Strojenie poleceń SQL

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

- **składa się z rekordów,**
- **rekord 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 zapytanie z warunkiem zawierającym poindeksowany atrybut,**
2. **SZBD szuka w indeksie klucza (zbioru kluczy), którego wartość (wartości) odpowiada wartości poindeksowanego atrybutu w warunku zapytania,**
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?

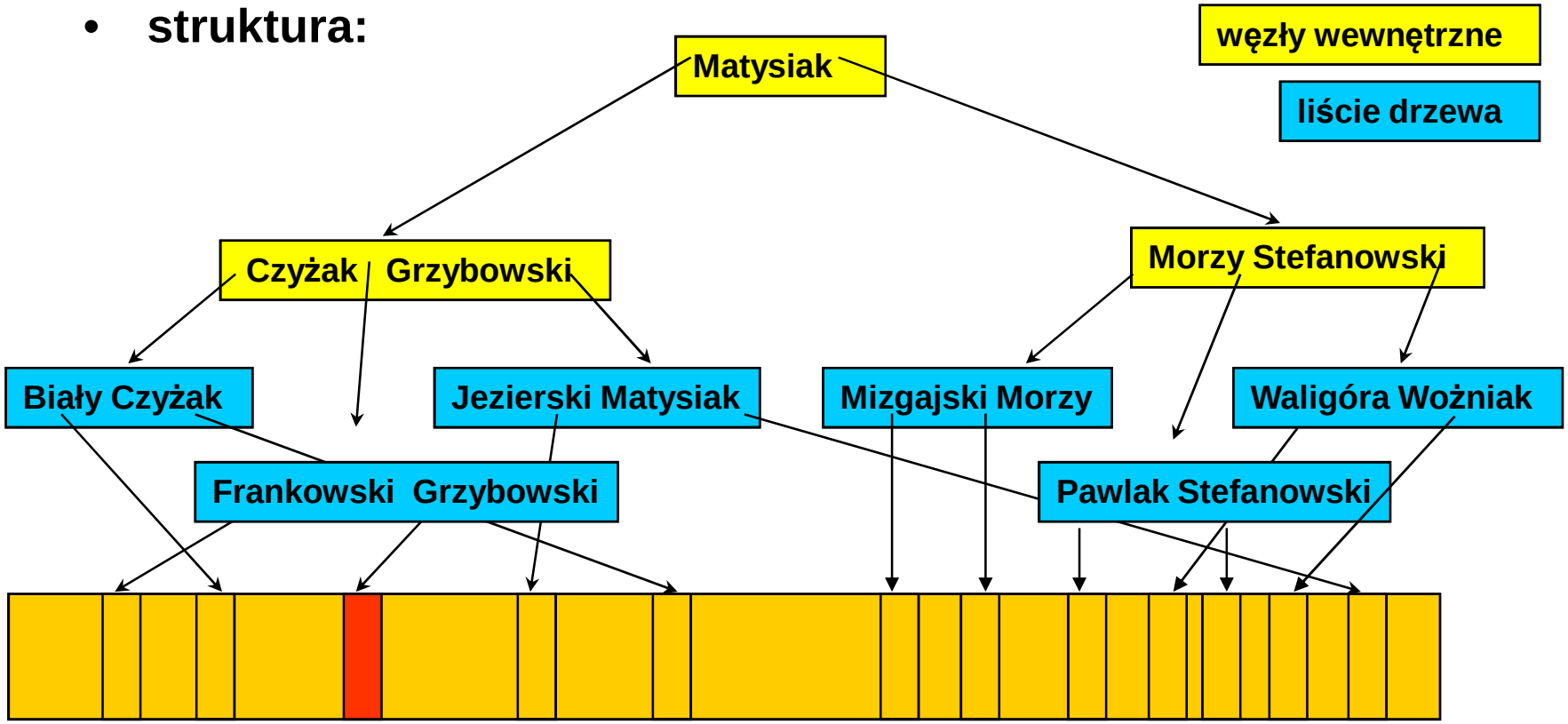
- **atrybuty często używane w klauzulach WHERE zapytań,**
- **atrybuty często używane w warunkach połączeniowych,**
- **atrybuty rzadko modyfikowane,**
- **atrybuty będące kluczami obcymi relacji,**

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,**

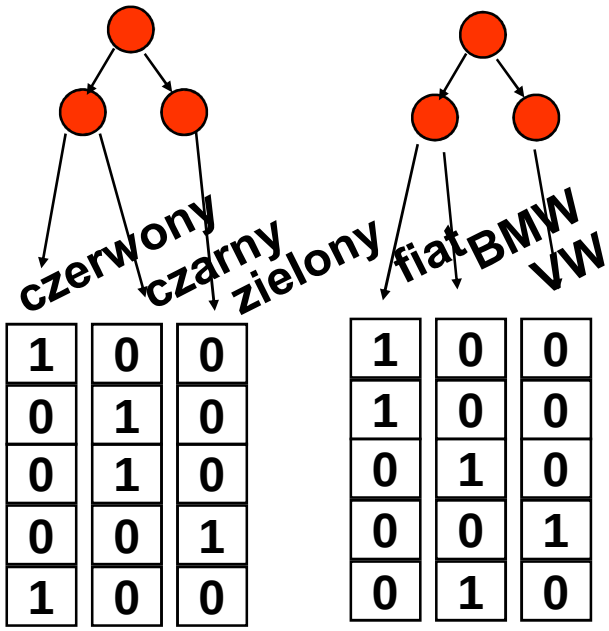
Indeks bitmapowy (cd)

- **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);
```

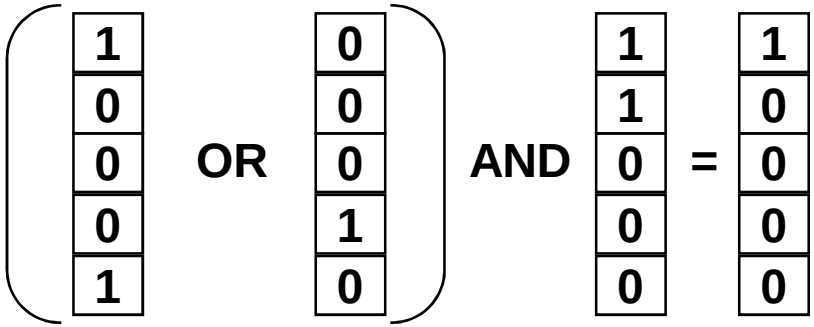
Indeks bitmapowy (cd)

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 ta 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

- **stosowane do indeksowania sekwencji, powodują rozproszenie wartości w indeksie,**
- **w poleceniu tworzenia indeksu dodatkowa klauzula REVERSE,**
- **przykład:**

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

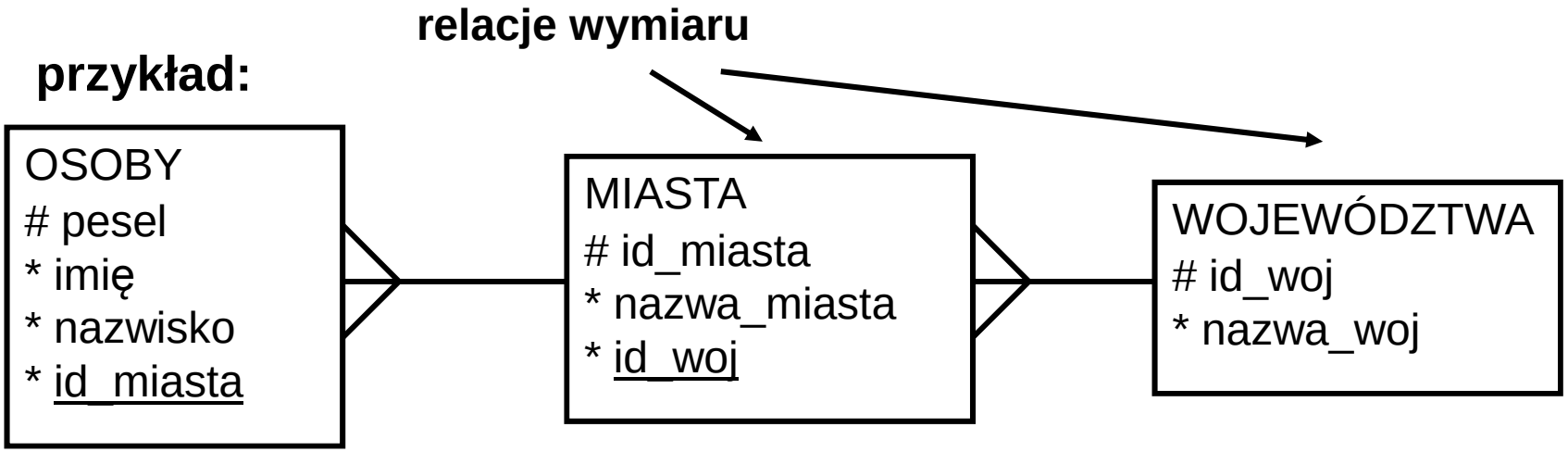
Bitmapowy indeks połączeniowy

- **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,**
  - **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 kluczu, które mają zostać skompresowane,**
- **przykład:**

