Bazy Danych laboratorium

Laboratorium **BD6**

Na potrzeby zajęć zostanie wykorzystany model bazy danych opisany i zaimplementowany w ramach Laboratorium BD3.

Poniżej omówione zostaną dwie zaawansowane techniki tworzenia raportów przy pomocy zdania select - zapytania krzyżowe (pivot query) mające zastosowanie w analityce biznesowej oraz podzapytania (subquery) zwiększające dynamikę opracowywanych zapytań.

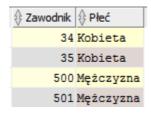
Tworzenie i zastosowanie wyrażenia CASE

Wyrażenie CASE w języku SQL zastępuje instrukcję warunkową if...then...else...endif; i może mieć jedną z dwóch postaci:

Pierwsza postać:

```
case kolumna_w_tabeli
              when wartość_1 then wynik_1
              when wartość_2 then wynik_2
               when wartość n then wynik n
       else wynik inny
       end;
np. zdanie:
       select nr_zawodnika as "Zawodnik",
              ( case plec
                      when 'K' then 'Kobieta'
                      when 'M' then 'Meżczyzna'
               else 'Brak danych'
               end) as "Płeć"
       from bd3_zawodnicy
       where nr_zawodnika in (34, 35, 500, 501);
```

tworzy zbiór wynikowy:



Druga postać:

```
case
       when zdanie_logiczne_1 then wynik_1
       when zdanie_logiczne _2 then wynik_2
       ......
       when zdanie_logiczne _n then wynik_n
else wynik inny
end:
```

np. zdanie:

```
select nr_zawodnika as zawodnik, rezultat_min, rezultat_sek,

( case

when rezultat_min < 35 then 'Mistrzowska'
when rezultat_min < 38 then 'I klasa'
when rezultat_min < 40 then 'II klasa'
else Null
end ) as klasa_sportowa

from bd3_wyniki
where nr_zawodow = 1 and nr_zawodnika in ( 616, 636, 577, 34 )
order by rezultat_min, rezultat_sek;
```

daje poniższe wyniki:

616	33	38	Mistrzowska
577	36	0	I klasa
636	39	11	II klasa
34	51	52	(null)

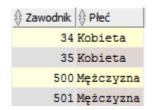
W języku SQL w wydaniu Oracle pierwsza postać wyrażenia *case* może być zastąpiona wyrażeniem *decode*, którego postać jest następująca:

```
..., decode (kolumna,
wartość_1, wynik_1,
wartość_2, wynik_2,
....
wartość_n wynik_n,
wynik_inny), ...
```

Porównywane są wartości pola w kolumnie z kolejnymi wartościami *wartość_...* i w przypadku równości zwracany jest odpowiedni *wynik_...* . Jeśli *wynik_inny* jest pominięty to zwracany jest *NULL*.

Powyższy przykład z pierwszą postacią case może być przedstawiony tak:

i daje ten sam wynik:



Wyrażenia case lub decode można użyć do budowy zdań SQL umożliwiających otrzymywanie raportów krzyżowych. Przykładem niech będzie poniższy raport pokazujący liczbę zawodników startujących w poszczególnych zawodach według kategorii wiekowych:

```
select z.nr_kategorii as "Nr kategorii", nazwa_kategorii as "Kategoria",
  sum ( decode ( nr_zawodow , 1 , 1 , 0 )) as "Zawody I",
sum ( decode ( nr_zawodow , 2 , 1 , 0 )) as "Zawody II",
  sum ( decode ( nr_zawodow , 3 , 1 , 0 )) as "Zawody III",
  sum ( decode ( nr_zawodow , 4 , 1 , 0 )) as "Zawody IV",
  count(*) as "Łącznie"
from bd3_zawodnicy z join bd3_wyniki w on z.nr_zawodnika = w.nr_zawodnika
 join bd3_kategorie k on k.nr_kategorii = z.nr_kategorii
group by z.nr_kategorii, nazwa_kategorii
union
select null, 'Razem',
  sum ( decode ( nr_zawodow , 1 , 1 , 0 )),
  sum ( decode ( nr_zawodow , 2 , 1 , 0 )),
  sum ( decode ( nr_zawodow , 3 , 1 , 0 )),
  sum ( decode ( nr_zawodow , 4 , 1 , 0 )),
  count(*)
from bd3 zawodnicy z join bd3 wyniki w on z.nr zawodnika = w.nr zawodnika
  join bd3 kategorie k on k.nr kategorii = z.nr kategorii
```

order by 1;

♦ Nr kategorii						∳ Łącznie
2	II	8	3	3	5	19
3	III	54	28	29	31	142
4	IV	56	45	38	61	200
5	V	40	31	27	29	127
6	VI	45	36	36	30	147
7	VII	10	10	5	3	28
8	VIII	6	5	4	2	17
9	IX	3	1	0	1	5
10	X	0	1	0	0	1
20	K-I	0	0	1	0	1
21	K-II	7	5	3	3	18
22	K-III	17	5	7	6	35
23	K-IV	12	7	9	14	42
24	K-V	7	6	10	5	28
25	K-VI	8	8	4	6	26
26	K-VII	1	0	1	0	2
(null)	Razem	274	191	177	196	838

Należy zwrócić uwagę, że raport posiada dwa podsumowania: według zawodów i według kategorii wiekowych.

