	Bazy Danych laboratorium	**Laboratorium BD2**

Pozyskiwanie informacji z jednej tabeli - opis przykładu

Tabela BD2_ZBIORCZA zawiera ewidencję wyników cyklu biegowych zawodów sportowych. W ramach jednego biegu zapisywana jest klasyfikacja zawodników, na którą składają się ewidencyjne dane zawodnika, punkty zdobyte w klasyfikacji generalnej i punkty zdobyte w klasyfikacji wiekowej.

Ewidencyjne dane zawodnika to numer ewidencyjny, imię i nazwisko, płeć, rok urodzenia, kategoria wiekowa, przynależność klubowa w postaci numeru klubu i jego nazwy. Zawodnicy są podzieleni na kategorie wiekowe według roku urodzenia, np. kategoria IV to mężczyźni urodzeni między 1957 i 1967 rokiem, a K-II to kobiety urodzone między 1977 i 1986 rokiem.

Punkty klasyfikacji generalnej – pierwszych pięćdziesięciu mężczyzn bez względu na wiek otrzymuje punkty według zasady: 1 miejsce – 50 pkt, 2 miejsce – 49 pkt, ……, 50 miejsce – 1 pkt, pozostali – pole nie jest wypełnione. Analogicznie jest wśród kobiet.

Punkty w klasyfikacji wiekowej (w kategoriach) – pierwszych pięćdziesięciu zawodników w danej kategorii wiekowej (np. II) otrzymuje punkty według podobnej zasady jak w klasyfikacji generalnej. Czyli w ramach tej klasyfikacji zarówno najlepszy zawodnik z kategorii II otrzymuje 50 pkt, jak również najlepszy zawodnik kategorii V też otrzymuje 50 pkt. Analogicznie jest wśród kobiet.

Na końcu cyklu składającego się z kilku biegów sumuje się punkty zdobyte przez zawodników w poszczególnych biegach i na tej podstawie ustala końcowe klasyfikacje: generalną i w kategoriach dla zawodników i generalną dla klubów powstającą na podstawie sumy punktów zdobytych przez zawodników danego klubu w klasyfikacji generalnej.

Poniżej przedstawiona jest struktura tabeli BD2_ZBIORCZA:

BD2_ZBIORCZA			
nr_zawodow	NUMBER(2)	<pk>	not null
nr_zawodnika	NUMBER(4)	<pk>	not null
imie	VARCHAR2(15)		null
nazwisko	VARCHAR2(30)		not null
plec	VARCHAR2(1)		not null
rok_urodzenia	NUMBER(4)		null
nr_klubu	NUMBER(3)		not null
klub	VARCHAR2(40)		not null
kategoria	VARCHAR2(6)		null
pkt_generalna	NUMBER(2)		null
pkt_kategorie	NUMBER(2)		null

Zdania tworzące tę tabelę zawarte są w skrypcie *lab_BD2_tab.sql*. Definicje kolumn *imie, rok_urodzenia* i *kategoria* dopuszczają wartości *null*. Nie jest to w pełni zgodne z rzeczywistością, ale zostało tak przygotowane w celu wykonania serii ćwiczeń na wartościach *null*.

Porównując skrypt *lab_BD2_tab.sql* z diagramem tabeli można zauważyć dodatkowe definicje (*constraint*) typu *check* służące do nadawania pewnych ograniczeń na wartości umieszczane w wybranych kolumnach. Kolumny *pkt_generalna* i *pkt_kategorie* mogą zawierać tylko wartości z przedziału [1..50] lub *null* - co wynika z regulaminu zawodów, a kolumna *plec* - jednoliterowe kody płci.

Warto zwrócić uwagę na złożony klucz główny. Każdy zawodnik może startować w kilku zawodach w sezonie i dlatego, aby jednoznacznie określić, o który wynik chodzi przy przeszukiwaniu bazy danych trzeba podać numer zawodów i numer zawodnika. I dopiero ta para wartości jednoznacznie wskaże wiersz w tabeli. Dodatkowo istotny jest sposób definiowana złożonego klucza głównego. Jeden ze sposobów został zaprezentowany w skrypcie *lab_BD2_tab.sql*, a drugim jest użycie zdania SQL *alter table....add constraint....primary key....*

Import danych do tabeli z pliku płaskiego typu wycinek (csv) przy użyciu Oracle SQL Developer

Do ładowania danych do tabel bazodanowych w środowisku Oracle używa się kilku narzędzi lub metod. Do najpopularniejszych narzędzi należą SQL Loader i Oracle Data Pump, które w tym materiale nie będą omawiane. Szczegółowe informacje można znaleźć w dokumentacji Oracle.

