algorytmem *hash join*
- tabela z kluczem podstawowym (tab. wymiaru) ⇒ tabela zewnętrzna
- tabela z kluczem obcym (tab. faktów) ⇒ tabela wewnętrzna



## Indeksy bitmapowe w systemach komercyjnych (9)

⇒ Wartości klucza podstawowego tabeli zewnętrznej są odwzorowywane w mapę bitową za pomocą funkcji haszowej

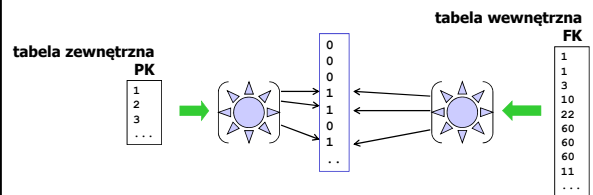
- f.haszowa(PK) → nr bitu
- ⇒ Wartość klucza podstawowego po przetworzeniu funkcją haszową powoduje nadanie odpowiedniemu bitowi wartości 1



## Indeksy bitmapowe w systemach komercyjnych (10)

⇒ Haszuje się wartości klucza obcego tabeli wewnętrznej za pomocą tej samej funkcji haszowej

- jeżeli rekord  $R_i$  z tabeli wewnętrznej haszuje się do bitu z wartością 1, wówczas  $R_i$  łączy się z rekordem z tabeli zewnętrznej





## Indeksy bitmapowe w systemach komercyjnych (11)

### ⇒ Sybase IQ

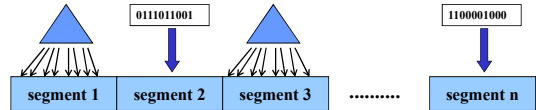
- definiowane jawnie przez administratora
- Low Fast ⇒ dla atrybutów o wąskiej dziedzinie
  - max krotność 10 000 wartości
  - największa efektywność dla krotności do 1000
- High Non Group ⇒ dla atrybutów o szerokiej dziedzinie
  - optymalizacja zapytań z warunkami zakresu i wyciszających agregaty



## Indeksy bitmapowe w systemach komercyjnych (12)

### ⇒ SAS Scalable Performance Data (SPD) Server

- indeks hybrydowy
- podział tabeli na segmenty (np. 8192 rekordy), z których każdy jest indeksowany niezależnie
  - indeksem bitmapowym lub
  - indeksem B-drzewo
- rodzaj indeksu wybierany automatycznie przez system na podstawie
  - gęstości wartości indeksowanego atrybutu
  - dystrybucji wartości



## Podsumowanie

### ⇒ Indeks bitmapowy

- efektywny dla atrybutów o wąskich dziedzinach
- mniej efektywny dla atrybutów o szerokich dziedzinach ⇒ rozmiar indeksu
  - podział dziedziny
  - kodowanie map bitowych
  - kompresja
    - kodowanie run length
    - algortymy BBC, WAH, RLH
- Oracle, IBM DB2, Sybase IQ, MS SQL Server, SAS SPD Server