Uwagi:

1. Przy budowie raportów krzyżowych należy pamiętać, że każda nowa kolumna powstaje poprzez użycie niezależnego wyrażenia *decode* oraz funkcji agregującej,

- 2. Podsumowanie kolumn (stopka raportu) powstaje poprzez połączenie *union*, w którym liczba kolumn musi być taka sama jak w pierwszym zdaniu *select*,
- 3. Klucze sortowania odnoszą się do pierwszego zdania select.
- 4. W drugim zdaniu *select* na liście brak jest kolumn z tabeli (znajduje się tam tylko *Null* i napis 'Razem') i dlatego mimo zastosowania funkcji agregującej nie ma potrzeby użycia klauzuli *group by.*
- 5. Ta konstrukcja ma wadę polegającą na statycznym układzie kolumn. Przedstawione powyżej przykłady zapytania krzyżowego realizującego raport liczby startujących zawodników w podziale na kategorie wiekowe i numery zawodów zawierają stałą liczbę kolumn obrazującą analizę dotyczącą czterech zawodów. Zapytanie to nie będzie uwzględniało liczności zawodników w przypadku zarejestrowania wyników kolejnego piątego i dalszych zawodów.
- 6. Język SQL został poszerzony o frazę *pivot*, przy pomocy której możliwe jest wygenerowanie raportu krzyżowego. Poniżej przedstawiony został przykład użycia tej frazy przy tworzeniu raportu obrazującego sumaryczne zdobycze punktowe klubów w klasyfikacji generalnej:

⊕ Nr klubu ⊕ Klub	⊕ Zawody I	A Zawody II		
1 Allianz Warszawa	341	268	395	421
2 KB Orientuz Warszawa	(null)	55	51	22
3 KB Gymnasion Warszawa	498	422	408	435
4 Legia Warszawa	77	51	75	76
5 Flota Gdynia	(null)	(null)	(null)	(null)
6 KB Promyk Ciechanów	153	120	158	137
7 KB Pułaski Strong Warka	(null)	(null)	(null)	(null)
8 AZS SGGW Warszawa	112	83	43	(null)
9 Grunwald Poznań	(null)	(null)	(null)	(null)
10 AZS Uniwersytet Warszawski	71	95	48	80
11 KB Amator Mińsk Mazowiecki	(null)	(null)	(null)	(null)
12 Śląsk Wrocław	(null)	(null)	(null)	(null)
13 KB Promyk Ursus	(null)	39	(null)	(null)
14 AZS-AWF Warszawa	41	61	33	25

...

Powyższe zdanie ma wadę polegającą na statycznej konstrukcji z uwagi na liczbę zawodów oraz dodatkowo na brak możliwości zrealizowania podsumowań według obu wymiarów (zawody i kluby).

7. Współczesne środowiska programowe wyspecjalizowane w tworzeniu raportów na podstawie danych z bazy (na przykład Jasper Reports) mają "zaszytą" w sobie powyższą metodę. W takim przypadku wystarczy opracować standardowe zdanie select, na podstawie którego zostanie wygenerowany raport krzyżowy, na przykład:

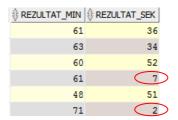
Na jego podstawie jest możliwe wygenerowanie raportu (na przykład w formacie pdf) tożsamego z przedstawionym powyżej i zawierającym podsumowania według obu wymiarów.

Zadanie do samodzielnego wykonania:

1. Zdanie select:

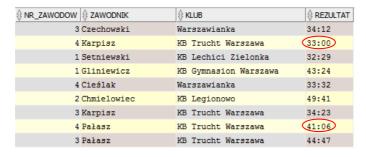
select rezultat_min, rezultat_sek
from bd3_wyniki;

tworzy poniższy zbiór:



-- brak 0 prowadzącego dla wartości < 10

Należy opracować perspektywę bd3_wyniki_zawodow tworzącą zbiór o postaci:



, a następnie w oparciu o nią tworzyć zbiory wyników dla poszczególnych zawodów wykorzystując zmienną wiązania (:Nr_zawodow).

Należy wykorzystać konstrukcję case oraz funkcję to_char w połączeniu z działaniem konkatenacji.

Zapytania zagnieżdżone (podzapytania) - podstawy

Jednym ze sposobów optymalizacji zdań SQL jest odpowiednie budowanie ich składni. Składnia języka daje kontrolę nad kolejnością przeprowadzanych złączeń przez odpowiednie konstruowanie fraz from i where.1

Drugim mechanizmem umożliwiającym otrzymanie identycznych rezultatów bez konieczności wykonywania złączeń (lub ich ograniczania) są zapytania zagnieżdżone. Stosowanie podzapytań dynamizuje zdanie SQL, gdyż w jednej konstrukcji można zawrzeć ciąg kilku zdań SQL w celu osiągnięcia tego samego wyniku.

Zapytanie zagnieżdżone to konstrukcja zawierająca pełne zdanie select umieszczone w innym zdaniu select. Rezultat wewnętrznego zdania select (podzapytania) staje się zbiorem wejściowym dla wyrażenia zewnętrznego.

Postać ogólna zdania zagnieżdżonego przedstawia się następująco:

```
select kolumna_1, kolumn_2,.....
       from tabela1
       where kolumna N operator ( select kolumna M
                                      from tabela2
                                      where .....);
```

Obowiązują następujące zasady budowania zapytania zagnieżdżonego:

- 1. Wewnętrzne zdanie ujęte jest w nawiasy () i zawsze zaczyna się klauzulą select;
- 2. Wewnętrzne zdanie musi zwracać albo wielkość skalarną (pojedyncza liczba, napis lub data) lub wektorową czyli wartości z jednej kolumny,²
- 3. Operatorami mogą być operatory używane do tej pory we frazie where (=, <, >, itp), jak również operatory mnogościowe, takie jak: in, any, all (łączone lub nie z operatorem not).

Przykładowo zdanie:

```
select nazwisko, imie, to char ( data urodzenia, 'DD-MM-YYYY') "Data urodzenia"
from bd3 zawodnicy
where data urodzenia = ( select min ( data urodzenia )
                         from bd3 zawodnicy
                         where plec = 'M')
and plec = 'M';
```

nakazuje znaleźć dane najstarszego zawodnika (lub zawodników) płci męskiej:

```
♠ NAZWISKO | ♠ IMIE | ♠ Data urodzenia |

1 Werthein
              Edmund 08-03-1926
```

Zdanie wewnętrzne zwraca konkretną wartość (jedną datę) i dlatego w zdaniu zewnętrznym można użyć operatora porównania (=).

opr. Józef Woźniak

6

¹ Optymalizacja zdań SQL nie będzie omawiana w tym materiale.

² Daje się zauważyć zmianę w tym zakresie, np. w systemie Oracle podzapytanie może zwracać wartości z kilku kolumn.