```
CREATE INDEX prac_idx ON pracownicy(id_zesp, etat)
COMPRESS 1;
```

Zarządzanie indeksami

- **Usunięcie indeksu:**

```
DROP INDEX nazwa_indeksu;
```

- **Przebudowa indeksu:**

```
ALTER INDEX nazwa_indeksu REBUILD;
```

- **Wyliczenie statystyk dla indeksu:**

```
CREATE INDEX ... COMPUTE STATISTICS;
ALTER INDEX nazwa_indeksu COMPUTE STATISTICS;
```

- **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 i.index_name, i.index_type, i.uniqueness, c.column_name
FROM user_indexes i, user_ind_columns c
WHERE i.table_name = 'PRACOWNICY‘
AND i.index_name = c.index_name
ORDER BY i.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 lista
atrybutów]];
```

- **informacje w słowniku: USER_TABLES**

Optymalizacja

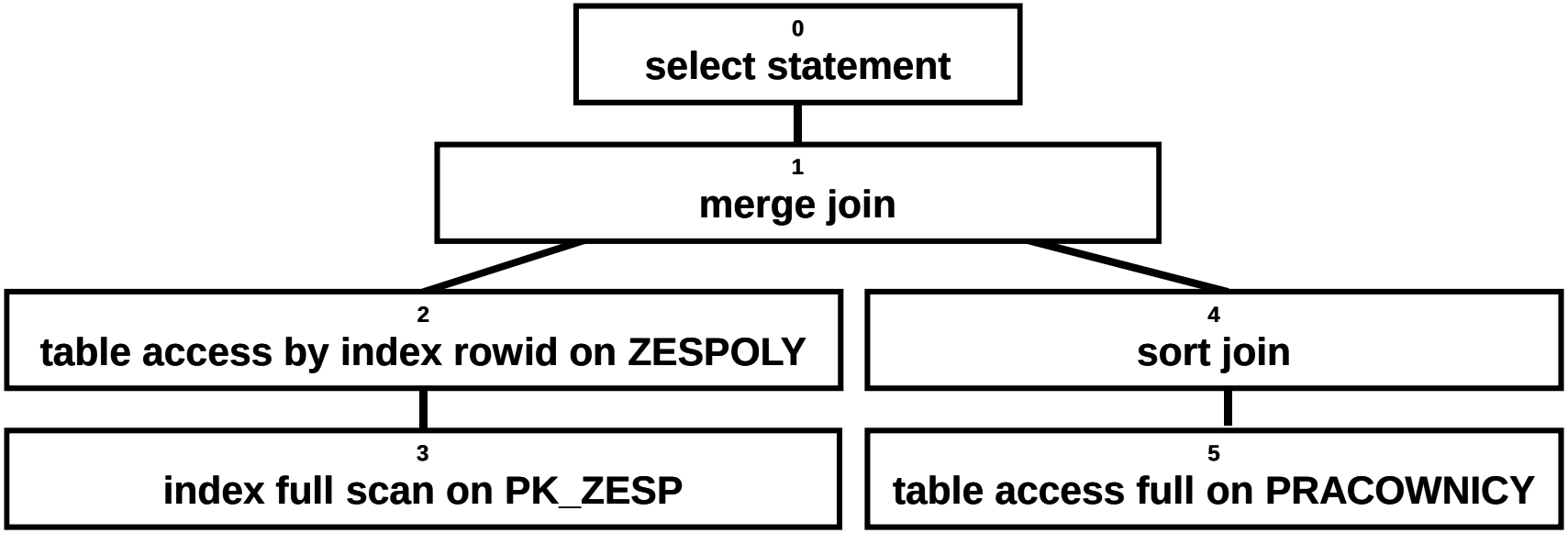
- Proces doboru odpowiednich struktur danych, metod dostępu i operacji, w celu zminimalizowania kosztu realizacji polecenia SQL.
- Wykonywana przez wyspecjalizowany moduł SZBD – optymalizator zapytań.
- Rodzaje:
  - regułowa:
    - oparta na rankingu metod dostępu do struktur danych,
    - niestosowana, preferowana dla aplikacji spadkowych,
  - kosztowa:
    - oparta na szacowaniu kosztu (czas zajętości procesora, liczby operacji we/wy, zajętość pamięci operacyjnej itp.), wykonania wszystkich potencjalnych sposobów wykonania polecenia SQL,
    - zalecana dla wszystkich nowopowstających aplikacji,
    - zakłada duże obciążenie systemu: dużą współbieżność operacji, niski współczynnik trafień w bufor danych.

Plan wykonania polecenia SQL (1)

- **Sekwencja operacji, które SZBD używa do wykonania polecenia SQL.**
- **Podstawowe informacje zawarte w planie wykonania:**
  - **ścieżki dostępu (ang. access path) do każdej z relacji, użytych w poleceniu SQL,**
  - **algorytmy łączenia relacji,**
  - **kolejność łączenia relacji,**
  - **operacje na danych, takie jak filtrowanie, sortowanie, agregacja.**
- **Dodatkowe informacje w planie:**
  - **koszt operacji (% wykorzystania czasu procesora),**
  - **czas wykonania operacji,**
  - **liczba rekordów i rozmiar danych, przetwarzanych przez operację.**
- **Operacje w planie tworzą drzewo.**

Plan wykonania polecenia SQL (2)

