	Bazy Danych laboratorium	**Laboratorium BD3**

Pozyskiwanie informacji z wielu tabel - opis przykładu

Model bazy danych, na podstawie którego będą realizowane to i dalsze laboratoria stanowi rozwinięcie przykładu poprzedniego (Laboratorium BD2). Ten model zawiera kilka tablic połączonych relacjami ze sobą, ale istota problemu pozostaje ta sama. Poniżej przedstawiony jest schemat relacyjnego modelu zobrazowany przy pomocy aplikacji DBeaver w notacji *Crows Foot*:

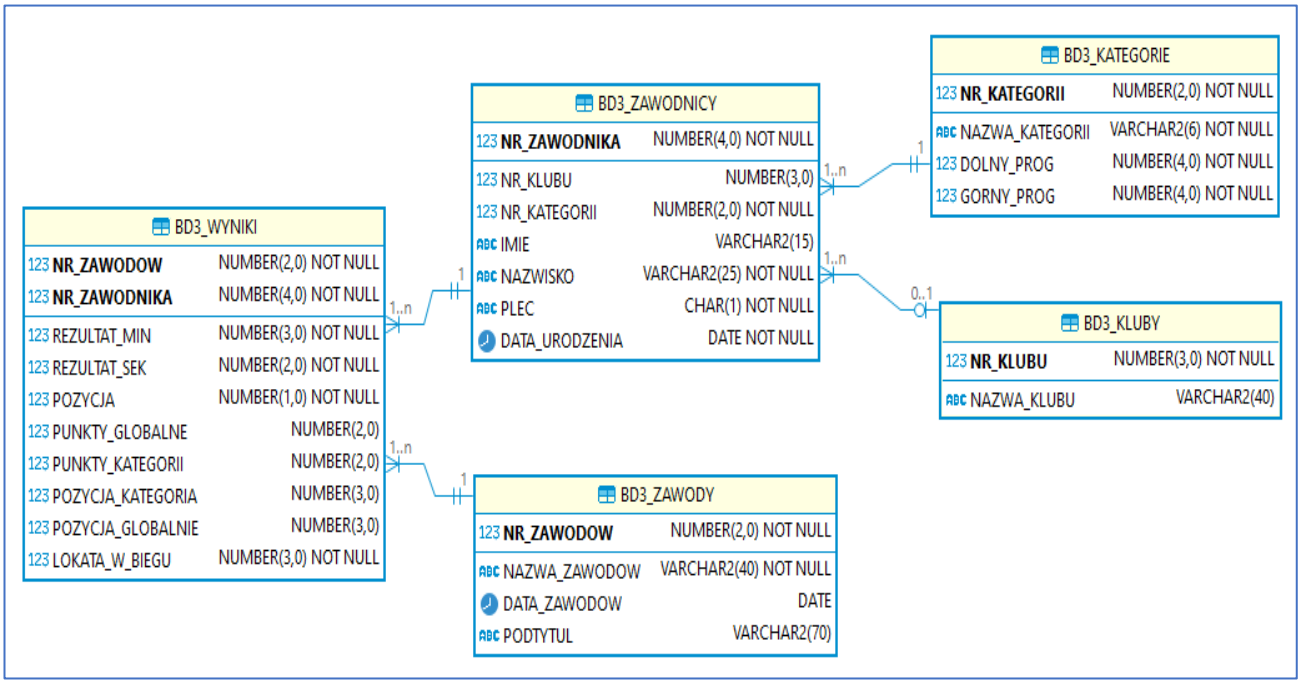


Tabela BD3_WYNIKI zawiera ewidencję wyników cyklu biegowych zawodów sportowych. W ramach jednego biegu zapisywana jest klasyfikacja zawodników, na którą składają się wyniki sportowe zawodnika, punkty zdobyte w klasyfikacji generalnej i punkty zdobyte w klasyfikacji wiekowej.
Tabela BD3_ZAWODNICY to numer ewidencyjny, imię i nazwisko, płeć, data urodzenia. Kategoria wiekowa i przynależność klubowa zawodnika realizowana jest w relacji z odpowiednimi tabelami.

Zawodnicy są podzieleni na kategorie wiekowe (tabela BD3_KATEGORIE) według roku urodzenia, np. kategoria IV to mężczyźni urodzeni między 1957 (dolny próg) i 1967 (górny próg) rokiem, a K-II to kobiety urodzone między 1977 i 1986 rokiem.

Punkty klasyfikacji generalnej (*Punkty globalne* w tabeli BD3_WYNIKI) – 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 jest niewypełnione. Analogicznie jest wśród kobiet.