Ale poniższe zdanie:

```
select nazwisko, imie
from bd3_zawodnicy
where nr_zawodnika = ( select nr_zawodnika
                      from bd3 wyniki
                      where nr_zawodow = 2);
```

nie może być zrealizowane:

```
Error report -
ORA-01427: jednowierszowe podzapytanie zwraca więcej niż jeden wiersz
```

, gdyż zdanie wewnętrzne zwraca zbiór wartości, a nie jedną liczbę. W tym przypadku trzeba zastosować operator mnogościowy.

Istnieją dwa typy zapytań zagnieżdżonych: skorelowane i nieskorelowane.

W zapytaniu nieskorelowanym interpreter poleceń SQL potrafi zakończyć przetwarzanie wewnętrznego zdania select przed przystąpieniem do analizy zdania zewnętrznego.

W powyższym przykładzie dotyczącym znajdowania najstarszych zawodników płci męskiej w ewidencji kolejność wykonywania zdania jest następująca:

- 1. Wykonywane jest jednorazowo zdanie wewnętrzne, efektem czego jest konkretna wartość wskazująca na najstarszą datę urodzenia (1926/03/08),
- 2. Obliczona wielkość wstawiana jest do zdania zewnętrznego i powstaje warunek:

```
where data_urodzenia = ' 26/03/08 ';
```

3. Realizowane jest zdanie zewnętrzne.

Z kolei w zapytaniu skorelowanym interpreter nie jest w stanie wykonać zapytania wewnętrznego bez informacji pochodzących ze zdania zewnętrznego.

Przykładowo zdanie:

```
select nr_zawodnika, nr_klubu
from bd3 zawodnicy z
where extract ( year from sysdate ) - extract ( year from data urodzenia ) >
( select avg ( extract ( year from sysdate ) - extract ( year from data_urodzenia ))
  from bd3 zawodnicy x
 where x.nr_klubu = z.nr_klubu );
```

realizuje zestawienie tych zawodników, których wiek jest większy od średniej wieku wszystkich zawodników jego klubu.

Kolejność wykonywania się tego zdania jest następująca:

- 1. Na podstawie zdania zewnętrznego pobierany jest pierwszy wiersz z tabeli BD3_ZAWODNICY o aliasie z,
- 2. Numer klubu z tego wiersza (z.nr klubu) wstawiany jest do zdania wewnętrznego i na tej podstawie wyliczana jest średnia wieku zawodników jednego klubu,
- 3. Wynik ten zwracany jest do zdania zewnętrznego i na podstawie warunku we frazie where następuje, bądź nie, kwalifikacja pierwotnie odczytanego wiersza do zbioru wynikowego,

4. Czytany jest kolejny wiersz z tabeli BD3 ZAWODNICY z aliasem z i cykl się powtarza.

Zapytania skorelowane wymagaja wielokrotnego wykonania wewnetrznego select i dlatego sa mało wydaine.

Wada ta nie są obciążone zapytania nieskorelowane, które można wykorzystywać do zastępowania złaczeń i przyspieszania operacji na bazie danych, na przykład:

Zdanie:

```
select nr_zawodnika, z.nr_klubu
from bd3 zawodnicy z
    join bd3_kluby k on z.nr_klubu = k.nr_klubu
where nazwa_klubu like ' %AZS Uniwersytet% ';
```

można zastąpić zdaniem:

```
select nr_zawodnika, nr_klubu
from bd3_zawodnicy
where nr_klubu = ( select nr_klubu
                  from bd3 kluby
                  where nazwa_klubu like ' %AZS Uniwersytet% ');
```

Operatory mnogościowe

W języku SQL stosuje się trzy operatory mnogościowe: IN, ANY i ALL, służące do budowy zapytań zagnieżdżonych w przypadku, gdy zapytanie wewnętrzne zwraca wiele wartości, na przykład zbiór numerów zawodników czy zbiór numerów klubów.

IN

Przykład:

Należy utworzyć listę zawodników, którzy brali udział w cyklu zawodów czyli wystąpili w co najmniej jednym biegu (analiza aktywności zawodników).

Poprzez równozłączenie zdanie to będzie wyglądało tak:

```
select distinct nazwisko, imie
from bd3_wyniki w join bd3_zawodnicy z
on w.nr zawodnika = z.nr zawodnika
order by nazwisko;
```

Równoważnym zdaniem skonstruowanym poprzez zapytanie zagnieżdzone i operator IN jest zdanie:

```
select nazwisko, imie
from bd3 zawodnicy
where nr_zawodnika in ( select nr_zawodnika
                        from bd3_wyniki)
                                             -- zbiór zawiera 452 nazwiska
order by nazwisko;
```

Zdanie wewnętrzne tworzy zbiór wynikowy w postaci kolumny nr_zawodnika, która zawiera wszystkie numery zawodników biorących udział w cyklu zawodów. Zdanie zewnętrzne dokonuje wyboru kolumn nazwisko i imię dla wybranych numerów. Warto zauważyć, że w powyższej konstrukcji zdanie wewnętrzne tworzy zbiór niepowtarzających się numerów zawodników (czyli stosowana jest niejawnie klauzula distinct).

8

Można również stosować operator NOT IN będący zaprzeczeniem operatora IN, na przykład w celu określenia liczby czy wyboru zawodników, którzy nie startowali w żadnych zawodach, a są w ewidencji:

```
select count(*)
from bd3 zawodnicy
where nr_zawodnika not in ( select nr_zawodnika
                              from bd3_wyniki);
                                             -- zbiór zawiera 319 nazwisk
```

ANY i ALL

Podobnie jak IN, operator ANY (ang. any - jakikolwiek) oraz ALL (ang. all - wszystkie) umożliwia porównanie kolumny tabeli do zbioru wartości.

W swej najprostszej postaci ANY jest odpowiednikiem IN, gdyż:

IN jest równoważne zapisowi = ANY

Przykładowo zdanie:

```
select nr_klubu, nazwa_klubu
from bd3_kluby
where nr klubu = any ( select nr klubu
                       from bd3 zawodnicy
                       where nr_klubu > 15)
order by nr_klubu;
```

tworzy zestawienie klubów o numerach powyżej 15, które mają zarejestrowanych zawodników.