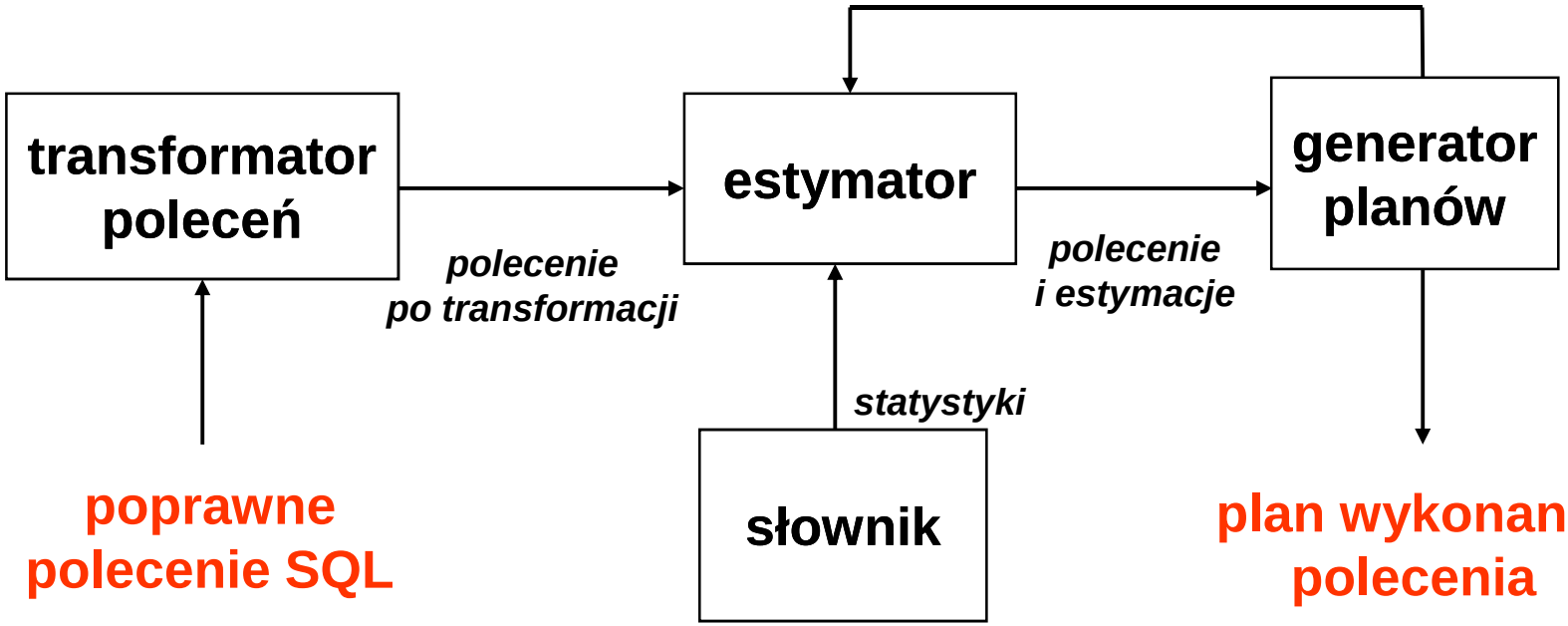
```
SELECT * FROM pracownicy NATURAL JOIN zespoly;
```




```
-------------------------------------------------------------------------------
| Id | Operation                    | Name       | Rows | Bytes| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|  0 | SELECT STATEMENT             |            | 14   | 1036 |      6 (17)| 00:00:01 |
|  1 |  MERGE JOIN                  |            | 14   | 1036 |      6 (17)| 00:00:01 |
|  2 |   TABLE ACCESS BY INDEX ROWID| ZESPOLY    | 6    | 174  |      2  (0)| 00:00:01 |
|  3 |    INDEX FULL SCAN           | PK_ZESP    | 6    |      |      1  (0)| 00:00:01 |
|  4 |   SORT JOIN                  |            | 14   | 630  |      4 (25)| 00:00:01 |
|  5 |    TABLE ACCESS FULL         | PRACOWNICY | 14   | 630  |      3  (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Proces optymalizacji polecenia SQL

1. Sprawdzenie poprawności polecenia przez parser.
2. Przetransformowanie polecenia do optymalnej postaci.
3. Wygenerowanie zbioru potencjalnych planów dla polecenia SQL.
4. Oszacowanie kosztu wykonania każdego planu na podstawie tzw. statystyk.
5. Wybór do realizacji planu z najniższym kosztem wykonania.



Transformator polecenia

- **Przekształca polecenie do optymalnej postaci.**
- **Przykładowe operacje:**
  - **Scalanie perspektyw (ang. view merging).**
  - **Zastąpienie podzapytania połączeniem (ang. subquery unnesting).**
  - **Eliminacja połączeń.**
  - **Zastąpienie warunków z operatorami OR w kilka zapytań, połączonych operatorem UNION ALL (ang. or-expansion).**
  - **Zastąpienie zapytań z operatorami MINUS i INTERSECT przez połączenie.**

Generator planów wykonania polecenia

- Tworzy zbiór różnych planów wykonania tego samego polecenia.
- Plany wykonania różnią przez:
  - zastosowanie różnych kombinacji ścieżek dostępu,
  - zastosowanie różnych algorytmów połączenia relacji,
  - inną kolejność łączenia relacji.
- Dla każdego planu następuje oszacowanie kosztu wykonania.
- Moment zakończenia generacji zbioru planów:
  - Jeśli koszt aktualnego planu jest wysoki, generator szuka lepszego planu bardziej intensywnie (rozpatruje więcej alternatywnych planów).
  - Jeśli koszt aktualnego planu jest niski, wówczas generator szybko kończy poszukiwania z uwagi na małe prawdopodobieństwo poprawy.

Estymator kosztu planu wykonania

- **Zajmuje się szacowaniem kosztu plany wykonania polecenia, wyliczając tzw. miary.**
- **Zbiór rekordów – wynik wykonania operacji w planie: relacja, perspektywa, wynik operacji połączenia, wynik działania operatora GROUP BY.**
- **Miary, pozwalające na oszacowanie całkowitego kosztu planu:**
  - **Selektywność – ułamek reprezentujący liczbę rekordów odczytanych ze zbioru przez operację w stosunku do liczby rekordów w zbiorze. Ściśle związana z warunkami, zdefiniowanymi w poleceniu SQL. Zakres: <0, 1>, selektywność 0 – nie zostały odczytane żadne rekordy, selektywność 1 – zostały odczytane wszystkie rekordy ze zbioru.**
  - **Liczność – liczba rekordów, odczytanych ze zbioru.**
  - **Koszt – reprezentuje jednostki pracy lub zasoby użyte do realizacji operacji w planie wykonania, np. liczba operacji we/wy, użycie procesora, użycie pamięci.**

Polecenie EXPLAIN PLAN (1)

- **Polecenie SZBD (będzie działać w różnych narzędziach).**
- **Zapytanie nie jest wykonywane.**
- **Przebieg:**
  1. **Utworzenie relacji PLAN_TABLE (raz w schemacie):**

```
SQL> @$ORACLE_HOME\RDBMS\ADMIN\UTLXPLAN.SQL
```

  2. **Wygenerowanie planu wykonania zapytania:**

```
EXPLAIN PLAN
[ SET STATEMENT_ID = 'identyfikator' ]
FOR SELECT ... FROM ... WHERE ... ;
```

Polecenie EXPLAIN PLAN (2)

- **Przebieg (cd):**
  3. **Obejrzenie wyników:**
     a) **wykonanie skryptu utlxpls.sql**

```
SQL> @$ORACLE_HOME\RDBMS\ADMIN\UTLXPLS.SQL
```

     b) **wykonanie polecenia**

```
SELECT PLAN_TABLE_OUTPUT
FROM TABLE(DBMS_XPLAN.DISPLAY(NULL,
'identyfikator','SERIAL'));
```

     - **w iSQL*Plus ustawić zmienną markup html preformat na wartość on**

```
set markup html preformat on
```

Polecenie EXPLAIN PLAN (3)

- **Przebieg (cd):**
  3. **Obejrzenie wyników:**
     - **przykład:**

```
SQL> EXPLAIN PLAN set statement_id = ‘plan01’ FOR
     SELECT * FROM pracownicy;
Wyjaśniono
SQL> set markup html preformat on
SQL> SELECT PLAN_TABLE_OUTPUT
     FROM TABLE(DBMS_XPLAN.DISPLAY(NULL,’plan01','BASIC'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 363874572
---------------------------------------
| Id | Operation         | Name       |
---------------------------------------
| 0 | SELECT STATEMENT  |            |
| 1 | TABLE ACCESS FULL | PRACOWNICY |
---------------------------------------
SQL> set markup html preformat off
```

Polecenie EXPLAIN PLAN (4)

- **Przebieg (cd):**
  3. **Obejrzenie wyników:**
     c) **wykonanie zapytania:**

```
SELECT cardinality "Rows", lpad(' ',level-1)||operation||'
'||options||' '||object_name "Plan"
FROM PLAN_TABLE
CONNECT BY prior id = parent_id
AND prior statement_id = statement_id
START WITH id = 0
AND statement_id = ‘identyfikator'
ORDER BY id;
```

Dyrektywa AUTOTRACE (1)

- **Działa tylko w SQL*Plus.**
- **Po wykonaniu polecenia wyświetlany jest raport zawierający plan wykonania polecenia i dodatkowe informacje.**
- **Administrator musi użytkownikowi nadać rolę PLUSTRACE:**

```
SQL> @$ORACLE_HOME\SQLPLUS\ADMIN\PLUSTRCE.SQL
SQL> GRANT PLUSTRACE  TO SCOTT;
```

- **Dyrektywa:**

```
SET AUTOTRACE [ON | OFF] [TRACEONLY]
        [ EXPLAIN ] [ STATISTICS ]
```

Dyrektywa AUTOTRACE (2)

SET AUTOTRACE OFF	wyłączenie generowania raportu
SET AUTOTRACE ON EXPLAIN	wynik zapytania + plan wykonania
SET AUTOTRACE ON STATISTICS	wynik zapytania + statystyki wykonania
SET AUTOTRACE ON	wynik zapytania + plan wykonania + statystyki wykonania
SET AUTOTRACE TRACEONLY	plan wykonania + statystyki

```
SQL> SET AUTOTRACE ON EXPLAIN
SQL> SELECT nazwisko FROM pracownicy WHERE id_prac=100;
NAZWISKO
-----------
KOWALSKI

