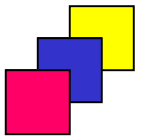
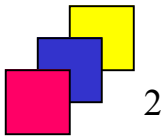
Optymalizacja poleceń SQL

Indeksy



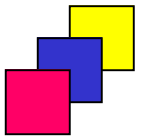
Indeksy

- **Dodatkowe struktury służące przyspieszaniu dostępu do danych.**
- **Tworzone dla relacji, są jednak niezależne logicznie i fizycznie od danych relacji.**
- **O użyciu indeksu przy realizacji operacji decyduje SZBD.**
- **Są automatycznie pielęgnowane przez bazę danych.**
- **Zalety:**
  - **przyspieszają odczyt danych (nie zawsze!),**
  - **wpływają na zwiększenie stopnia współbieżności wykonywanych w bazie danych operacji.**
- **Wady:**
  - **mogą znacznie spowolnić operacje modyfikacji danych,**
  - **zajmują przestrzeń dyskową.**



Struktura indeksu

- **Indeks składa się z rekordów.**
- **Rekord indeksu złożony jest z dwóch pól:**
  - **klucz – zawiera wartości występujące w atrybutach relacji, na których założono indeks, tzw. atrybutach indeksowych, lub wartości wyrażeń, zbudowanych z atrybutów relacji,**
  - **wskaźnik – określa blok zawierający rekordy, których wartości atrybutów indeksowych są równe wartościom klucza. W SZBD Oracle wskaźnik jest implementowany w postaci adresu rekordu (ang. rowid).**



Adres rekordu

- **Określa dokładną fizyczną lokalizację rekordu w bazie danych.**
- **Struktura: OOOOOOFFFBBBBBBRRR:**
  - **OOOOOO – numer identyfikacyjny obiektu bazy danych (np. relacji), w której znajduje się rekord,**
  - **FFF – numer pliku bazy danych,**
  - **BBBBBB – numer bloku w pliku,**
  - **RRR – numer rekordu w bloku.**
- **Odczyt adresu rekordu:**

```
SELECT nazwisko, etat, rowid FROM pracownicy;
```

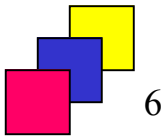
- **Adres rekordu jest <u>niezmienny</u>.**

Użycie indeksu

1. **Użytkownik wykonuje polecenie z warunkiem zawierającym poindeksowany atrybut.**
2. **SZBD szuka w indeksie klucza (zbioru kluczy), którego wartość (wartości) odpowiada wartości poindeksowanego atrybutu w warunku polecenia.**
3. **SZBD odczytuje adres rekordu (zbiór adresów rekordów) ze znalezionego w kroku 2. klucza (kluczy).**
4. **SZBD odczytuje rekord (zbiór rekordów), którego adresy (adresy) odczytał w kroku 3.**

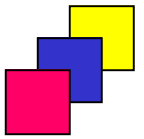
Jakie atrybuty indeksować?

- **Atrybuty często używane w klauzulach WHERE zapytań.**
- **Atrybuty często używane w warunkach połączeniowych.**
- **Atrybuty rzadko modyfikowane.**
- **Atrybuty będące kluczami obcymi relacji.**