Operator ANY i ALL może być używany łącznie z innymi operatorami relacji, takimi jak >, <, lub <>. Rozważmy przykładowe zdanie:

```
select A
       from Tabela Zewnętrzna
       where A < any ( select B
                              from Tabela_Wewnetrzna );
```

Załóżmy, że zawartość obu tabel jest następująca:

Tabela_Zewnętrzna	Tabela_Wewnętrzna
2	3
4	5
6	7
8	
10	

Zbiór wynikowy powyższego zdania zawierał będzie wartości: 2,4 i 6, gdyż dla każdej z nich można w Tabeli_Wewnętrznej dobrać wartość od niej większą. Innymi słowy warunek where tego zdania można czytać jako: ".....jeśli A jest mniejsze od jakiejkolwiek wartości zbioru B."

Natomiast zdanie:

```
select A
       from Tabela Zewnętrzna
       where A > any ( select B
                              from Tabela_Wewnetrzna );
```

9

zwróci wartości: 4, 6, 8 i 10, gdyż tylko 2 nie jest większe od jakiejkolwiek wartości zbioru B. W przypadku użycia operatora ALL zdanie:

```
select A
       from Tabela Zewnętrzna
       where A > all ( select B
                              from Tabela Wewnetrzna);
```

należy czytać jako: ".....jeśli A jest większe od wszystkich wartości zbioru B." Zbiór wynikowy będzie składał się z liczb 8 i 10, gdyż tylko one są większe od wszystkich wartości w zbiorze B.

Uwagi:

1. Jeśli zbiór wartości zwrócony przez zdanie wewnętrzne jest zbiorem pustym, wtedy zdanie zewnętrzne też zwróci zbiór pusty. Przykładowo zdanie:

```
select nazwisko, imie
from bd3 zawodnicy
where nr_zawodnika = ( select nr_zawodnika
                       from bd3_wyniki
                       where nr zawodow = 5);
```

zwróci zbiór pusty, gdyż brak jest w bazie wyników zawodów o numerze 5.



2. Konstrukcji zagnieżdżonych można używać również w zdaniach DML (insert, update, delete). Przykładowo:

```
update bd3 zawodnicy
       set nr_klubu = ( select nr_klubu
                       from bd3_zawodnicy
                       where nr_zawodnika = 334)
where nr_zawodnika < 300
and plec = 'M';
```

modyfikuje tabelę BD3_ZAWODNICY w ten sposób, że mężczyzn o numerach mniejszych niż 300 przenosi do klubu, w którym jest zarejestrowany zawodnik o numerze 334.

A zdanie:

```
delete bd3 zawodnicy
where nr zawodnika not in
       ( select w.nr_zawodnika
        from bd3_wyniki w join bd3_zawodnicy z on w.nr_zawodnika = z.nr_zawodnika
                          join bd3 zawody za on w.nr zawodow = za.nr zawodow
        where extract ( year from data_zawodow ) between extract ( year from sysdate ) - 10
          and extract ( year from sysdate )
       );
```

kasuje z ewidencji zawodników, którzy nie startowali w zadanym okresie w żadnych zawodach, mino, że mogli startować wcześniej.

Uwaga: Aby powyższe zdanie mogło się wykonać należy zmodyfikować model BiegiBaza tak, aby możliwe było kasowanie obiektów nadrzędnych wraz z pozostającymi z nimi w relacji obiektami podrzędnymi. W tym przypadku obiektem nadrzędnym jest wiersz w tabeli BD3_ZAWODNICY, a obiektami podrzędnymi wiersze w tabeli BD3 WYNIKI jego dotyczące.

Można to uczynić zdaniami SQL modyfikującym relacje między tymi tabelami:

```
alter table bd3_wyniki
drop constraint fk_bd3_zawodnicy_bd3_wyniki;
alter table bd3_wyniki
add constraint fk_bd3_zawodnicy_bd3_wyniki foreign key (nr_zawodnika)
references bd3_zawodnicy (nr_zawodnika) on delete cascade;
```

3. Nie istnieją żadne ścisłe ograniczenia związane z głębokością zagnieżdżania zdań wewnętrznych³:

, jak również liczbą zdań wewnętrznych zastosowanych jednocześnie, na przykład:

4. Podsumowując zagadnienie zapytań zagnieżdżonych należy pamiętać, że im większe są tabele bazodanowe, tym wydajniejsza jest ta metoda w porównaniu z metodą łączenia tabel.

Zapytania zagnieżdżone - zaawansowane zastosowania

Do tej pory omawiane było zastosowanie zapytań zagnieżdżonych do celów filtrowania w zdaniu *select* , *update* lub *delete* czyli były one umieszczane we frazie *where....* oraz do modyfikacji danych w tabeli czyli w zdaniu *update* we frazie *set....* .

Technika zapytań zagnieżdżonych może być stosowana również w innych frazach zdania *select* takich jak fraza *select..., from...* i *having...,* jak również w zdaniu *insert* we frazie *values ...* .

Podzapytanie we frazie having zdania select

Fraza *having* powoduje, że zbiór wyników generowanych przez zdanie *select* z funkcją agregującą zawiera tylko te wiersze, które spełniają warunek logiczny zawarty we frazie *having*.

Na przykład zdanie:

opr. Józef Woźniak

-

³ Oracle pozwala na używanie zagnieżdżeń na 255 poziomach we frazie *where*, co wydaje się być wielkością czysto teoretyczną.