Plan wykonania
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=1 Card=1 Bytes=22)
   1    0   TABLE ACCESS (BY ROWID) OF 'PRACOWNICY' (Cost=1 Card=1 Bytes=22)
   2    1     INDEX (UNIQUE SCAN) OF 'PK_PRAC' (UNIQUE)
SQL> SET AUTOTRACE OFF
```

Dyrektywa AUTOTRACE (3)

Statystyki wykonania polecenia:

- **db block gets – liczba bloków danych, które zostały odczytane z bufora bazy danych przez polecenia INSERT, UPDATE, DELETE i SELECT FOR UPDATE,**
- **consistent gets – liczba bloków danych, które zostały odczytane z bufora bazy danych przez zapytania bez klauzuli FOR UPDATE.**
- **physical reads – liczba bloków, które zostały odczytane z dysków dla poleceń INSERT, UPDATE, DELETE, SELECT i SELECT FOR UPDATE.**
- **sorts (memory) – liczba operacji sortowania, wykonanych całkowicie w pamięci**
- **sorts (disk) – liczba operacji sortowania, zapisu na dysku**
- **rows processed – liczba przetworzonych rekordów.**
- **logical reads = db block gets + consistent gets – całkowita liczba bloków odczytanych z bufora bazy danych.**

Wybór celu optymalizacji (1)

- **Najlepsza przepustowość – wykorzystanie jak najmniejszych zasobów systemowych do uzyskania wszystkich rekordów polecenia SQL.**
  - **Dla aplikacji wykonywanych wsadowo, np. drukowanych raportów.**
  - **Najważniejszy krótki czas zakończenia całego zadania, mniej ważny czas odpowiedzi.**
- **Najkrótszy czas odpowiedzi – wykorzystanie jak najmniejszych zasobów do uzyskania pierwszego rekordu polecenia SQL.**
  - **Dla aplikacji interaktywnych, np. formularzy ekranowych, zapytań w SQL*Plus.**
  - **Najważniejszy krótki czas odpowiedzi – użytkownik chce jak najszybciej zobaczyć pierwszy rekord (lub kilka pierwszych rekordów) polecenia.**

Wybór celu optymalizacji (2)

- **Dla bieżącej sesji – parametr OPTIMIZER_MODE:**
  - **ALL_ROWS: kosztowa maksymalizująca przepustowość**
  - **FIRST_ROWS: kosztowa minimalizująca czas odpowiedzi**
  - **FIRST_ROWS_n: kosztowa minimalizująca łączny czas odczytania pierwszych n (1,10,100,1000) krotek**

```
ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS;
```

Statystyki (1)

Informacje, opisujące dane i struktury obiektów bazy danych.

Przechowywane w słowniku danych.

Używane przez optymalizator do oszacowania kosztu planu wykonania polecenia.

Tylko aktualne statystyki użyteczne!

Przykłady statystyk:

- **dla relacji:**
  - **liczba rekordów,**
  - **liczba bloków,**
  - **średnia długość rekordu,**
- **dla atrybutu relacji:**
  - **liczba różnych wartości,**
  - **liczba rekordów, w których atrybut ma wartość pustą,**
  - **rozkład wartości (histogram),**

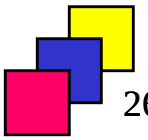
Statystyki (2)

Przykłady statystyk (cd.):

- **dla indeksu:**
  - **liczba bloków-liści,**
  - **wysokość drzewa,**
  - **współczynnik zgrupowania (określa, na ile wiersze w relacji są uporządkowane wg klucza indeksu),**
- **systemowe:**
  - **wykorzystanie procesora,**
  - **liczba operacji we/wy.**

Histogramy – szczegółowe statystyki opisujące rozkład wartości poszczególnych kolumn.

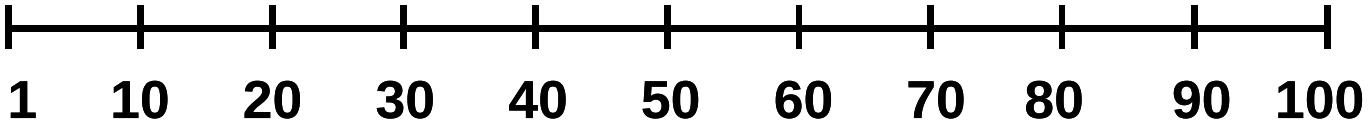
Statystyki mogą być gromadzone automatycznie (przez SZBD) lub ręcznie (na żądanie użytkownika) przy użyciu pakietu DBMS_STATS.



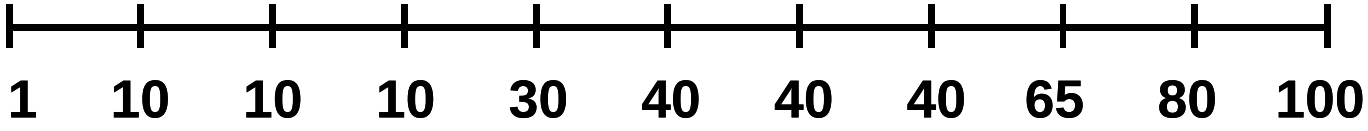
Rodzaje histogramów

Histogramy o zrównoważonej wysokości (ang. height balanced) – zbiór wartości kolumny dzielony jest na przedziały o tej samej liczbie rekordów, przykład (zakres wartości: <1, 100>, liczba przedziałów: 10):

- **równomierny rozkład wartości atrybutu:**



- **nierównomierny rozkład wartości atrybutu:**



Histogramy częstotliwości (ang. frequency) – każda wartość kolumny odpowiada jednemu przedziałowi, każdy przedział zawiera liczbę wystąpień tej wartości. Tworzone wtedy, gdy liczba wartości kolumny jest mniejsza bądź równa żądanej liczbie przedziałów histogramu.

Ręczne zbieranie statystyk

Metody:

- **na podstawie pełnych danych,**
- **szacowanie na podstawie próbki, próbka określana w procentach liczby rekordów.**

Procedury zbierające statystyki:

- **DBMS_STATS.GATHER_INDEX_STATS – dla indeksu,**
- **DBMS_STATS.GATHER_TABLE_STATS – dla relacji.**

Procedury usuwające statystyki:

- **DBMS_STATS.DELETE_INDEX_STATS – dla indeksu,**
- **DBMS_STATS.DELETE_TABLE_STATS – dla relacji,**
- **DBMS_STATS.DELETE_COLUMN_STATS – dla kolumny.**

Zbieranie statystyk dla indeksu

```
exec DBMS_STATS.GATHER_INDEX_STATS(
  ownname => <nazwa_schematu>, indname => <nazwa_indeksu>,
  estimate_percent => <procentowa_wielkość_próbki>);
```

jeśli wartość <procentowa_wielkość_próbki> określono jako:

- **null, wówczas statystyki zbierane na podstawie pełnych danych,**
- **liczbę z przedziału <0,00001; 100>, wówczas szacowanie na podstawie próbki o zadanym rozmiarze,**
- **DBMS_STATS.AUTO_SAMPLE_SIZE – rozmiar próbki dobiera system.**

```
exec DBMS_STATS.GATHER_INDEX_STATS(
  ownname => 'SCOTT', indname => 'PK_PRAC', estimate_percent => 20);
```

Zbieranie statystyk dla relacji

```
exec DBMS_STATS.GATHER_TABLE_STATS(
   ownname => <nazwa_schematu>, tabname => <nazwa_relacji>,
   estimate_percent => <procentowa_wielkość_próbki>,
   method_opt => <rodzaj_statystyk>);
```

<rodzaj_statystyk> określa parametry zbierania histogramów dla kolumn analizowanej relacji:

- **FOR ALL COLUMNS [SIZE liczba_przedziałów | AUTO]**
- **FOR COLUMNS [SIZE liczba_przedziałów | AUTO] kolumna1 [SIZE liczba_przedziałów | AUTO], kolumna2 [SIZE liczba_przedziałów | AUTO], ...**
- **domyślnie: FOR ALL COLUMNS SIZE AUTO**