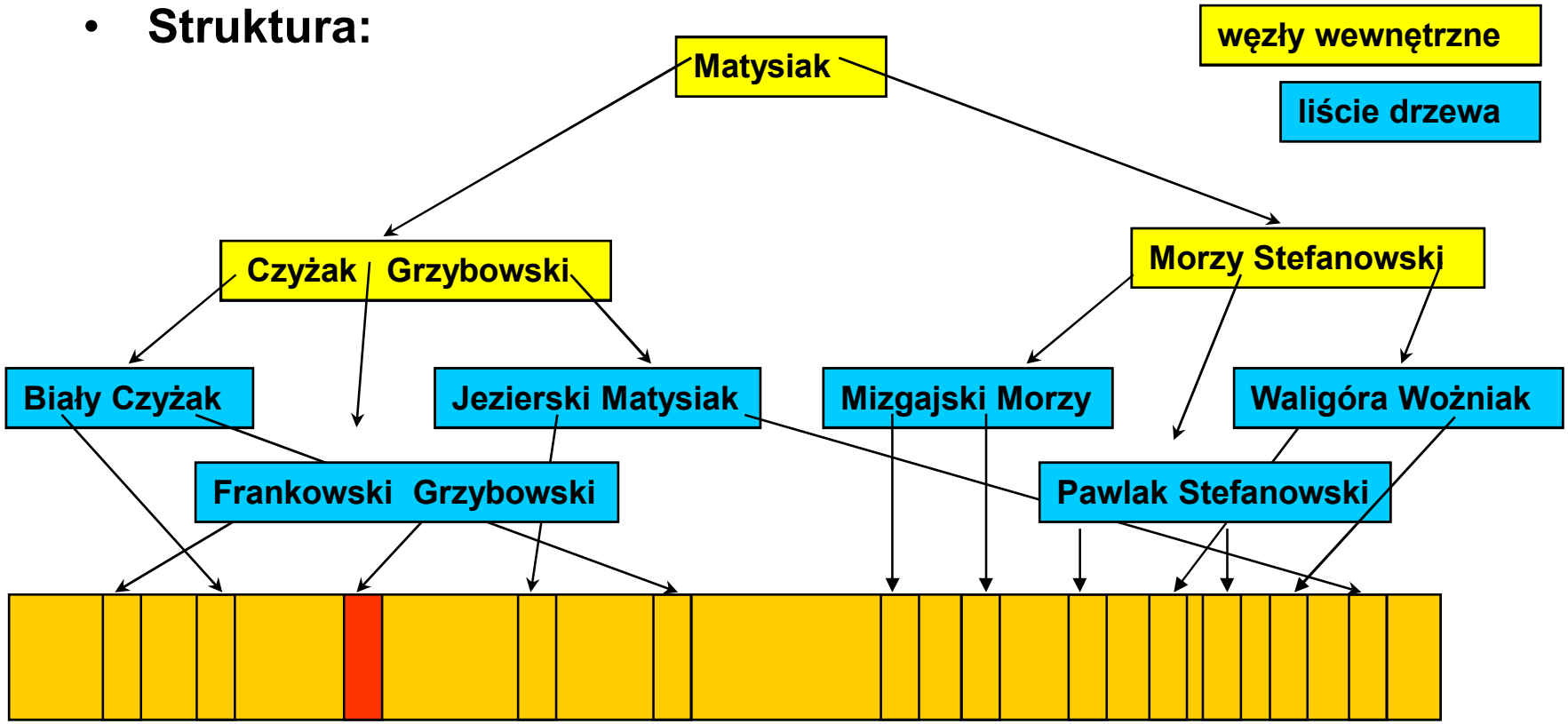
Podział indeksów

- Ze względu na strukturę:
  - B-drzewa, bitmapowe.
- Ze względu na liczbę atrybutów indeksowych w kluczu:
  - indeksy zwykłe i indeksy złożone.
- Ze względu na unikalność wartości klucza:
  - indeksy unikalne i indeksy nieunikalne.
- Ze względu na kolejność wartości klucza:
  - indeksy zwykłe i indeksy odwrócone.
- Ze względu na sposób składowania:
  - indeksy nieskompresowane i indeksy skompresowane.
- Ze względu na zastosowania:
  - indeksy funkcyjne i bitmapowe indeksy połączeniowe.



Indeks typu B-drzewo

- Najczęściej stosowany w systemach OLTP (np. systemach obsługi bieżącej).
- Definiowany tylko dla atrybutów o dużej selektywności.
- Struktura:



Indeks typu B-drzewo (cd)

- **Cechy:**
  - **z kluczem indeksowym przechowywana lista adresów rekordów, w których wartości atrybutów indeksowych są równe wartości klucza,**
  - **wielkość indeksu słabo zależna od rozmiaru dziedziny atrybutu indeksowego,**
  - **efektywne wykonywanie operacji: koniunkcji, zapytań równościowych i przedziałowych, sortowania, testowania unikalności atrybutu, wyliczania wartości minimalnej i maksymalnej, grupowania, eliminacji powtórzeń,**
  - **wysoka współbieżność modyfikacji,**
  - **niski koszt pojedynczej modyfikacji, wysoki koszt modyfikacji grupy rekordów,**
  - **rozmiar indeksu może znacznie przewyższać rozmiar danych relacji,**

Indeks typu B-drzewo (cd)