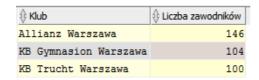
utworzy zbiór:

∜ Klub	Liczba zawodników
Allianz Warszawa	146
KB Gymnasion Warszawa	104
KB Trucht Warszawa	100
KB Lechici Zielonka	73
AZS Uniwersytet Warszawski	72
KB Promyk Ciechanów	46
AZS SGGW Warszawa	39
Warszawianka	32
KB Legionowo	32

Chcąc ograniczyć liczbę wierszy w tym zbiorze należy zastosować frazę having:

```
select nazwa_klubu "Klub", count (*) "Liczba zawodników"
from bd3_zawodnicy z
       join bd3_kluby k on z.nr_klubu = k.nr_klubu
group by nazwa klubu
having count ( * ) >= 100
order by "Liczba zawodników" desc;
```

i wtedy zbiór wyników zostanie zawężony:



Użycie warunku ograniczającego count (*) >= 100 we frazie where powoduje błąd wykonania:

```
SQL Error: ORA-00934: funkcja grupowa nie jest tutaj dozwolona
00934. 00000 - "group function is not allowed here"
```

Można natomiast użyć frazy where do filtrowania wierszy przed użyciem funkcji agregującej, na przykład:

```
select nazwa klubu "Klub", count (*) "Liczba zawodników"
from bd3 zawodnicy z
  join bd3 kluby k on z.nr klubu = k.nr klubu
where plec = 'K'
group by nazwa_klubu
having count (*) > 10
order by "Liczba zawodników" desc;
        KB Gymnasion Warszawa
Allianz Warszawa
                              18
KB Trucht Warszawa
```

Fraza where plec = 'K' filtruje wiersze z tabeli BD3 ZAWODNICY dotyczące tylko kobiet i tylko te wiersze są poddane funkcji agregującej count. Natomiast fraza having, w powstałym zbiorze wynikowym, pozostawia tylko te wiersze, które spełniają warunek użyty w niej.

13

W obu zaprezentowanych przykładach z frazą having można zauważyć, że w zdaniu tam występującym jedna ze stron tego zdania zawiera konkretną wartość (100 lub 10). W takim razie należy przypuszczać, że zamiast konkretnej wartości można zastosować podzapytanie wzorem poprzednio omawianych zagadnień.

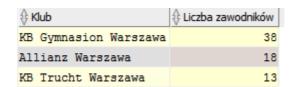
12

Na przykład zdanie:

```
select nazwa klubu "Klub", count (*) "Liczba zawodników"
from bd3 zawodnicy z
       join bd3 kluby k on z.nr klubu = k.nr klubu
where plec = 'K'
group by nazwa_klubu
having count (*) >= ( select count (*)
                      from bd3_zawodnicy
                      where nr_klubu = 31 and plec = 'K')
```

order by "Liczba zawodników" desc;

wygeneruje dokładnie ten sam zbiór wynikowy, co poprzednie:



Istotna różnica jest taka, że poprzednio "na sztywno" ustawiona była wartość 10, a teraz ten warunek ma charakter dynamiczny, gdyż liczba zarejestrowanych kobiet w klubie o numerze 31 może ulegać zmianie z upływem czasu. Przy pomocy tak opracowanego zdania select reguła biznesowa mówiąca: ugeneruj zawsze raport o klubach, w których liczba kobiet jest nie mniejsza od liczby kobiet w klubie."... KB Trucht Warszawa..." będzie zawsze spełniona.

Drugim przykładem jest zdanie:

```
select nazwa klubu "Klub",
```

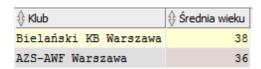
round (avg (extract (year from sysdate) - extract (year from data_urodzenia))) "Średnia wieku" from bd3_zawodnicy z join bd3_kluby k on z.nr_klubu = k.nr_klubu

where plec = 'K'

group by nazwa_klubu

having avg (extract (year from sysdate) - extract (year from data_urodzenia)) < 40 order by "Średnia wieku" desc;

tworzące zbiór wynikowy:



czyli zestawienie klubów, w których średnia wieku kobiet nie przekracza 40 lat.

Realizacja tego zdania odbywa sie według scenariusza:

- 1. Wybierane są z tabeli BD3_ZAWODNICY wiersze dotyczące kobiet i łączone z odpowiadającymi im nazwami klubów z tabeli BD3_KLUBY,
- 2. Obliczane są średnie wieku kobiet dla poszczególnych klubów,
- 3. Z tak otrzymanego zbioru filtrowane są tylko te wiersze, w których ta średnia jest mniejsza od 40 lat.

Warto zauważyć, że we frazie having, jak również group by nie można używać aliasów.

Trzeci przykład: Chcemy utworzyć podobny do poprzedniego zbiór wynikowy prezentujący tylko te kluby, w których średnia wieku kobiet jest mniejsza od średniej wieku wszystkich zaewidencjonowanych kobiet.

13

Zdanie pomocnicze:

```
select avg (extract (year from sysdate) - extract (year from data urodzenia))
from bd3 zawodnicy
where plec = 'K';
```

oblicza tę średnią:

```
AVG(EXTRACT(YEARFROMSYSDATE)-EXTRACT(YEARFROMDATA_URODZENIA))
                45,38461538461538461538461538461538461538
```

Zatem zdanie zasadnicze można skonstruować tak:

```
select nazwa_klubu "Klub",
 round ( avg ( extract ( year from sysdate ) - extract ( year from data_urodzenia ))) "Średnia wieku"
from bd3_zawodnicy z join bd3_kluby k on z.nr_klubu = k.nr_klubu
where plec = 'K'
group by nazwa_klubu
having avg ( extract ( year from sysdate ) - extract ( year from data_urodzenia )) <</pre>
                ( select avg ( extract ( year from sysdate ) - extract ( year from data_urodzenia ))
                 from bd3 zawodnicy
                 where plec = 'K)
order by "Średnia wieku" desc;
```

i otrzymamy zbiór:

∜ Klub	∜ Średnia wieku
AZS Uniwersytet Warszawski	45
KB Gymnasion Warszawa	45
KB Trucht Warszawa	44
KB Lotos Jabłonna	44
Warszawianka	43
Legia Warszawa	43
KB Promyk Ciechanów	41
Bielański KB Warszawa	38
AZS-AWF Warszawa	36

Uwaga:

- 1. Należy zawsze pamiętać, że zdanie podrzędne musi zaczynać się frazą select i być umieszczone w nawiasach okrągłych.
- 2. Zaokrąglenie liczb zostało zastosowane tylko do wyświetlenia końcowych wyników (we frazie select zdania zewnętrznego), a nie do liczenia średnich i filtrowania we frazie having.