```
exec DBMS_STATS.GATHER_TABLE_STATS(
   ownname => 'SCOTT', tabname => 'PRACOWNICY',
   estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
   method_opt => 'FOR COLUMNS placa_pod SIZE AUTO, nazwisko SIZE AUTO');
```

Oglądanie statystyk

Dla relacji:

- **USER_TABLES, USER_TAB_STATISTICS**

Dla kolumn:

- **USER_TAB_COLUMNS, USER_TAB_COL_STATISTICS, USER_TAB_HISTOGRAMS**

Dla indeksów:

- **USER_INDEXES, USER_IND_STATISTICS**

```
SELECT num_rows, blocks, last_analyzed, sample_size
FROM USER_TAB_STATISTICS
WHERE table_name = 'PRACOWNICY';
```

Usuwanie statystyk

```
exec DBMS_STATS.DELETE_INDEX_STATS(
  ownname => <nazwa_schematu>, indname => <nazwa_indeksu>);
```

```
exec DBMS_STATS.DELETE_TABLE_STATS(
  ownname => <nazwa_schematu>, tabname => <nazwa_relacji>);
```

```
exec DBMS_STATS.DELETE_COLUMN_STATS(
  ownname => <nazwa_schematu>, tabname => <nazwa_relacji>,
  colname => <nazwa_kolumny>);
```

Metody dostępu do danych

Określają, w jaki sposób dane polecenia SQL są odczytywane z miejsca ich fizycznej lokalizacji.

Dostęp do relacji:

- **pełne przeglądnięcie,**
- **dostęp przy pomocy adresu rekordu.**

Dostęp do indeksu:

- **unikalne przeglądnięcie indeksu,**
- **(odwrócone) zakresowe przeglądnięcie indeksu,**
- **przeglądnięcie indeksu z pominięciem kolumn,**
- **pełne przeglądnięcie indeksu,**
- **szybkie pełne przeglądnięcie indeksu,**
- **dostęp do indeksu bitmapowego,**
- **połączenie indeksów.**

Przy dostępie do indeksu dane zwykle zwracane w kolejności rosnącej.

Ogólne zasady dostępu do danych:

- **odczyt dużej części rekordów relacji – pełne przeglądnięcie relacji,**
- **odczyt pojedynczych rekordów relacji – dostęp za pomocą indeksu.**

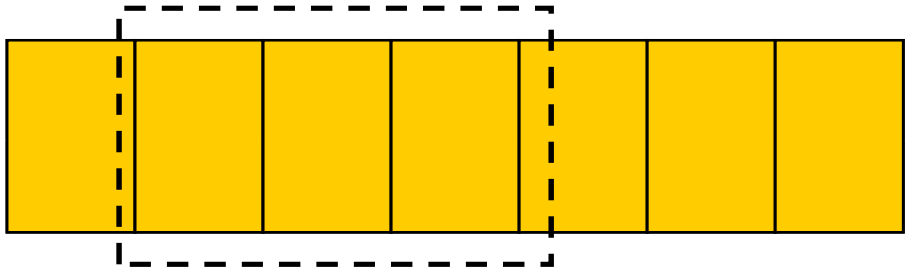
Pełne przeglądniecie relacji

Ang. full table scan

Sekwencyjny odczyt wszystkich bloków danych, w których relacja przechowuje swoje rekordy, odfiltrowanie rekordów nie spełniających zdefiniowanych w poleceniu SQL kryteriów selekcji (np. w klauzuli WHERE).

Stosowane gdy:

- **brak indeksu dla relacji lub nie można użyć istniejących indeksów,**
- **zostanie odczytana duża część wszystkich bloków, w których relacja składuje swoje dane,**
- **rozmiar relacji jest niewielki.**



Możliwy odczyt wieloblokowy – pobranie w jednej operacji I/O wielu przyległych bloków danych, bardziej efektywne niż wiele odczytów pojedynczych bloków.

Dostęp przy pomocy adresu rekordu

Ang. rowid scan

Odszukanie rekordu relacji na podstawie dostarczonego adresu rekordu (rowid).

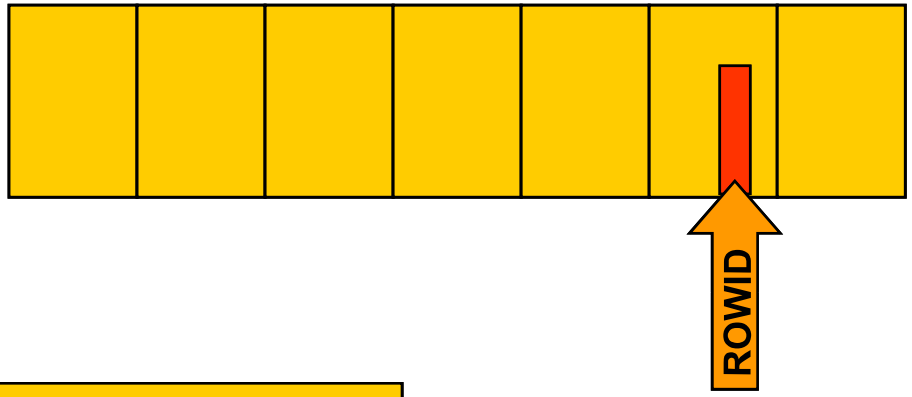
Najszybszy sposób dostępu do rekordów relacji.

Źródło adresu rekordu:

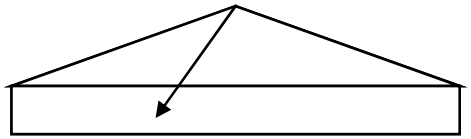
- **warunek selekcji polecenia SQL,**