- **Cechy (cd):**
  - **nie przechowuje informacji o wartościach pustych,**
  - **dla relacji połączonych kluczem obcym eliminuje konieczność blokady relacji podrzędnej (tej, w której zdefiniowano klucz) w przypadku usuwania lub modyfikacji rekordów relacji nadrzędnej (tej, na którą wskazuje klucz).**
- **Składnia polecenia:**

```
CREATE  INDEX nazwa ON relacja(atrybut);
```

- **Przykład:**

```
CREATE INDEX nazwisko_idx ON pracownicy(nazwisko);
CREATE INDEX placa_pod_idx ON pracownicy(placa_pod);
```

Indeks bitmapowy

- **Stosowany najczęściej w systemach OLAP (np. hurtowniach danych).**
- **Definiowany tylko dla atrybutów o małej selektywności, atrybuty kandydujące:**
  - **liczba różnych wartości atrybutu powinna być mniejsza niż 1% liczby rekordów w relacji, lub**
  - **wartości atrybutu powtarzają się ponad 100 razy w relacji.**
- **Cechy:**
  - **z kluczem przechowywana bitmapa, której pozycje odpowiadają adresom rekordów o wartościach atrybutów indeksowych równych wartości klucza,**
  - **konwersja pozycji bitmapy na adres rekordu realizowana przez funkcję mapującą,**
  - **stosowane w zapytaniach z warunkami z operatorem „=”,**
  - **wielkość indeksu silnie zależna od rozmiaru dziedziny atrybutu indeksowego,**

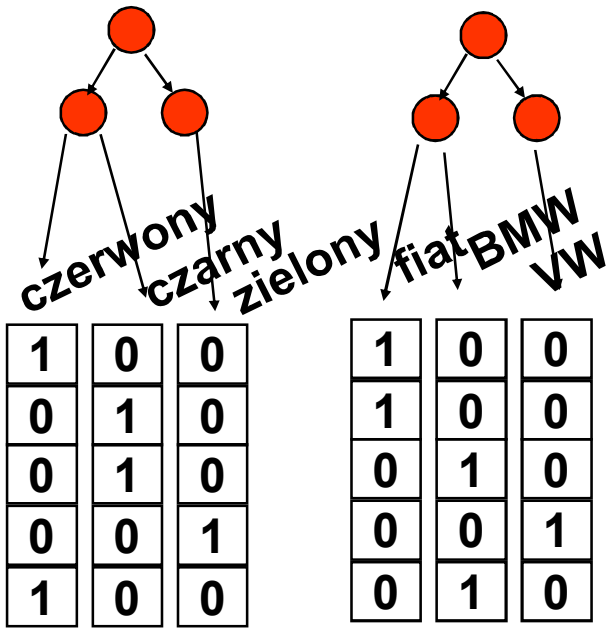
Indeks bitmapowy (cd)

- **Cechy (cd.):**
  - **efektywne wykonywanie operacji koniunkcji, alternatywy i negacji,**
  - **wykorzystywany w zapytaniach z poszukiwaniem wartości pustych,**
  - **niska współbieżność modyfikacji – konieczność blokady całej mapy bitowej,**
  - **wysoki koszt pojedynczej modyfikacji, niski koszt modyfikacji grupy rekordów,**
  - **rozmiar indeksu jest najczęściej ułamkiem rozmiaru danych relacji.**
- **W poleceniu tworzenia indeksu dodatkowa klauzula BITMAP,**
- **Przykład:**

```
CREATE BITMAP INDEX prac_plec_bmp_idx ON pracownicy (plec);
```

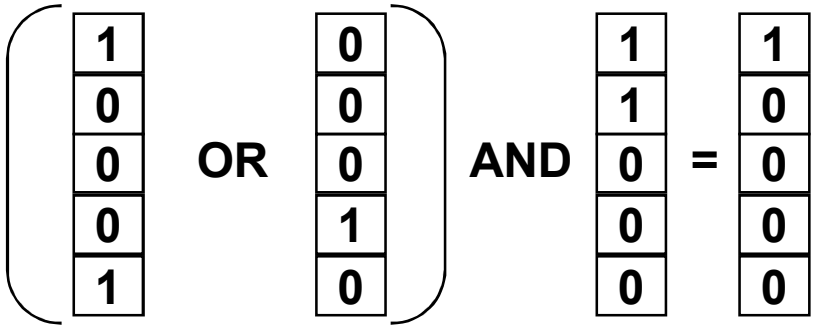
Indeks bitmapowy (cd)