Podzapytanie we frazie from zdania select

Fraza from w swej wersji podstawowej zawiera nazwy tabel, z których pochodzą kolumny użyte do tworzenia zbioru wynikowego. W wersji zaawansowanej dopuszcza się, aby na liście tej frazy znajdowały się dowolne zbiory, a więc oprócz tabel także podzapytania w postaci zdań select Przykładem niech będzie zdanie:

```
select nazwisko | ' ' | imie as "Zawodnik", nazwa_klubu as "Klub"
from bd3_zawodnicy z, ( select nazwa_klubu, nr_klubu
                              from bd3 kluby
                               where nr_klubu in (31, 10)) k
```

14

```
where z.nr_klubu = k.nr_klubu
and plec = 'K'
order by "Zawodnik";
```

Zasady łączenia tabel omawiane wcześniej mają w powyższej konstrukcji zastosowanie. Zdanie select o aliasie k ogranicza zbiór klubów do dwóch i ten zbiór jest łączony z tabelą BD3 ZAWODNICY poprzez nr klubu.

	∯ Klub				
Antropik Jolanta	KB Trucht Warszawa				
Biała Iza	AZS Uniwersytet Warszawski				
Bobrownicka Sylwia	AZS Uniwersytet Warszawski				
Dobosz Aleksandra	AZS Uniwersytet Warszawski				
Jabłońska Anna	KB Trucht Warszawa				

Oczywiście ten sam efekt można uzyskać łącząc ze sobą bezpośrednio dwie tabele BD3_ZAWODNICY i BD3 KLUBY. Ale w wielu przypadkach to rozwiązanie może okazać się wydajniejsze (szczególnie w hurtowniach danych, gdy liczba wierszy w tabelach jest bardzo duża), gdyż jeden ze zbiorów wejściowych już na początku jest ograniczony tylko do dwóch wierszy.

Drugim praktycznym zastosowaniem podzapytań we frazie from jest problem limitowania liczby wierszy w zbiorze wynikowym.

Jak wygenerować zbiór zawierający sumaryczne wyniki dziesięciu najlepszych klubów lub opracować raport zawierający pięciu najlepszych zawodników w każdych zawodach?

Pseudokolumna rownum

Pseudokolumna (wirtualna) rownum numeruje wiersze w zbiorze wynikowym podobnie jak popularna kolumna Lp na papierowych zestawieniach.

Sposób użycia przedstawia poniższe zdanie:

```
select rownum, nazwisko || ' ' || imie "Zawodnik", data_urodzenia "Data urodzenia"
from bd3_zawodnicy
where nr_klubu = 3;
```

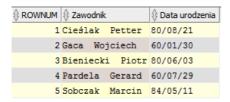
i wygenerowany zbiór wynikowy:



Możliwe jest użycie pseudokolumny rownum do ograniczenia liczby wyświetlanych wierszy, na przykład:

```
select rownum, nazwisko || ' ' || imie "Zawodnik", data_urodzenia "Data urodzenia"
from bd3 zawodnicy
where nr_klubu = 2 and rownum <= 5;
```

15

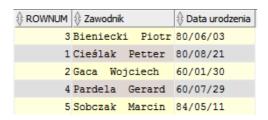


Powyższy zbiór zawiera pięć wierszy, zgodnie z dyspozycją dotyczącą rownum, ale należy zadać sobie pytania: Co to są za wiersze? Według jakiego kryterium został dobrany ten zbiór wynikowy? Widać, że nie jest to ani nazwisko ani data urodzenia zawodnika. Wybór jest przypadkowy i zależy od położenia wierszy w tabeli czyli w jakiej kolejności zostały do niej wprowadzane, na przykład zdaniem insert.

Po uwzględnieniu sortowania według, na przykład, nazwiska, czyli:

```
select rownum, nazwisko || ' ' || imie "Zawodnik", data_urodzenia "Data urodzenia"
from bd3_zawodnicy
where nr_klubu = 3 and rownum <= 5
order by nazwisko:
```

otrzymamy zbiór o zawartości:



Zbiór jest posortowany według nazwisk, ale "liczba porządkowa" generowana przez rownum jest zakłócona. Po analizie tego przykładu można dojść do wniosku, że w czasie wykonywania się powyższego zdania select najpierw dokonuje się numeracja wierszy (rownum), a potem sortowanie zbioru wynikowego. I tak jest w istocie. Sortowanie zbioru wynikowego jest zawsze ostatnią czynnością wykonywaną w zdaniu select.

Limitowanie typu rownum <= wartość_progowa

W takim przypadku należy tak skonstruować zdanie select, aby najpierw odbyło się sortowanie, a potem numerowanie wierszy. Z pomocą przychodzi zastosowanie podzapytania we frazie from.

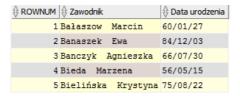
Zdanie:

```
select rownum, "Zawodnik", "Data urodzenia"
from ( select nazwisko || ' ' || imie "Zawodnik", data_urodzenia "Data urodzenia"
       from bd3_zawodnicy
       where nr klubu = 3
       order by nazwisko)
where rownum <= 5;
```

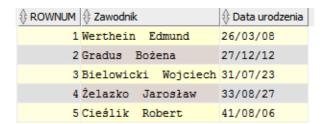
zostanie wykonane według scenariusza:

- 1. Wybranie danych o zawodnikach klubu o numerze 3 (zdanie wewnętrzne),
- 2. Posortowanie wybranych zawodników według nazwisk (zdanie wewnętrzne),
- 3. Ponumerowanie wierszy (rownum we frazie select zdania zewnętrznego),
- 4. Filtrowanie wierszy według frazy where zdania zewnętrznego.

Ostateczna postać zbioru wynikowego przedstawia się następująco:



Zbiór zawiera pięć pierwszych nazwisk zawodników klubu o numerze 3. Zmieniając klucz sortowania w podzapytaniu z nazwiska na datę urodzenia otrzymamy zbiór:



prezentujący dane pięciu najstarszych zawodników tego klubu.