Zostanie, natomiast, omówiona metoda wypełniania tabel danymi przy użyciu Oracle SQL Developera i wbudowanej funkcjonalności Import Data.

Po założeniu tabeli (skrypt *lab_BD2_tab.sql*) należy postępować jak poniżej:

Wyświetlić strukturę utworzonej tabeli i poprzez Actions... przejść do importu danych (Import Data...) ustawiając:

- nazwę importowanego pliku: *lab_BD2_dane.csv*
- umieszczenie nagłówka pliku (Header): nie
- format: csv
- kodowanie: windows-1250 (Cp1250)
- separator pól (Delimiter): przecinek,
- znaki ograniczające dane tekstowe (Left i Right Enclosure): apostrof (')
- metodę importu: Insert Script
- wybór kolumn (Choose Columns): pozostawić bez zmian
- mapowanie kolumn (Match By): Position
- definicje kolumn: sprawdzić (ewentualnie zmienić) poprawność mapowania kolumn danych (Source Data Columns i Target Table Columns)

, na koniec obejrzeć szczegóły zdefiniowanych ustawień i zakończyć proces importu.

Zapoznać się z wygenerowanych skryptem umieszczonym automatycznie w Oracle SQL Developer i go wykonać, a następnie zatwierdzić transakcję zdaniem *commit*. Dodatkowo znaleźć w systemie operacyjnym folder: C:\Users\%USER\AppData\Local\Temp oraz plik o nazwie *Import-lab_BD2_dane-csv_…….sql* i wyświetlić jego zawartość.

Obejrzeć zawartość tabeli BD2_ZBIORCZA, szczególną uwagę zwrócić na kodowanie polskich znaków. Wygenerowany skrypt zapamiętać jako *lab_BD2_insert.sql*.

Wykonać zdanie SQL:

*select * from bd2_zbiorcza;*

W tabeli powinny znajdować się dane:

	NR_ZAWODOW	NR_ZAWODNIKA	IMIE	NAZWISKO	PLEC	ROK_URODZENIA	NR_KLUBU	KLUB	KATEGORIA	PKT_GENERALNA	PKT_KATEGORIE
148	4	173	Paulina	Stachurska	K	1989	380	UKS Bielany	K-I	47	50
149	1	182	Aleksandra	Dobrowolska	K	1975	348	ENTRE.PL Team	K-III	20	44
150	2	182	Aleksandra	Dobrowolska	K	1975	348	ENTRE.PL Team	K-III	30	49
151	3	182	Aleksandra	Dobrowolska	K	1975	348	ENTRE.PL Team	K-III	33	47
152	4	182	Aleksandra	Dobrowolska	K	1975	348	ENTRE.PL Team	K-III	25	40
153	1	202	Stefan	Karpisz	M	1947	67	Warszawa-Praga Południe	V	(null)	21
154	1	203	Grzegorz	Stańczyk	M	1955	41	Warszawa-Ursynów	V	(null)	23
155	1	204	Andrzej	Gromko	M	1957	50	Warszawa-Śródmieście	IV	(null)	22
156	1	205	Dawid	Żywek	M	1985	400	OKS Start Otwock	II	48	48

Wykonać drugie zdanie:

*select count (*) from bd2_zbiorcza;*

Powinien pojawić się wynik liczbowy wskazujący ile wierszy zostało wpisanych do tabeli:

	COUNT(*)
1	838

W przypadku niepowodzenia należy proces ładowania danych powtórzyć.

Zdanie SELECT – podstawy

W swej najprostszej postaci zdanie SELECT wymaga tylko wskazania tabeli, w której znajdują się oczekiwane dane. Ta najprostsza forma wygląda jak poniżej:

select * *from* nazwa_tabeli

np.: *select * from bd2_zbiorcza;*

Zbiorem wynikowym powyższego zdania będą wszystkie dane zawarte w tabeli czyli wszystkie wiersze i wszystkie kolumny.