Nr_rej	Kolor	Marka
PWG01425	czerwony	fiat
WAW3456	czarny	fiat
POZ3756	czarny	BMW
KTW3756	zielony	VW
PNR8956	czerwony	BMW




```
CREATE BITMAP INDEX marka_bmp_idx
ON samochody (marka);
CREATE BITMAP INDEX kolor_bmp_idx
ON samochody(kolor);
```

```
SELECT COUNT(*) FROM samochody
WHERE kolor IN ('czerwony', ‘zielony')
AND marka = 'fiat’;
```



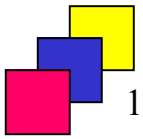
Indeks złożony

- **Klucz indeksu zawiera więcej niż jeden atrybut relacji.**
- **Maksymalnie 32 atrybuty w kluczu indeksu (30 dla indeksu bitmapowego).**
- **Dla indeksu założonego na atrybutach ABC kombinacje atrybutów: A, AB i ABC to tzw. części wiodące klucza, w przeciwieństwie do kombinacji B, BC oraz C.**
- **Przykład:**

```
CREATE INDEX nazw_etat_idx ON pracownicy(nazwisko, etat);
CREATE INDEX id_prac_etat_placa_idx
                ON pracownicy(id_prac, etat, placa_pod);
```

Indeks złożony (cd)

- **Kiedy zakładać indeks złożony:**
  - **na atrybutach często występujących razem w klauzuli WHERE zapytań**
  - **na atrybutach często odczytywanych wspólnie przez wiele zapytań.**
- **Jak wybrać kolejność atrybutów w kluczu:**
  - **atrybuty wykorzystywane w klauzuli WHERE powinny stanowić część wiodącą klucza,**
  - **atrybuty wykorzystywane częściej w klauzuli WHERE powinny stanowić część wiodącą klucza,**
  - **jeśli częstotliwość atrybutów jest taka sama, pierwszym atrybutem powinien być ten, wg którego wartości danych są fizycznie posortowane.**



Indeks unikalny i nieunikalny

- **Indeks unikalny – gwarantuje, że w relacji nie będzie dwóch rekordów z tą samą wartością atrybutu indeksowego (atrybutów indeksowych w przypadku indeksu złożonego), w przeciwieństwie do indeksu nieunikalnego.**
- **W SZBD Oracle indeksy unikalne są tworzone automatycznie przy definiowaniu ograniczeń integralnościowych typu klucz podstawowy i klucz unikalny.**
- **W poleceniu tworzenia indeksu dodatkowa klauzula UNIQUE.**
- **Przykład:**

```
CREATE UNIQUE INDEX id_prac_idx ON pracownicy(id_prac);
CREATE UNIQUE INDEX id_zesp_idx ON zespoly(id_zesp);
```

- **Uwaga! Nie można utworzyć unikalnego indeksu bitmapowego!**

Indeks funkcyjny

- **Definiowany dla atrybutów, które w zapytaniach często używane są jako parametry funkcji (np. upper(nazwisko)) bądź elementy wyrażeń (np. placa_pod * 1.2).**
- **Może być zaimplementowany jako indeks B-drzewo lub indeks bitmapowy,.**
- **SZBD nie użyje indeksu niefunkcyjnego, założonego na atrybucie A, gdy w zapytaniu A jest parametrem funkcji lub elementem wyrażenia.**
- **Indeksowane wyrażenie nie może zawierać funkcji agregujących.**
- **Przykład:**

```
SELECT * FROM pracownicy
WHERE UPPER(nazwisko) = 'NOWAK'
AND placa_dod*2 = placa_pod;
```

```
CREATE INDEX nazw_idx_fun ON pracownicy(UPPER(nazwisko));
CREATE INDEX placa_dod_idx_fun ON pracownicy(placa_dod*2);
```

Indeks odwrócony

- **Wartości w kluczu indeksowym składowane są w postaci odwróconej.**

wartość oryginalna	wartość składowana
802121	121208
802122	221208
802123	321208

- **Stosowany do indeksowania sekwencji, powoduje rozproszenie wartości w indeksie.**
- **Zapobiega rywalizacji transakcji o możliwość zapisu tego samego bloku indeksu.**
- **W poleceniu tworzenia indeksu dodatkowa klauzula REVERSE.**
- **Przykład:**

```
CREATE INDEX lp_rev_idx ON pracownicy(lp) REVERSE;
```