Limitowanie typu wartość_progowa_min <= rownum <= wartość_progowa_max

W analogiczny sposób należy postępować w przypadku, gdy limitowanie liczby wierszy dotyczy warunku:

.....rownum between numer_wiersza_od and numer_wiersza_do

Zdanie:

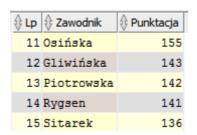
```
--3 select * from (

--2 select rownum "Lp", "Zawodnik", "Punktacja" from (

--1 select nazwisko "Zawodnik", sum ( nvl ( punkty_globalne, 0 )) "Punktacja" from bd3_wyniki w join bd3_zawodnicy z on z.nr_zawodnika = w.nr_zawodnika group by nazwisko order by "Punktacja" desc
)

) where "Lp" between 11 and 15;
```

wygeneruje zbiór:



Zdanie to zostanie wykonane według scenariusza:

- 1. Wybranie danych zgodnie ze zdaniem --1 (posortowanie malejąco według punktacji),
- 2. Ponumerowanie wierszy od 1 i nadanie tej kolumnie aliasu "Lp" (select --2),
- 3. Filtrowanie zbioru według "Lp" w zakresie [11..15] (select --3).

Są to wiersze zbioru od pozycji 11 do 15 według kryterium kolumny "Punktacja".

Warto zauważyć, że w zdaniu select --2 kolumna "Punktacja" nie jest już agregatem tylko kolumną skalarną, co oznacza, że w tym zdaniu można by było użyć filtrowania ...where "Punktacja" > 100..., a nie ...having "Punktacja" > 100....

Pseudokolumna *rownum* została opatrzona aliasem "Lp", dzięki czemu w zdaniu *select* --3 nie posiada już właściwości *rownum*, tylko jest zwykłą kolumną, której można używać do filtrowania wierszy.

Limitowanie typu rownum >= wartość_progowa

Limitowanie według *rownum* oznacza sprecyzowanie, ile wierszy ma znaleźć się w zbiorze wynikowym. Warunek *rownum* <= 5 determinuje co najwyżej pięć wierszy, warunek *rownum between* 11 *and* 15 - również.

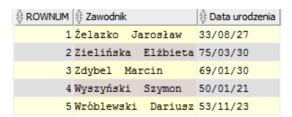
Na podstawie warunku *rownum* >= 5 nie można określić górnej granicy liczby wierszy. Dlatego zdanie z jednego z poprzednich przykładów ze zmodyfikowanym warunkiem filtrowania:

```
select rownum, "Zawodnik", "Data urodzenia"
from (     select nazwisko || ' ' || imie "Zawodnik", data_urodzenia "Data urodzenia"
     from bd3_zawodnicy
     where nr_klubu = 3
        order by nazwisko )
where rownum >= 5;
```

zwraca zbiór pusty.

Ponieważ limitowanie zawsze musi być związane z określeniem jakiegoś kryterium należy na etapie sortowania dokonać odpowiedniego uporządkowania zbioru. Zdanie powyższe może wyglądać tak:

a wynik:



Został odwrócony kierunek sortowania czyli w zbiorze wynikowym znalazło się pięć ostatnich nazwisk, ale uporządkowanych począwszy od ostatniego.

Jeśli istnieje potrzeba posortowania otrzymanego zbioru według nazwisk w porządku naturalnym należy zagnieżdżać opracowane zdania:

```
    --3 select rownum "Lp", "Zawodnik", "Data urodzenia" from
    --2 (select "Zawodnik", "Data urodzenia" from
    --1 (select "Zawodnik", "Data urodzenia"
```

```
from ( select nazwisko || ' ' || imie "Zawodnik", data_urodzenia "Data urodzenia"
          from bd3 zawodnicy
          where nr klubu = 3
          order by nazwisko desc )
     where rownum <= 5
 order by "Zawodnik"
);
```

Zbiór wynikowy tworzony jest według scenariusza:

- 1. Na podstawie select --1 powstaje zbiór pięciu wierszy z nazwiskami z końca alfabetu posortowany odwrotnie czyli od Ż do W,
- 2. Zbiór ten jest sortowany według naturalnego porządku nazwisk zdaniem select --2 czyli od W do Ż,
- 3. Powstały zbiór jest numerowany w zdaniu select -- 3.

Uwaga:

Począwszy od wersji Oracle 12c zdanie select zostało wzbogacone o implementację limitowania bez potrzeby używania konstrukcji przedstawionych powyżej. Szczegóły zawarte są w dokumentacji Oracle lub w innych zasobach internetowych pod hasłem (row limiting clause oracle).

Podzapytanie we frazie select zdania select

W podstawowej swojej konstrukcji zdanie select zawiera we frazie select nazwy kolumn lub funkcje agregujace opatrzone lub nie aliasami.

Istnieje możliwość umieszczenia w tej frazie podzapytania zwracającego wartość skalarną.

Przy pomocy zdania:

```
select nazwisko||' '||imie "Zawodnik", nazwa_klubu "Klub", sum(punkty_globalne) "Pkt indywidualne"
       from bd3_zawodnicy z join bd3_kluby k on z.nr_klubu = k.nr_klubu
       join bd3_wyniki w on z.nr_zawodnika = w.nr_zawodnika
group by nazwisko, imie, nazwa klubu
having sum(punkty globalne) is not null
order by "Pkt indywidualne" desc;
```

tworzony jest standardowy zbiór wynikowy zawierający osiągnięcia zawodników w postaci sumarycznej liczby punktów przez nich zdobytych:

∜ Zawodnik	∜ Klub		
Muzyka Urszula	Legia Warszawa	198	
Karpisz Zbigniew	KB Trucht Warszawa	194	
Pałasz Joanna	KB Trucht Warszawa	192	
Ciemski Jan	KB Lechici Zielonka	189	
Gawryszewski Radosław	KB Trucht Warszawa	187	
Margoła Bartłomiej	Allianz Warszawa	185	

Chcemy poszerzyć ten zbiór o dodatkowe kolumny zawierające sumaryczną ilość punktów zdobytych przez wszystkich zawodników klubu, do którego należy zawodnik prezentowany w każdym wierszu zbioru oraz liczbę zawodników należących do tego samego klubu.