```
SELECT * FROM pracownicy
WHERE rowid = 'AAAMMUAAEAAAAAtAAG';
```

- **pobranie z indeksu relacji.**



Unikalne przeglądnięcie indeksu

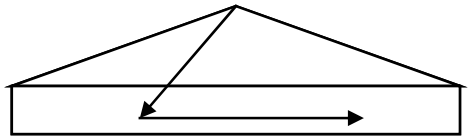


Ang. index unique scan

Dostęp do indeksu unikalnego, operacja zwraca co najwyżej jeden adres rekordu.

Stosowane, gdy w poleceniu SQL zastosowano warunek równościowy z atrybutem, na którym zdefiniowano indeks unikalny (również ograniczenia klucz podstawowy i klucz unikalny).

Zakresowe przeglądnięcie indeksu



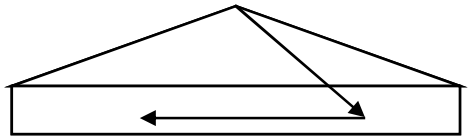
Ang. index range scan

Dostęp do indeksu unikalnego (warunek inny niż równościowy) lub nieunikalnego, operacja zwraca zakres adresów rekordów.

Stosowane, gdy w poleceniu SQL:

- **warunek selekcji z kolumnami z części wiodącej indeksu, takie jak:**
  - **kolumna = ‘wartość’, kolumna > ‘wartość’, kolumna < ‘wartość’, kolumna like ‘ABC%’ (% nie może być na początku wzorca),**
- **warunek złożony z ww. warunków ze spójnikiem AND,**
- **klauzula ORDER BY lub GROUP BY z atrybutami z części wiodącej indeksu.**

Odwrócone zakresowe przeglądnięcie indeksu



Ang. index range scan descending

Odmiana zakresowego przeglądnięcia indeksu.

Dane zwracane w kolejności malejącej .

Stosowane, gdy:

- **w poleceniu konieczne posortowanie danych w porządku malejącym,**
- **przy poszukiwaniu wartości mniejszych niż wartość wyspecyfikowana.**

Przeglądnięcie indeksu z pominięciem kolumn

Ang. index skip scan

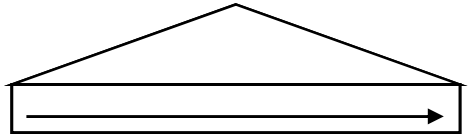
Operacja korzystająca z indeksu złożonego dla polecenia, w którym nie występuje kolumna z początku części wiodącej klucza indeksowego:

- **indeks dzielony jest na mniejsze podindeksy, liczba podindeksów jest równa liczbie wartości pierwszej kolumny w kluczu indeksowym,**
- **podindeksy skanowane są kolejno – operacja zastępuje pełne przeglądnięcie relacji.**

Przykład:

- **relacja Pracownicy(id_prac, adres, płeć), indeks o strukturze klucza: (płeć, id_prac), zapytanie: select * from Pracownicy where id_prac = 100**
- **indeks zostaje podzielony na dwa podindeksy: dla wartości płeć = 'M' i dla wartości płeć = 'K', podindeksy zostają przeskanowane kolejno.**

Pełne przeglądnięcie indeksu



Ang. full index scan

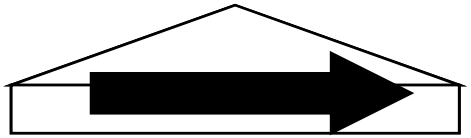
Stosowane, gdy:

- **w warunku polecenia SQL odwołania do kolumn z klucza indeksowego, kolumny nie muszą być częścią wiodącą klucza,**
- **brak odwołań do poindeksowanych kolumn w warunku polecenia, ale:**
  - **wszystkie kolumny, do których występuje odwołanie w poleceniu (np. w klauzuli SELECT), znajdują się w kluczu indeksowym,**
  - **przynajmniej jedna z tych kolumn nie jest pusta.**

Odczytywane są wszystkie liście indeksu w porządku, bloki indeksu odczytywane pojedynczo.

Może być użyty do eliminacji operacji sortowania relacji – dane są posortowane wg klucza indeksowego.

Szybkie pełne przeglądnięcie indeksu



Ang. fast full index scan

Stosowane, gdy:

- **wszystkie kolumny, które są używane w poleceniu SQL, występują w kluczu indeksowym,**
- **co najmniej jedna z tych kolumn ma zdefiniowane ograniczenie NOT NULL.**

Zastępuje pełne przeglądnięcie relacji – wynik polecenia SQL uzyskuje się bezpośrednio z indeksu, bez konieczności dostępu do relacji.

Odczytywane są wszystkie liście indeksu przy zastosowaniu odczytu wieloblokowego – większa wydajność niż pełne przeglądnięcie indeksu, ale nie zostaje zachowane uporządkowanie.

Nie może być używany do eliminacji operacji sortowania relacji – dane nie są posortowane wg klucza indeksowego.

Dostęp do indeksu bitmapowego

Składa się z dwóch kroków:

1. **dostęp do bitmapy,**
2. **konwersja bitmapy do adresów rekordów (krok opuszczany w przypadku możliwości realizacji polecenia bez dostępu do relacji).**

W przypadku poleceń z warunkiem złożonym (spójniki AND i OR, negacja), operacje koniunkcji, alternatywy i negacji wykonywane bezpośrednio na bitmapach (widoczne w planie wykonania polecenia).

Połączenie indeksów

Ang. index join

Stosowane w przypadku, gdy wszystkie kolumny, używane w poleceniu SQL, znajdują się w kluczach kilku różnych indeksów.

Wynik polecenia uzyskuje się tylko z indeksów, bez konieczności dostępu do relacji.

Nie może być stosowane do eliminacji operacji sortowania relacji.

Przykład:

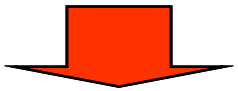
```
SELECT id_prac FROM pracownicy WHERE placa_pod >1000;
```

Range scan(indeks na *placa_pod)*

placa_pod	ROWID
1600	00000001.001.001
...	...
2000	00000001.0A1.01E

Fast Full Scan(indeks na *id_prac)*

id_prac	ROWID
120	00000001.0A1.01E
...	...
140	00000001.001.001



join (hash)

120
...
140

Połączenie

Operacja binarna – zawsze udział biorą dwie relacje, jedna zostaje nazwana relacją zewnętrzną, druga relacją wewnętrzną.

W przypadku polecenia łączącego więcej niż dwie relacje (np. A , B i C), połączenie realizowane jest zawsze dla pary relacji (np. A z B, wynik z C, albo A z C i wynik z B, itd.).

Wybór kolejności łączenia relacji jest kluczowy! Główna zasada: kolejność łączenia relacji powinna jak najbardziej ograniczać zbiór rekordów.

Realizowane przy użyciu jednego z algorytmów:

- **nested loops,**
- **sort merge,**
- **hash join.**

  Wybór algorytmu zależy od rozmiaru relacji, postaci warunku połączeniowego i spodziewanego rozmiaru wyniku połączenia.

Algorytm nested loops

Stosowany, gdy:

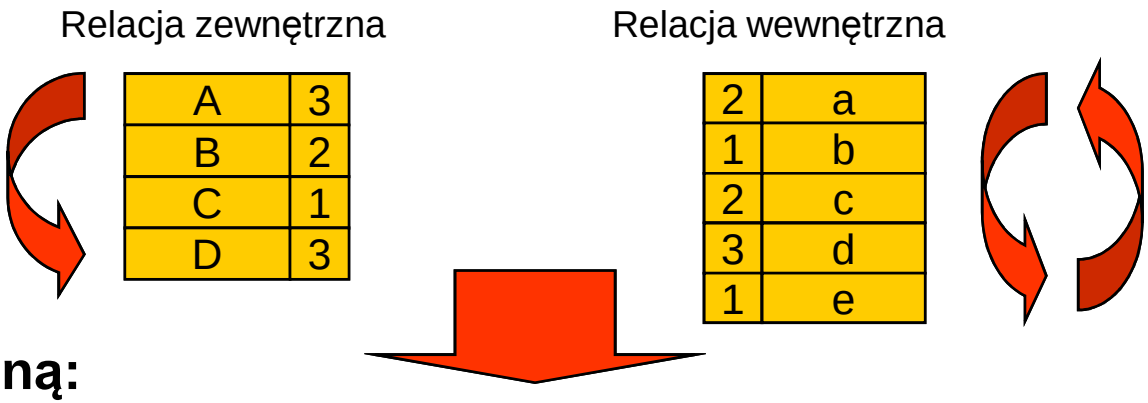
- **w połączeniu bierze udział mała część rekordów relacji,**
- **istnieje efektywna metoda dostępu do danych relacji wewnętrznej (indeks założony na kolumnie w warunku połączeniowym).**

Główny koszt – koszt odczytu rekordów relacji zewnętrznej i znalezienia odpowiadających rekordów relacji wewnętrznej.

Algorytm:

W planie wykonania relacja zewnętrzna ponad relacją wewnętrzną:

NESTED LOOPS
relacja_zewnętrzna
relacja_wewnętrzna



A	3	3	d
B	2	2	a
B	2	2	c
C	1	1	b
C	1	1	e
D	3	3	d

Wynik połączenia

Algorytm sort merge

Stosowany, gdy:

- **łączone relacje są niezależne (brak połączenia kluczem obcym),**
- **warunek połączeniowy z operatorami: <, <=, >, >= (ale nie !=) i duże rozmiary łączonych relacji (zachowuje się lepiej niż nested loop),**
- **relacja już są posortowane lub nie ma potrzeby realizacji sortowania (bo np. istnieje odpowiedni indeks).**

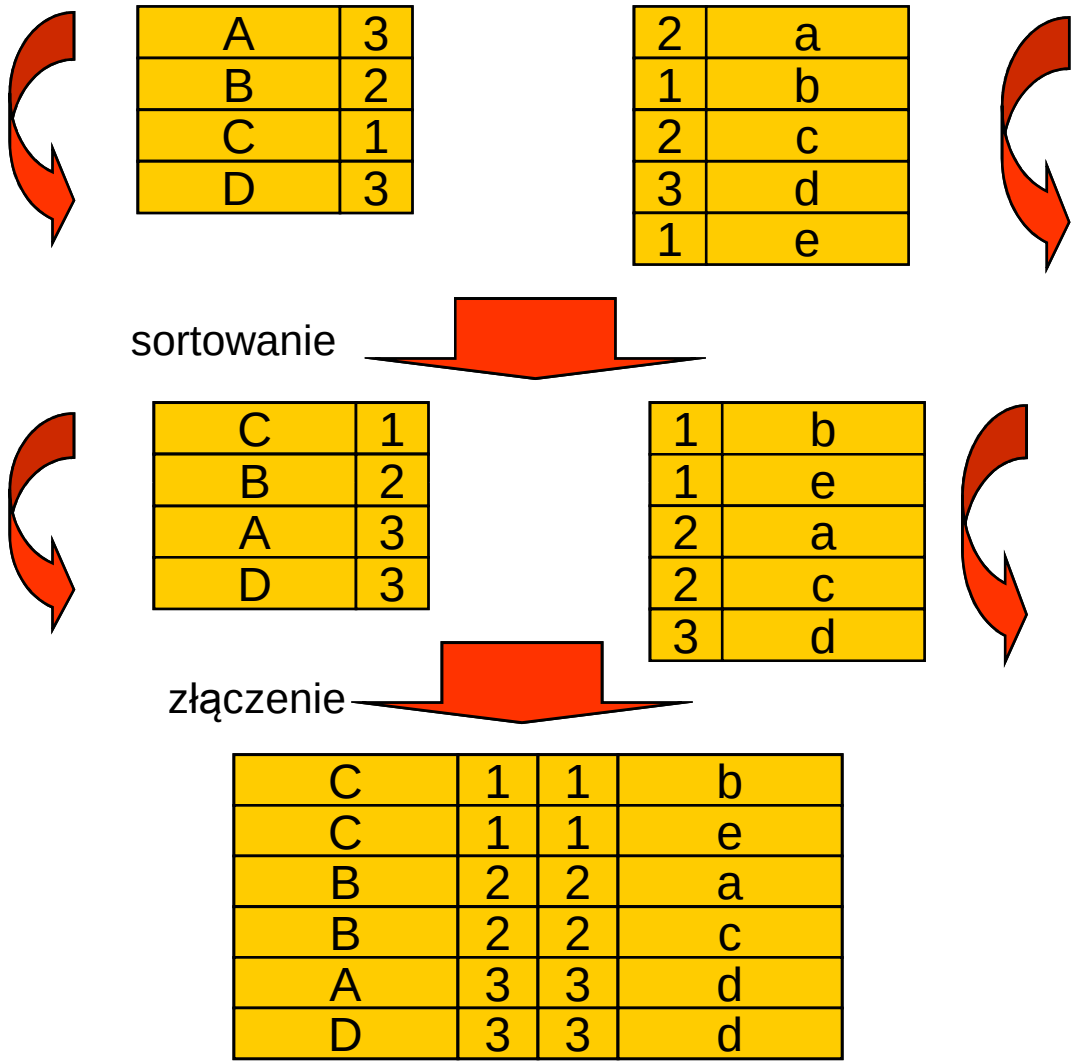
Główny koszt – koszt wczytania obu relacji do pamięci i ich posortowania.

Brak podziału na relację zewnętrzną i wewnętrzną.

Algorytm:

1. **Posortowanie obu relacji ze względu na wartości kolumn w warunku połączeniowym.**
2. **Połączenie rekordów o tych samych wartościach kolumn w warunku połączeniowym.**

Algorytm sort merge (cd)



Algorytm hash join

Stosowany, gdy:

- **warunek połączeniowy jest warunkiem równościowym, i**
- **łączone relacje o dużym rozmiarze lub większa część rekordów mniejszej relacji bierze udział w połączeniu.**

Główny koszt – zbudowanie tabeli haszowej dla relacji zewnętrznej i odczyt rekordów z relacji wewnętrznej.

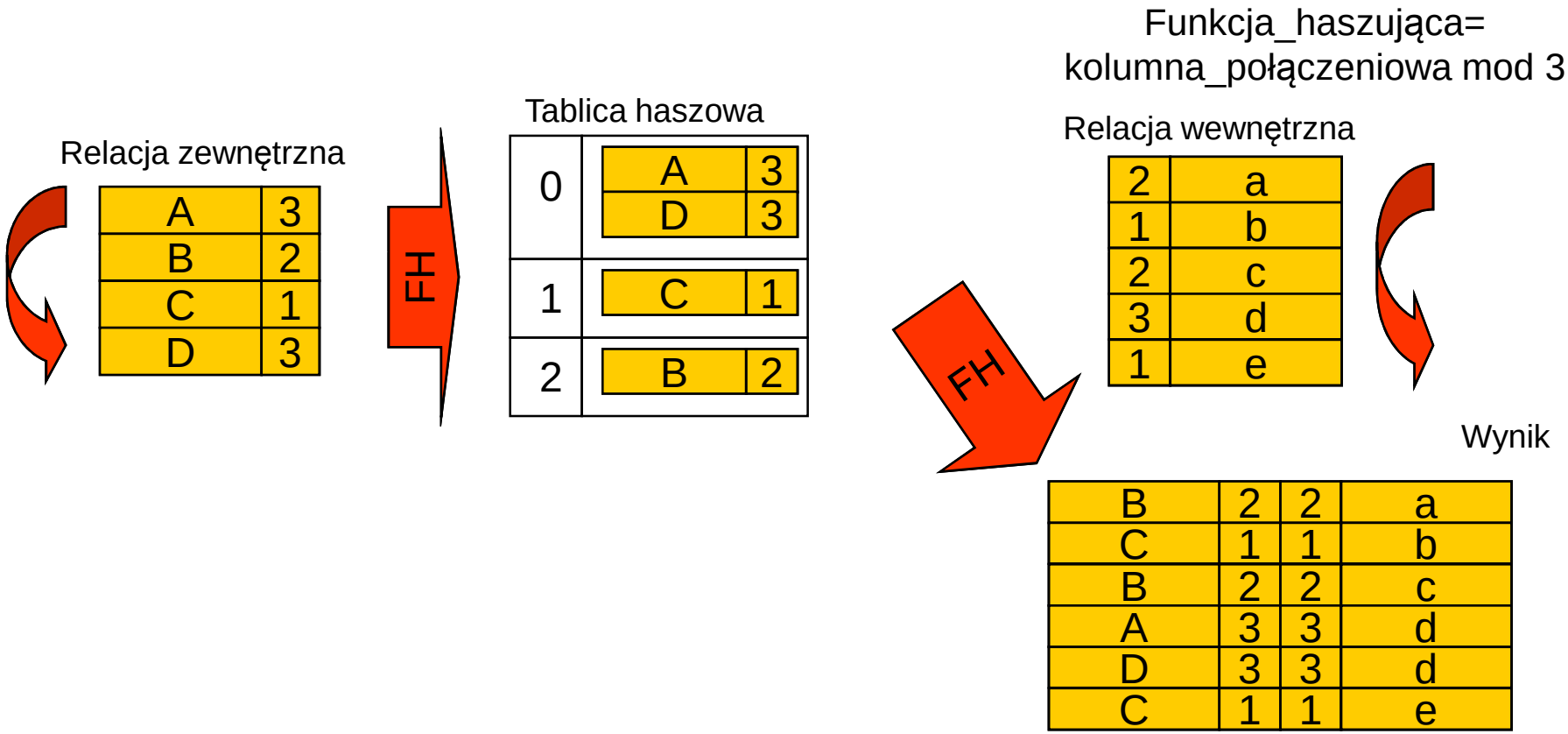
Relacja zewnętrzna – mniejsza z relacji, najlepiej, jeśli mieści się w pamięci.

W planie wykonania pierwsza relacja, z której zbudowano tablicę haszową:

```
HASH JOIN
    relacja_zewnętrzna
    relacja_wewnętrzna
```

Algorytm hash join (cd)

Algorytm:



Operacje sortowania

SORT ORDER BY – gdy w poleceniu wyrażenie ORDER BY.

```
SELECT * FROM zespoly
ORDER BY adres DESC;
```

SORT AGGREGATE – gdy w poleceniu wyliczana funkcji grupowa na całym zbiorze rekordów.

```
SELECT MAX(zatrudniony)
 FROM pracownicy;
```

SORT GROUP BY – gdy w poleceniu wyliczana funkcji grupowa dla kilku grup rekordów.

```
SELECT etat, AVG(placa_pod)
 FROM pracownicy GROUP BY etat;
```

Operacje sortowania (cd)

SORT UNIQUE – gdy w poleceniu użyto klauzuli DISTINCT.

```
SELECT DISTINCT etat
 FROM pracownicy;
```

SORT JOIN – przy wykonywaniu operacji połączenia wg algorytmu sort merge.

```
SELECT *
FROM pracownicy JOIN etaty ON placa_pod between
placa_min and placa_max;
```

Wskazówki

Wskazówki (ang. hints) umożliwiają określenie bezpośrednio w poleceniu następujących elementów pracy optymalizatora:

- **celu optymalizacji,**
- **ścieżki dostępu do danych,**
- **kolejności łączonych relacji przy operacji połączenia,**
- **sposobu realizacji połączenia**

Wskazówki umieszcza się w komentarzu bezpośrednio po klauzulach SELECT, INSERT, UPDATE, DELETE, przy czym pierwszym znakiem wskazówki musi być + (plus).

Uwaga! Błędnie sformułowana wskazówka nie powoduje błędu wykonania polecenia – jest ignorowana!