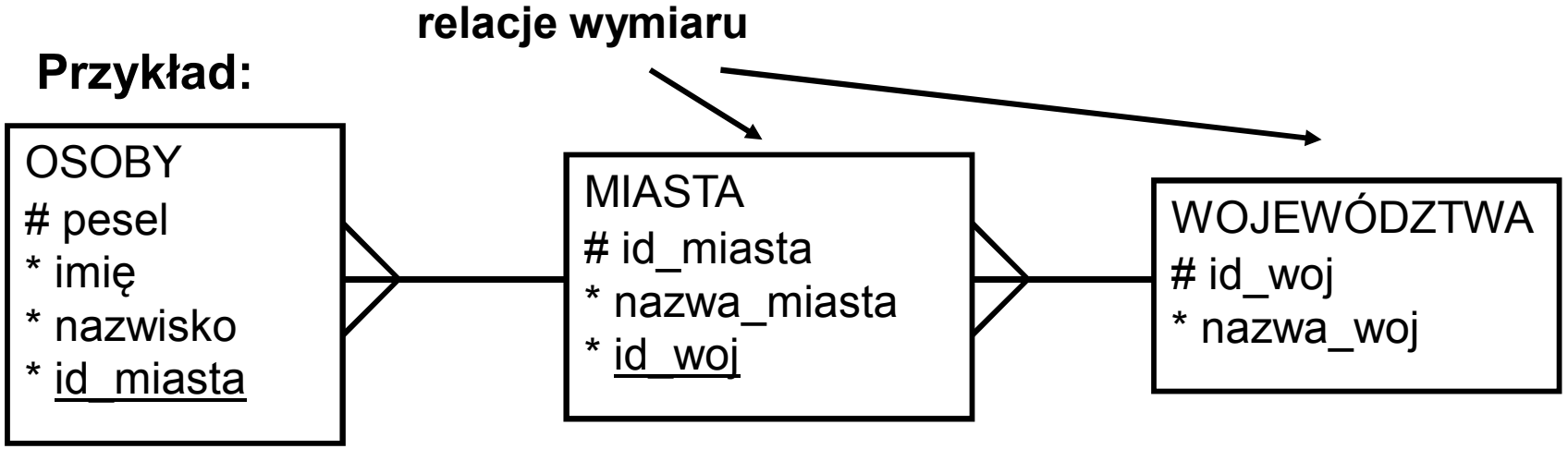
Bitmapowy indeks połączeniowy

- **Indeks definiowany dla operacji równościowego połączenia dwóch lub więcej relacji.**
- **Dla każdej wartości atrybutu indeksowego relacji składowane są adresy rekordów drugiej relacji, które mają tą samą wartość atrybutu połączeniowego.**
- **Wykorzystywany przy zapytaniach łączących relacje.**
- **Składnia:**

```
CREATE BITMAP INDEX nazwa ON relacja (lista_atrybutów)
FROM relacja_1, relacja_2, ..., relacja_n
WHERE warunek_połączeniowy_1 AND warunek_połączeniowy_2 ...
AND warunek_połączeniowy_n-1;
```

Bitmapowy indeks połączeniowy (cd)

- **Przykład:**




```
SELECT COUNT(*)
FROM osoby NATURAL JOIN miasta NATURAL JOIN województwa
WHERE nazwa_woj = 'Wielkopolskie';
```

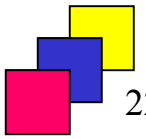
```
CREATE BITMAP INDEX os_mi_woj_bmp_idx
ON osoby(nazwa_woj)
FROM osoby o, miasta m, województwa w
WHERE o.id_miasta = m.id_miasta AND m.id_woj = w.id_woj;
```

Bitmapowy indeks połączeniowy (cd)

- **Ograniczenia:**
  - **relacja nie może pojawiać się wielokrotnie w złączeniu,**
  - **przy tworzeniu indeksu w definicji połączenia nie można używać składni ANSI,**
  - **poindeksowane atrybuty muszą być atrybutami relacji wymiarów,**
  - **atrybuty relacji wymiarów, umieszczone w warunku połączeniowym, muszą mieć zdefiniowane ograniczenia typu klucz podstawowy lub klucz unikalny.**