Podstawowa składnia tego zdania jest następująca:

select kolumn1, kolumn2, ……, kolumna_n *from* nazwa_tabeli

np.: *select nr_zawodow, imie, nazwisko from bd2_zbiorcza;*

NR_ZAWODOW	IMIE	NAZWISKO
1	Krystyna	Sawicka
1	Anna	Jasek
1	Ewa	Tkaczyk
2	Ewa	Tkaczyk
4	Ewa	Tkaczyk
2	Wanda	Misiaczek
3	Wanda	Misiaczek
1	Anna	Kolarska
1	Maria	Stańczak
3	Maria	Stańczak
1	Agnieszka	Teichert

…..

Uwagi:

1. Kolejność nazw kolumn jest dowolna i może być podzbiorem wszystkich kolumn stanowiących strukturę tabeli czyli powyższe zdanie może wyglądać następująco:

   select imie, nazwisko, nr_zawodow from bd2_zbiorcza;

2. Zawężenie wyboru kolumn może prowadzić do powtarzalności wierszy powodując nadmiarowość zbioru wynikowego.

   Zdanie:

   select nazwisko, imie from bd2_zbiorcza;

da wynik:

NAZWISKO	IMIE
Sawicka	Krystyna
Jasek	Anna
Tkaczyk	Ewa
Tkaczyk	Ewa
Tkaczyk	Ewa
Misiaczek	Wanda
Misiaczek	Wanda
Kolarska	Anna
Stańczak	Maria
Stańczak	Maria

.....

Aby uzyskać unikatową listę nazwisk należy użyć konstrukcji:

*select **distinct** nazwisko, imie from bd2_zbiorcza;*

Wtedy wynik będzie następujący:

NAZWISKO	IMIE
Kolarska	Anna
Teichert	Agnieszka
Gliniewicz	Elżbieta
Zielona	Magdalena
Dobosz	Elżbieta
Hirska	Ela
Markowska	Barbara
Paczkowska	Olga
Butkiewicz	Emilia

.....

Warto zaznaczyć, że klauzula *distinct* nie odnosi się do kolumny, przed którą bezpośrednio stoi, lecz do układu wszystkich kolumn na liście *select*. Błędne jest zatem zdanie:

*select **distinct** nazwisko, **distinct** imie from bd2_zbiorcza;*

Filtrowanie danych wynikowych za pomocą klauzuli WHERE

Fraza WHERE umożliwia definiowanie jednego lub wielu warunków, które muszą być spełnione przez każdy wiersz tabeli kwalifikowany do ostatecznego zbioru wynikowego. W ramach tej frazy można używać operatorów porównania (=, <, >, <=, >=, <>), operatorów logicznych (AND, OR, NOT) oraz dodatkowo takich operatorów jak BETWEEN, ANY, SOME, IN oraz LIKE.

Zastosowanie operatorów porównania:
np.

```
select distinct imie, nazwisko, rok_urodzenia
        from bd2_zbiorcza
        where nr_klubu = 10;
```

zwraca dane zawodników z klubu o identyfikatorze 10, którzy chociaż raz wystartowali w cyklu zawodów:

IMIE	NAZWISKO	ROK_URODZENIA
Krzysztof	Mróz	1973
Karol	Maciuszek	1986
Jarosław	Nizak	1970

Zwrócić należy uwagę, że bez klauzuli *distinct* wynik będzie inny:

IMIE	NAZWISKO	ROK_URODZENIA
Karol	Maciuszek	1986
Karol	Maciuszek	1986
Karol	Maciuszek	1986
Krzysztof	Mróz	1973
Jarosław	Nizak	1970

gdyż jeden zawodnik może startować kilka razy w cyklu zawodów.

Zastosowanie operatorów logicznych:

np.

```
……..
where plec = 'M' and rok_urodzenia > 1980        (iloczyn logiczny)
```

oznacza wybór wierszy dotyczących mężczyzn urodzonych po 1980 roku,

```
……..
where plec = 'M' or rok_urodzenia < 1980         (suma logiczna)
```

oznacza wybór wszystkich wierszy dotyczących mężczyzn oraz dodatkowo kobiet urodzonych przed 1980 rokiem.

Przykład złożonego zdania logicznego:

```
…….
where not
(       (nr_klubu < 4 or nr_klubu = 20)
 and
        (rok_urodzenia > 1980 or rok_urodzenia <= 1960)
)
```

Należy zwrócić uwagę na obowiązującą kolejność wykonywania działań z operatorami *and* i *or* oraz konieczność używania nawiasów w celu jej zmiany.[1]

Na podstawie prawa de Morgana[2] można powyższy warunek przekształcić następująco:

```
……
where
        not (nr_klubu < 4 or nr_klubu = 20)
or
        not (rok_urodzenia > 1980 or rok_urodzenia <= 1960)
```

Szczegółową analizę tego przykładu pozostawia się czytelnikowi. Werbalnie można stwierdzić, że na podstawie drugiego wariantu powyższego przykładu do zbioru wynikowego zostaną zakwalifikowane wiersze, które albo nie spełniają pierwszego warunku (nr_klubu) albo nie spełniają drugiego warunku (rok_urodzenia). Innymi słowy (na podstawie wariantu pierwszego) do zbioru wynikowego nie zostaną zakwalifikowane wiersze, które spełniają pierwszy i drugi warunek jednocześnie.