Punkty w klasyfikacji wiekowej(*Punkty kategorii* w tabeli BD3_WYNIKI) – 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.

W skrypcie *lab_BD3_create.sql* można zapoznać się dokładnie ze zdaniami SQL implementującymi powyższy model w bazie danych.[1]

W powyższym skrypcie występują dodatkowe (poza definicją kluczy głównych i obcych) ograniczenia na wybrane kolumny:

1. Klauzula *not null* – system zarządzania baza danych (SZBD) nie dopuści do wpisania wiersza do tabeli, w którym kolumna posiadająca tę klauzulę nie będzie miała określonej wartości,
2. Klauzula *check* – ogranicza wartości w danej kolumnie do wyspecyfikowanych, np. ograniczenie *Plec in ('M', 'K')* zapewnia, że w kolumnie *Plec* może wystąpić tylko litera M lub K (lecz nie 'm' lub 'k'),
3. Klauzula *default* - definiuje domyślną wartość w kolumnie.

Implementacja modelu w schemacie Oracle

Implementacja modelu będzie podobna do tej przeprowadzonej w Laboratorium BD2, przy czym tym razem nie będzie to jedna tabela tylko pięć połączonych ze sobą relacjami w postaci kluczy obcych.
W takim przypadku może być istotna kolejność zarówno tworzenia tabel, jak i wypełniania ich danymi.
Analizując graficzną prezentację modelu można zauważyć hierarchię obiektów. Na przykład relacja między tabelami BD3_KATEGORIE i BD3_ZAWODNICY określa tę pierwszą tabelę jako obiekt nadrzędny w stosunku do drugiej. To zawodnicy są "podpięci" pod daną kategorię wiekową (do jednej kategorii należy wielu zawodników). W zdaniach SQL *create table*....lub *alter table*.... tworzących referencje objawia się to tym, że definicja referencji (klucza obcego) znajduje się w tabeli podrzędnej. Klucz obcy wiążący tabele BD3_KATEGORIE i BD3_ZAWODNICY znajduje się w tabeli drugiej czyli kolejność implementacji tabel podlega zasadzie: "najpierw nadrzędny, potem podrzędny".

```
create table BD3_KATEGORIE
(
        NR_KATEGORII        NUMBER(2)     primary key,
        NAZWA_KATEGORII     VARCHAR2(6) not null,
        DOLNY_PROG          NUMBER(4)     not null,
        GORNY_PROG          NUMBER(4)     not null,
);

create table BD3_ZAWODNICY
(
NR_ZAWODNIKA        NUMBER(4) primary key,
NR_KLUBU            NUMBER(3) references BD3_KLUBY (NR_KLUBU),
NR_KATEGORII        NUMBER(2) not null references BD3_KATEGORIE (NR_KATEGORII),
IMIE                VARCHAR2(15),
NAZWISKO            VARCHAR2(25)   not null,
PLEC                CHAR(1)        not null,
DATA_URODZENIA      DATE           not null,
);
```

Niedogodnością tego rozwiązania jest konieczność odpowiedniego uszeregowania tworzonych obiektów, jak również kolejność wprowadzania danych z plików zewnętrznych. Problem ten występuje wyraźnie w modelach zawierających kilkadziesiąt lub więcej tabel.

[1] Dodatkowo w materiałach do laboratorium znajduje się skrypt *lab_BD3_drop.sql*, przy pomocy którego można całkowicie usunąć model BiegiBaza ze swojego schematu.

Drugim wariantem implementacji jest opóźnienie definiowania referencji przy pomocy zdań *alter table*..., które następuje po wszystkich zdaniach *create table*...., w których nie ma definicji referencji. W takim przypadku możliwe jest uszeregowanie tworzonych tabel według alfabetu, następnie wprowadzenie danych do tabel nie analizując skutków działania referencji i na końcu zdefiniowanie referencji. Skrypt *lab_BD3_create.sql* umożliwia implementację modelu według tego wariantu poprzez uruchamianie go partiami.

Trzeci wariant polega na zmianie kolejności wykonywanych czynności według scenariusza:

tworzenie tabel (zdania *create table*... według alfabetu),
tworzenie referencji (zdania *alter table*... w dowolnej kolejności),
wprowadzanie danych do tabel (w odpowiedniej kolejności wynikającej z hierarchii tabel).