Zdanie może wyglądać tak:

```
select nazwisko "Zawodnik", nazwa klubu "Klub"
    ,( select suma_z from ( select sum ( nvl (punkty_globalne, 0 ) ) suma_z
                   from bd3 wyniki wz
                   where wz.nr zawodnika = z.nr zawodnika)
     ) "Pkt zawodnika"
    ,( select suma_k from ( select sum ( nvl (punkty_globalne, 0 ) ) suma_k
                   from bd3_wyniki wk, bd3_zawodnicy zk
                   where zk.nr_klubu = z.nr_klubu and zk.nr_zawodnika = wk.nr_zawodnika)
     ) "Pkt klubu"
    ,( select liczba_z from ( select count(*) liczba_z
                   from bd3_zawodnicy zl
                   where z.nr klubu = zl.nr klubu )
     ) "Liczność klubu"
from bd3_zawodnicy z, bd3_kluby k
where z.nr klubu = k.nr klubu
order by "Zawodnik";
```

, a fragment zbioru wynikowego:

					Liczność klubu I
787	Chrapek	Warszawianka	(null)	218	32
855	Chruśliński	Warszawianka	0	218	32
684	Chrzanowski	Allianz Warszawa	0	1425	146
657	Chrześcijański	KB Pułaski Strong Warka	(null)	0	8
460	Chudek	KB Lotos Jabłonna	28	432	17
547	Ciecierski	KB Promyk Ursus	(null)	39	16
831	Ciemski	KB Lechici Zielonka	(null)	821	73
840	Ciemski	KB Lechici Zielonka	189	821	73
303	Ciesielski	KB Promyk Ciechanów	63	568	46
268	Cieślak	KB Gymnasion Warszawa	0	1763	104
264	Cieślak	Warszawianka	97	218	32

Scenariusz wykonania tego zdania jest następujący:

- 1. Zdaniem zewnętrznym *select* pobierane są dane o pierwszym zawodniku ze zbioru uporządkowanego według alfabetu,
- 2. Z odczytanego wiersza pobierany jest numer zawodnika **z.nr_zawodnika** i "przenoszony" do podzapytania *select* w celu zsumowania jego wyników,
- 3. Z tego samego wiersza odczytywany jest numer klubu, do którego zawodnik należy **z.nr_klubu** i na jego podstawie w podzapytaniu *select* obliczane są sumaryczne punkty tego klubu (we frazie *where* jest ustawiony filtr zk.nr klubu = z.nr klubu).
- 4. Z tym samym numerem klubu **z.nr_klubu** realizowane jest trzecie podzapytanie *select* obliczające liczbę zawodników w tym klubie.
- 5. Zdaniem zewnętrznym select pobierane są dane kolejnego zawodnika.

Uwagi:

- 1. Na liście select znajdują się trzy podzapytania skorelowane ze zdaniem głównym.
- 2. Każde z tych podzapytań musi zwracać wielkość skalarną, stąd też każde z nich jest skonstruowane w sposób to zapewniający:

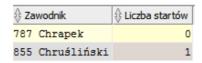
select skalar from (select agregat from...)

- 3. W konstrukcji występują fizycznie te same zbiory danych, ale opatrzone różnymi aliasami czyli logicznie sa to różne zbiory, na przykład tabela BD3 ZAWODNICY występuje jako alias z w zdaniu zewnetrznym oraz jako alias **zk** i **z**/ w podzapytaniach.
- 4. W zaprezentowanym zbiorze wyników w kolumnie "Pkt zawodnika" występuja wartości 0 oraz null. Wartość 0 oznacza, że dany zawodnik startował w zawodach, ale nie zdobył punktów w klasyfikacji generalnej czyli w tabeli BD3 WYNIKI figuruje. Wystąpienie null w tej kolumnie oznacza, że brak jest wyników zawodnika w tabeli wyników czyli zawodnik ten nie startował wcale. Zastosowanie funkcji nvl w podzapytaniu select umożliwia rozróżnienie tych faktów.

Wykonując zdanie:

```
select '787 Chrapek' "Zawodnik", count(*) "Liczba startów"
from bd3 wyniki
where nr_zawodnika = 787
union
select '855 Chruśliński', count(*)
from bd3_wyniki
where nr zawodnika = 855;
```

otrzymujemy wynik:



, co potwierdza powyższa tezę.

Zadania do samodzielnego wykonania:

- 1. Opracować zestawienie pokazujące zawodników i osiągnięte przez nich rezultaty, którzy należą do tego samego klubu, co zwycięzca zawodów nr 2 wśród mężczyzn.4
- 2. Opracować zestawienie kobiet należących do klubów, w których liczność zawodników przekracza liczbę 5.
- 3. Opracować zestawienie (Imię i nazwisko, Nazwa klubu i Nazwa Kategorii) pokazujące zawodniczki, które należą do klubów niewarszawskich i ich wiek jest większy od średniej wieku wszystkich kobiet w ewidencji.5
- 4. Opracować zdanie pokazujace zawodnika i zawodniczke (Nazwisko i imie, przynależność klubowa oraz kategoria wiekowa), którzy zdobyli najwiecej punktów w klasyfikacji generalnej.
- 5. (trudne) Opracować zdanie pokazujące najlepszych zawodników i zawodniczki (Nazwisko i imię, przynależność klubowa oraz kategoria wiekowa), którzy zdobyli najwięcej punktów w klasyfikacji w kategoriach.

W tym zadaniu należy wykorzystać następujące techniki:

- zamiana minut i sekund na sekundy,
- tworzenie na liście from perspektyw chwilowych: pierwszej zawierającej najlepszy wynik w każdej kategorii wiekowej i drugiej znajdującej numer zawodnika w każdej kategorii wiekowej, który ten najlepszy wynik uzyskał.

⁴ Należy skorzystać ze zdania podrzędnego, a rezultaty wyświetlić w postaci mm:ss.

⁵ Należy wykonać raport przy pomocy zdania podrzędnego. Jako kryterium kwalifikacji do klubów warszawskich przyjąć nazwę klubu, w której znajduje się fragment słowa Warszawa (np. Warsz) w dowolnym miejscu nazwy.