Zastosowanie pozostałych operatorów:

Operator **BETWEEN**:

Umożliwia określenie przedziału, do którego muszą należeć akceptowane wartości:
np.:

```
……
where nr_zawodow between 2 and 4
```

jest równoważne frazie:

```
……
where nr_zawodow >= 2 and nr_zawodow <= 4
```

A fraza:

```
……
where nr_zawodow  not between 2 and 4
```

lub jej równoważna:

```
……
where not nr_zawodow  between 2 and 4
```

jest zaprzeczeniem poprzedniej frazy.

Operator **IN**:

Sprawdza, czy wartość wskazanej kolumny odpowiada którejś z wartości wymienionej na liście argumentów tego operatora.

np.:

```
……
where kategoria in ('K-II', 'K-IV', 'K-VI')
```

[1] Szczególnie jest to istotne w przypadku używania operatora logicznego o*r* w połączeniu z operatorem *and*.
[2] Pierwsze prawo De Morgana - prawo zaprzeczania koniunkcji: not (p and q) <=> not p or not q, gdzie p i q oznaczają zdania logiczne.

Do zbioru wynikowego zostaną zakwalifikowane wiersze, w których pole kategoria przyjmuje jedną z trzech wartości w nawiasie.

Operator **LIKE**:

Umożliwia stosowanie symboli wieloznacznych w warunkach przeszukujących pola znakowe. Symbol wieloznaczny nie definiuje konkretnego znaku, a jedynie dopasowuje do zdefiniowanego wzorca dowolny znak.

Na ogół stosuje się dwa symbole wieloznaczne:

% -dowolna liczba znaków,
_ -jeden znak,

np.

…….
where nazwisko *like* 'Ko%'

oznacza wybór tych wierszy tabeli, w których nazwisko zaczyna się na Ko czyli np. Kos lub Kowalski,
ale:

…….
where nazwisko *like* 'Ko_'

oznacza wybór tych wierszy, w których nazwisko jest trzyliterowe i zaczyna się na Ko czyli np. Kot, Kos, ale nie Kowalski,
a np.:

……..
where imie like '%ś%'

oznacza wybór wierszy, w których imię zawiera w sobie literę 'ś'.

Zarządzanie wartościami NULL

Jak już wcześniej wspomniano, w przypadku, gdy do jakiegoś pola w tabeli nie wprowadzono określonej wartości to pole to ma ustawioną wartość *null*.

W przypadku wprowadzania wiersza do tabeli zdaniem *insert* można to osiągnąć w dwojaki sposób. Pierwszy polega na wyspecyfikowaniu wszystkich kolumn w tabeli a na pozycjach wartości w odpowiednim miejscu wpisanie *null* lub nie wpisanie niczego:

```
insert into tabela (kol1, kol2, kol3)
        values    (wartość_1, null, wartość_3);
```

lub

```
insert into tabela (kol1, kol2, kol3)
        values    (wartość_1, ' ' , wartość_3);  -- pusty ciąg znaków na pozycji drugiej
```

Natomiast zdanie:

```
insert into tabela (kol1, kol2, kol3)
        values    (wartość_1, '  ' , wartość_3);  -- spacja na pozycji drugiej
```

wprowadza do kolumny spację a nie *null*.

Drugi sposób polega na uwzględnieniu w specyfikacji zdania *insert* tylko tych kolumn, do których wprowadza się dane:

```
insert into tabela (kol1, kol3)
        values    (wartość_1, wartość_3); -- kol2 istnieje w strukturze tabeli
```

Należy mieć na uwadze fakt, że realizacja powyższego zdania jest możliwa tylko pod warunkiem, że w definicji tabeli dopuszcza się możliwość występowania *null* w kol2 czyli dla tej kolumny nie występuje *not null*.