```
SELECT /*+ wskazówka */ nazwisko
FROM pracownicy WHERE id_prac=100;
```

Wskazówki (cd)

Wybór celu optymalizacji:

- ALL_ROWS – przepustowość,
- FIRST_ROWS – czas odpowiedzi,
- FIRST_ROWS(n) – czas odpowiedzi (pierwszych n krotek).

Sposób dostępu do danych:

- FULL (nazwa_relacji) – pełne przeglądnięcie relacji,
- INDEX (nazwa_relacji [nazwa_indeksu]) – dostęp za pomocą indeksu,
- INDEX_COMBINE (nazwa_relacji [nazwa_indeksu]) – dostęp za pomocą indeksu bitmapowego,
- INDEX_DESC (*nazwa_relacji [nazwa_indeksu]*) – dostęp za pomocą odwróconego przeszukiwania indeksu,
- INDEX_FFS (*nazwa_relacji [nazwa_indeksu]*) – dostęp za pomocą szybkiego przeszukania indeksu,
- INDEX_JOIN (*nazwa_relacji [nazwa_indeksu]*) – wykonanie połączenia indeksów,

Wskazówki (cd)

Sposób dostępu do danych (cd):

- **INDEX_SS (*nazwa_relacji [nazwa_indeksu]*) – dostęp za pomocą przeglądnięcia indeksu z pominięciem kolumn,**
- **NO_INDEX[_SS | _FFS](*nazwa_relacji [nazwa_indeksu]*) – zakazanie użycia indeksu,**

Kolejność łączenia relacji:

- **LEADING(*relacja1, relacja2, ...*) – określa zbiór relacji, które mają być łączone jako pierwsze,**
- **ORDERED – określa, że relacje mają być łączone w takiej kolejności, jak zostały wymienione w klauzuli FROM.**

Algorytm łączenia relacji:

- **USE_NL(*relacja_wewnętrzna ...) - p*ołączenie NESTED LOOPS**
- **USE_HASH (*relacja_wewnętrzna ...) - p*ołączenie HASH JOIN**
- **USE_MERGE (*relacja1 relacja2 ...) - p*ołączenie SORT MERGE**
- **NO_USE_NL(...), NO_USE_HASH(...), NO_USE_MERGE(...) – zakaz użycia odpowiedniego algorytmu.**

Wskazówki (cd)

Inne:

- **USE_CONCAT – wymuszenie zastąpienia zapytania z warunkiem złożonym z operatorem OR przez kilka zapytań, połączonych operatorem UNION_ALL,**
- **NO_EXPAND – zabronienie wykonania powyższej transformacji.**

Przykład użycia wskazówek:

```
SELECT /*+ LEADING(p e) USE_MERGE(p e z) */ *
FROM pracownicy p NATURAL JOIN zespoly z JOIN ETATY e
ON placa_pod between placa_min and placa_max
WHERE nazwa = 'ALGORYTMY';
```

SQLTRACE

Narzędzie służące do monitorowania działania aplikacji pod kątem efektywności

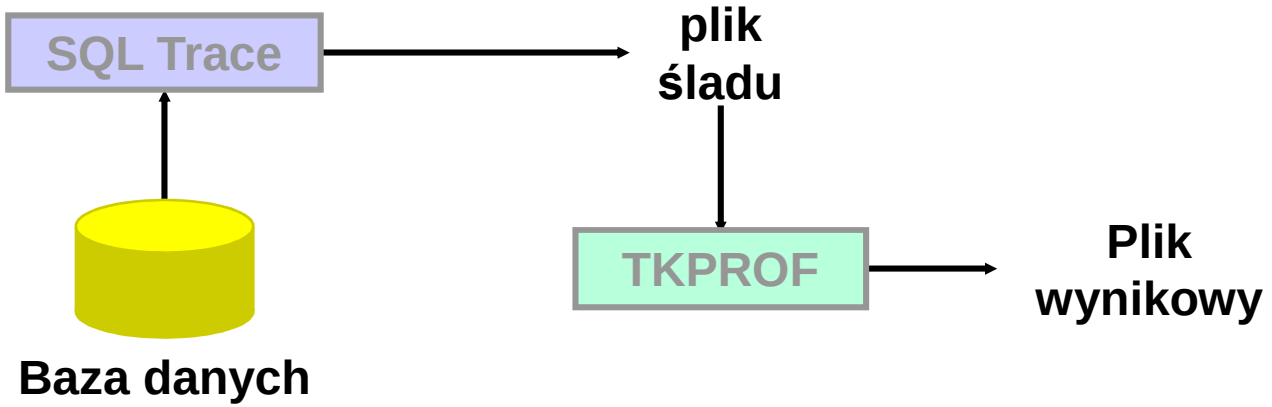
umożliwia zbieranie następujących statystyk:

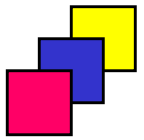
- **liczbę operacji *parse*, *execute* i *fetch* wykonanych na potrzebę realizacji poszczególnych poleceń SQL**
- **czas rzeczywisty i czas procesora poświęcony na wykonanie poszczególnych operacji,**
- **liczbę logicznych i fizycznych odczytów danych,**
- **liczbę przetworzonych rekordów,**
- **liczbę chybień w bufor zleceń SQL,**
- **nazwę użytkownika wykonującego zlecenie SQL,**
- **fakt realizacji polecenia zatwierdzenia lub wycofania transakcji**

wynik w postaci pliku śladowego umieszczonego w katalogu USER_DUMP_DEST

Jak używać SQL Trace

1. Ustawić parametry inicjalizacyjne.
2. Włączyć SQL Trace.
3. Uruchomić aplikację.
4. Wyłączyć SQL Trace i sformatować plik śladu.
5. Zinterpretować wyniki.





Uaktywnianie SQLTRACE

Na poziomie instancji za pomocą parametrów SQL_TRACE=true, TIMED_STATISTICS=TRUE, USER_DUMP_DEST= *nazwa_katalogu*

Na poziomie sesji

```
ALTER SESSION SET SQL_TRACE=true;
```

```
SELECT sid, serial#, username, osuser, machine
FROM v$session

SID  SERIAL# USERNAME   OSUSER      MACHINE
---  ------- ---------- ---------- -------
9    2       KADRY      Kowalski   venus
10   3       KADRY      Nowak       mars
exec DBMS_SYSTEM. SET_SQL_TRACE_IN_SESSION(9, 2, TRUE)
```

TKPROF

Narzędzie służące do formatowania zawartości pliku śladowego

```
TKPROF plik_śladowy[.trc] plik_wynikowy[.prf]
        [sys= yes|no]
        [aggregate=yes|no]
        [insert=nazwa_pliku]
        [record=nazwa_pliku]
```

```
TKPROF 34543.trc wynik.txt \
        sys= no \
        aggregate=no
```

Przykład pliku wynikowego TKPROF: bez indeksu

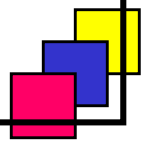
```
...
select max(cust_credit_limit)
from customers
where cust_city ='Paris'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.01       0.01          0          0          0           0
Execute      2      0.00       0.00          0          0          0           0
Fetch        4      0.08       0.07       1027       1044          0           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        8      0.09       0.08       1027       1044          0           2

Misses in library cache during parse: 2
Optimizer goal: CHOOSE
Parsing user id: 45

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=985 r=969 w=0 time=67351 us)
     77  TABLE ACCESS FULL CUSTOMERS (cr=985 r=969 w=0 time=67173 us)

********************************************************************************
```

Przykład pliku wynikowego TKPROF: z indeksem

```
…
select max(cust_credit_limit)
from   customers
where  cust_city = 'Paris'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0         59          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          0         59          0           1

Misses in library cache during parse: 1
Optimizer goal: CHOOSE
Parsing user id: 45

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=59 r=0 w=0 time=731 us)
     77   TABLE ACCESS BY INDEX ROWID CUSTOMERS (cr=59 r=0 w=0 time=661 us)
     77   INDEX RANGE SCAN CUST_CITY_IDX (cr=2 r=0 w=0 time=145 us)(object id 28482)

********************************************************************************
```