Skrypt *lab_BD3_create.sql* umożliwia implementację modelu według trzeciego wariantu i tak należy ją zrealizować metodą „*@c:\temp\lab_BD3_create.sql*”. Zestawy danych dotyczące każdej z tabel znajdują się w osobnych plikach typu wycinek (csv):

lab_bd3_kategorie.csv
lab_bd3_kluby.csv
lab_bd3_wyniki.csv
lab_bd3_zawodnicy.csv
lab_bd3_zawody.csv

Sposób implementacji jest analogiczny do implementacji tabeli BD2_ZBIORCZA wykonanej w ramach Laboratorium BD2. Jedyną różnicą jest istnienie w plikach z danymi nagłówków z nazwami kolumn tożsamymi z nazwami kolumn w tabelach. Należy to zaznaczyć w momencie importowania danych ustawiając opcję *Header* na *Yes*.

Po wykonaniu wszystkich czynności należy kontrolnie sprawdzić liczbę wierszy w każdej z tabel zdaniem *select count (*) from tabela*. Wyniki powinny być takie, jak poniżej:[2]

Tabela	Liczba wierszy
BD3_KATEGORIE	22
BD3_KLUBY	27
BD3_WYNIKI	838
BD3_ZAWODNICY	771
BD3_ZAWODY	4

Zakończyć implementację zatwierdzeniem transakcji (*commit*).

Wyprowadzanie danych z tabel do plików zewnętrznych lub skryptów INSERT przy pomocy Oracle SQL Developer

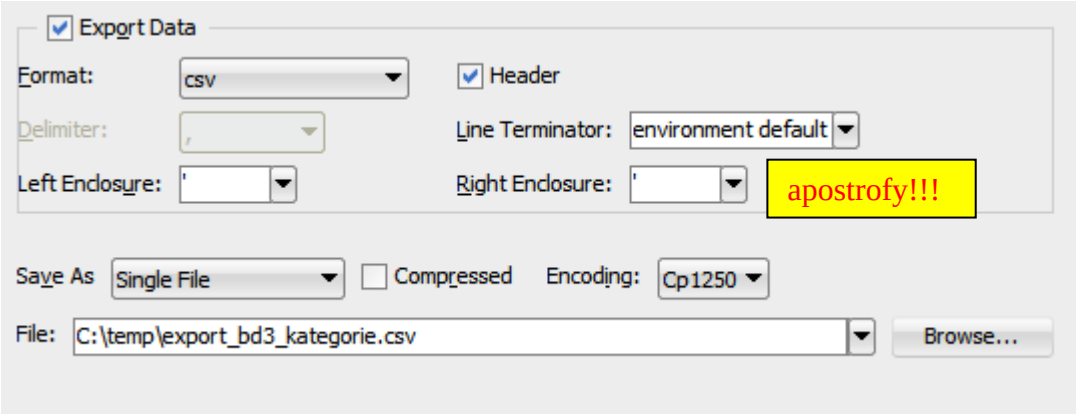
Mając tabelę bazodanową z umieszczonymi w niej danymi można dokonać eksportu, zarówno struktury tabeli (metadanych), jak i danych. Wykorzystuje się funkcję Export... usytuowaną pod przyciskiem Actions... po wyświetleniu struktury tabeli.

Chcąc dokonać eksportu danych z tabeli BD3_KATEGORIE należy:

1. Wybrać tabelę BD3_KATEGORIE w celu zobrazowania jej struktury,
2. Spod przycisku Actions... uruchomić funkcję Export...,
3. Usunąć wybór Export DDL (eksport metadanych),

[2] W materiałach do laboratorium znajduje się skrypt *lab_BD3_check.sql* zawierający przykładowe zdanie SQL.

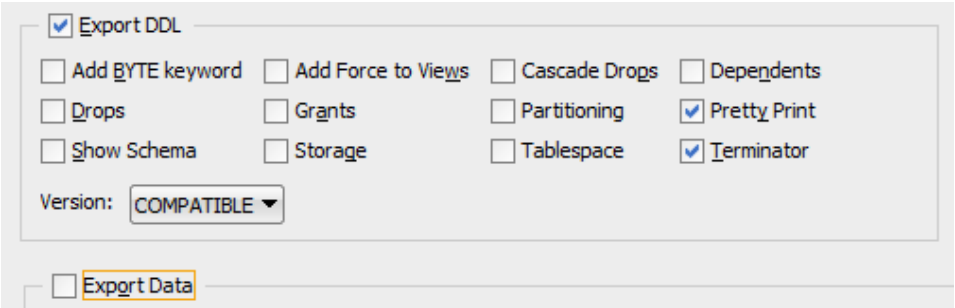
4. Opcje dotyczące eksportu danych ustawić na przykład tak, jak na poniższym rysunku:



5. Wyspecyfikować wszystkie wiersze i wszystkie kolumny tabeli (sekcja Specify Data),
6. Zakończyć proces eksportu,
7. Zapoznać się z wygenerowanym plikiem z danymi.