Chcąc ustawić jakieś pole lub kilka pól tej samej kolumny na wartość *null* można to zrobić na przykład tak:

```
update tabela
        set kol2 = null, kol4 = null
where kol1 like 'A%';
```

Null posiada następujące właściwości:

- null <> dowolna_określona_wartość
- null not > dowolna_określona_wartość
- null not < dowolna_określona_wartość
- null <> null
- null + dowolna_określona_wartość = null
- null and true = null
- null and false = false
- null or true = true
- null or false = null

Chcąc skasować określone wiersze w tabeli należy pamiętać o podstawowej własności *null* wyszczególnionej powyżej na pierwszym miejscu czyli *null* nie równa się "niczemu".

Zatem zdanie:

```
delete tabela
where kol2 = null;
```

nie skasuje żadnego wiersza w tabeli, pomimo faktu, że w kol2 znajdują się wartości *null*.
Kasowanie będzie skuteczne, gdy zamiast znaku równości użyte zostanie słowo *is*.

```
delete tabela
where kol2 is null;
```

Należy zatem pamiętać, aby przy przeszukiwaniu tabeli zdaniem *select* we frazie *where* również używać filtru ...***kolumna is null***, a nie ...*kolumna = null*.

Porządkowanie zbioru wynikowego przy pomocy frazy ORDER BY:

Standardowo zbiór wynikowy nie jest uporządkowany w żaden sposób. O kolejności wierszy decyduje system zarządzania bazą danych (SZBD). Na ogół jest zgodny z kolejnością wpisywania wierszy do tabeli (gdy brak jest zdefiniowanego klucza głównego) lub zgodnie z kluczem głównym. W przypadku, gdy zachodzi potrzeba uporządkowania zbioru według zadanych kryteriów należy użyć frazy *order by*, którą umieszcza się na końcu zdania *select*.

np. zdanie:

```
select  *
from bd2_zbiorcza
where nr_zawodow = 1
order by pkt_generalna;
```

zwróci zbiór wyników zawodów o identyfikatorze 1 posortowanych rosnąco (od 1 do 50 pkt) według zdobytych punktów. Należy zwrócić uwagę, że wiersze zawierające *null* w kolumnie *pkt_generalna* są umieszczone na końcu zbioru.

Chcąc umieścić wiersze z wartościami NULL w kolumnie klucza sortowania na początku zbioru wyników należy użyć konstrukcji:

```
.....
order by pkt_generalna nulls first;
```

Domyślnie ustawiony jest rosnący kierunek sortowania (*ascending* – skrót *asc*). W przypadku zmiany kierunku sortowania na malejący należy użyć klauzuli *desc* (*descending*),
np.:

```
.....
order by pkt_generalna desc;
```

Uwagi końcowe:

1. Istnieje możliwość łączenia ze sobą kilku kolumn znakowych w zdaniu *select*. Na przykład chcemy przedstawić imię i nazwisko pisane łącznie ze standardowym odstępem w postaci jednej spacji. W takim przypadku można skorzystać z operatora konkatenacji (łączenia ciągów znakowych) w sposób podany poniżej:

```
select distinct nazwisko || '  ' || imie, rok_urodzenia
from bd2_zbiorcza
order by rok_urodzenia desc;
```

NAZWISKO||' '||IMIE	ROK_URODZENIA
Dawidzka Katarzyna	1999
Bieliński Marcin	1993
Andres Anna	1992
Cygler Tomasz	1992
Banaszek Marta	1991
Kuczkowski Mateusz	1991
Chorążewicz Bartosz	1990
Fitzgibbon Sławomir	1990
Gajda Monika	1990
Jarosiewicz Bartek	1990

.....

2. W zdaniu *select* można zmieniać nazwy kolumn w zbiorze wyników. Na przykład łatwo zauważyć, że w powyższym przykładzie nazwa kolumny nie jest zbyt atrakcyjna. Wystarczy zastosować alias jak poniżej:

```
select distinct nazwisko || '  ' || imie as Nazwisko, rok_urodzenia as Rocznik
    from bd2_zbiorcza
    order by rok_urodzenia desc;
```

3. Nazwa kolumny w zbiorze wynikowym może składać się z kilku wyrazów oddzielonych spacjami. W takim przypadku należy pamiętać o użyciu cudzysłowów, jak w poniższym przykładzie (ale nie apostrofów):

```
select distinct nazwisko || '  ' || imie as "Nazwisko i imię", rok_urodzenia as "Rocznik"
    from bd2_zbiorcza
    order by rok_urodzenia desc;
```

Nazwisko i imię	Rocznik
Dawidzka Katarzyna	1999
Bieliński Marcin	1993
Andres Anna	1992
Cygler Tomasz	1992
Banaszek Marta	1991
Kuczkowski Mateusz	1991
Chorążewicz Bartosz	1990
Fitzgibbon Sławomir	1990
Gajda Monika	1990
Jarosiewicz Bartek	1990

.....