Kompresja indeksu

- **Redukuje zajętości przestrzeni dyskowej przez powtarzające się wartości klucza indeksu.**
- **Ograniczenia:**
  - **tylko dla indeksów typu B-drzewo (indeksy bitmapowe są kompresowane automatycznie),**
  - **dla indeksów nieunikalnych wszystkie atrybuty klucza mogą zostać skompresowane,**
  - **dla indeksów unikalnych przynajmniej jeden z atrybutów klucza musi pozostać nieskompresowany.**
- **Zalety:**
  - **duża oszczędność przestrzeni dyskowej – większa liczba kluczy indeksu składowana w bloku,**
  - **zwiększenie wydajności operacji we/wy z użyciem indeksu.**
- **Wady:**
  - **większe zużycie CPU w celu dekompresji kluczy przy przeglądaniu indeksu,**



Kompresja indeksu (cd)

- **Domyślnie indeksy są tworzone jako indeksy bez kompresji.**
- **Dodatkowa klauzula COMPRESS *n*, gdzie *n* to liczba atrybutów w części wiodącej klucza, które mają zostać skompresowane.**
- **Przykład:**

```
CREATE INDEX prac_idx ON pracownicy(id_zesp, etat, placa_pod)
COMPRESS 2;
```

  - indeks będzie kompresował powtarzające się wartości kolumn *id_zesp* i *etat.*

Zarządzanie indeksami

- **Usunięcie indeksu:**

```
DROP INDEX nazwa_indeksu;
```

- **Przebudowa indeksu:**

```
ALTER INDEX nazwa_indeksu REBUILD;
```

- **Zmiana nazwy indeksu:**

```
ALTER INDEX nazwa_indeksu RENAME TO nowa_nazwa;
```

Słownik danych

USER_INDEXES	**informacje o wszystkich indeksach, będących własnością użytkownika (synonim IND)**
USER_IND_COLUMNS	**informacje o poindeksowanych atrybutach**
USER_IND_EXPRESSIONS	**informacje o wyrażeniach, na których zbudowano indeksy funkcyjne**
USER_JOIN_IND_COLUMNS	**informacje o atrybutach w warunkach połączeniowych dla indeksów połączeniowych**

```
SELECT index_name, i.index_type, i.uniqueness, c.column_name
FROM user_indexes i JOIN user_ind_columns c USING (index_name)
WHERE i.table_name = 'PRACOWNICY'
ORDER BY index_name, c.column_position;
```

Relacja zorganizowana jak indeks (IOT)

- **Dane relacji przechowywane w strukturze B-drzewa.**
- **Uporządkowanie danych w drzewie wg wartości atrybutów klucza podstawowego relacji, rekord jest identyfikowany przez wartość klucza podstawowego a nie przez adres rekordu (ROWID).**
- **W liściu B-drzewa znajduje się cały rekord relacji, dla relacji o dużym rozmiarze rekordu można zdefiniować dodatkowe miejsce składowania, tzw. obszar przepełnienia, poza liśćmi B-drzewa dla atrybutów niekluczowych.**
- **Zalety:**
  - **szybszy dostęp do danych w zapytaniach z warunkiem zbudowanym z atrybutów klucza podstawowego,**
  - **brak konieczności zakładania indeksów na atrybutach klucza podstawowego – oszczędność przestrzeni dyskowej,**
  - **na atrybutach niekluczowych można zdefiniować dodatkowe indeksy (dzięki istnieniu logicznego ROWID).**

Relacja zorganizowana jak indeks (IOT) (cd)

- **Ograniczenia:**
  - **IOT musi mieć zdefiniowany klucz podstawowy,**
  - **polecenie modyfikacji wartości klucza podstawowego relacji może wymagać przebudowy całej struktury.**
- **Składnia polecenia:**

```
CREATE TABLE nazwa (
        definicja atrybutów
        definicja ograniczeń integralnościowych)
ORGANIZATION INDEX
[PCTTHRESHOLD procent]
[OVERFLOW TABLESPACE przestrzeń tabel [INCLUDING atrybut]];
```

- **Informacje w słowniku: USER_TABLES**