4. Słowo *as* w definicji aliasu można pominąć.

5. Aby posortować zbiór wyników według kilku kluczy sortowania należy użyć poniższej konstrukcji frazy *order by*:

   .....
   order by pierwszy_klucz_sortowania [*desc*], drugi_klucz_sortowania [*desc*],.....

   , przy czym kierunek sortowania należy ustalić dla każdego klucza oddzielnie.

6. We frazie *order by* jako klucze sortowania umieszcza się, na ogół, kolumny, które równocześnie występują we frazie *select*, na przykład:

```
select  nazwisko, imie, rok_urodzenia
from bd2_zbiorcza
order by rok_urodzenia;
```

Możliwa jest inna konstrukcja polegająca na sortowaniu zbioru wyników według kolumny nie występującej na liście *select*, na przykład:

```
select  nazwisko, imie
from bd2_zbiorcza
order by rok_urodzenia;
```

Ale powyższa konstrukcja nie jest możliwa w przypadku użycia klauzuli *distinct*, na przykład:

```
select  distinct nazwisko, imie
from bd2_zbiorcza
order by rok_urodzenia;
```

Jednym ze sposobów rozwiązania tego problemu jest użycie funkcji agregującej[3], np.:

```
select  nazwisko, imie
from bd2_zbiorcza
group by nazwisko, imie
order by max(rok_urodzenia);
```

Klauzulę *distinct* można zastąpić odpowiednio skonstruowaną frazą *group by*:

```
select  distinct nazwisko, imie
    from bd2_zbiorcza;
```

=

```
select  nazwisko, imie
      from bd2_zbiorcza
    group by nazwisko, imie;
```

[3] Funkcje agregujące zostaną omówione w Laboratorium BD4.

Zadania do samodzielnego wykonania

Działania na tabeli BD2_ZBIORCZA_TEMP:

1. Zmodyfikować plik lab_BD2_dane.csv poprzez wprowadzenie na początku nazw kolumn (zgodnych z nazwami kolumn w tabeli lub też nie) i przeprowadzić proces ładowania danych do tabeli BD2_ZBIORCZA_TEMP, która ma mieć dokładnie taką samą strukturę jak tabela BD2_ZBIORCZA.

2. W tabeli BD2_ZBIORCZA_TEMP zmienić kolumnę *plec* tak, aby zawierała wartości odpowiednio *Kobieta* lub *Mężczyzna* zamiast dotychczasowych *K* i *M*, a nazwę kolumny zmienić na *plec_nazwa*. Po przeprowadzeniu niezbędnych modyfikacji zdaniem *update* przetestować ich skuteczność (w kolumnie określającej płeć mogą być tylko wartości *Kobieta* lub *Mężczyzna*).

3. W tabeli BD2_ZBIORCZA_TEMP zmodyfikować kolumnę *pkt_kategorie* tak, aby konieczne było wprowadzenie do niej wartości różnej od null (czyli, aby otrzymała atrybut not null). W miejsce dotychczas występujących wartości null wstawić 0.

Działania na tabeli BD2_ZBIORCZA:

Zaprojektować zdania SQL generujące zbiory wynikowe przy następujących założeniach:

1. Mężczyźni urodzeni po 1975 roku i należący do klubów o numerach między 3 i 10. Uporządkowanie – według numerów klubów i roczników (malejąco).

2. Zawodnicy, których nazwisko kończy się na ‘ski’ lub ‘ska’ należący do jednej z kategorii (I, II, K-II, K-V). Uporządkowanie – na początku kobiety, a potem mężczyźni, a w ramach płci alfabetycznie.

3. Zawodnicy należący do klubu ‘KB Gymnasion Warszawa', którzy nie zdobyli punktów w klasyfikacji generalnej. Uporządkowanie - według kategorii wiekowej i nazwisk.

4. Zawodnicy, którzy nie zdobyli punktów w klasyfikacji kategoriami i należą do kategorii (II, III, V). Uporządkowanie – według kategorii, klubów i aliasu stanowiącego połączenie Nazwiska i imienia.