Powtórzyć powyższy scenariusz zmieniając Format na insert oraz nazwę pliku docelowego na *export_bd3_kategorie_insert.sql*. Wygenerowany zestaw zdań *insert* pojawi się w panelu SQL Developera, ale nie należy go wykonywać, tylko zapoznać się z utworzonym skryptem o zadanej nazwie oraz opcjonalnie zapamiętać.

Chcąc dokonać eksportu metadanych czyli struktury tabeli należy uaktywnić wybór Export DDL (ale dezaktywować podrzędne opcje według poniższego wzorca) i dezaktywować Export Data:



Wyeksportować metadane do pliku *export_bd3_kategorie_ddl.sql*. Obejrzeć i przeanalizować powstały skrypt. Zaleca się samodzielnie przećwiczyć efekty wyboru opcji Show Schema oraz Drops, jak również równoczesnego eksportu metadanych i danych w postaci zdań *insert*.

Tworzenie nowej tabeli z danymi na podstawie istniejącej tabeli

Przy pomocy języka SQL można tworzyć nową tabelę na podstawie już istniejącej i dodatkowo wypełnić ją danymi. Czynność ta jest bardzo często wykorzystywana przez administratorów baz danych lub projektantów w celu, na przykład, zabezpieczania swoich danych przed skutkami różnych eksperymentów służących testowaniu oprogramowania.

Pierwszy wariant:
Tabela źródłowa (*source*) istnieje i zawiera dane, a tabeli docelowej (*target*) nie ma.

W takim przypadku obowiązuje konstrukcja:

```
create table bd3_kluby_kopia as
  select * from bd3_kluby;
```

Odpytanie tabeli BD3_KLUBY_KOPIA zdaniem *select* da wynik:

NR_KLUBU	NAZWA_KLUBU
37	KU AZS WAT Warszawa
1	Allianz Warszawa
2	KB Orientuz Warszawa
3	KB Gymnasion Warszawa
4	Legia Warszawa
9	Grunwald Poznań
6	KB Promyk Ciechanów
7	KB Pułaski Strong Warka

....

Porównując metadane obu tabel (źródłowej i docelowej) można zauważyć jedną istotną różnicę. Utworzona tą metodą tabela nie ma klucza głównego.

Sekcja Constraints dla tabeli źródłowej:

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
PK_BD3_KLUBY	Primary_Key	(null)
SYS_C00299042	Check	"NR_KLUBU" IS NOT NULL

Sekcja Constraints dla tabeli docelowej:

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
SYS_C00299437	Check	"NR_KLUBU" IS NOT NULL

Chcąc, aby tabela docelowa miała klucz główny należy go oddzielnie zdefiniować przy pomocy zdania *alter table*... .

Uwagi:

1. Tabela docelowa nie dziedziczy również definicji kluczy obcych oraz ograniczeń typu *check* innych niż *not null*.
2. W konstrukcji *create table tabela_kopia as select * from tabela*, zdanie *select* może być dowolnej dopuszczalnej postaci, np.:

```
create table bd3_kluby_warszawa as
select nazwa_klubu from bd3_kluby
where nazwa_klubu like ' %Warszawa% ';
```

3. Chcąc utworzyć tabelę docelową o takiej samej strukturze jak tabela źródłowa, ale bez danych należy użyć konstrukcji:

```
create table bd3_kluby_kopia as
select * from bd3_kluby
where 1=0;
```

Zdanie logiczne 1=0 nie jest prawdziwe dla żadnego wiersza w tabeli, więc zbiór wyników będzie zbiorem pustym.

Drugi wariant:
Tabela źródłowa (*source*) zawierająca dane oraz tabela docelowa (*target*) istnieją.
W takim przypadku obowiązuje konstrukcja:

```
insert into bd3_zawodnicy_kobiety
select * from bd3_zawodnicy
where plec= 'K';
```

Sposób utworzenia tabeli docelowej jest dowolny:

- można tabelę utworzyć manualnie zdaniem *create table*...i *alter table*....,
- można wykorzystać omówioną powyżej konstrukcję *create table... as select * from* tabela,
- można wykorzystać funkcję eksportu metadanych poprzez funkcjonalność Actions.../Export...

Równozłączenie (złączenie wewnętrzne)

W swojej najczęściej wykorzystywanej postaci złączenie polega na utworzeniu nowych wierszy w zbiorze wynikowym na podstawie odpowiadających sobie wierszy z tabel składowych. Najczęściej to złączenie realizowane jest poprzez klucz główny jednej tabeli i odpowiadający mu klucz obcy w drugiej. W zbiorze wynikowym znajdą się wiersze, dla których w obu tabelach istnieją te same wartości kolumn stanowiących relację, a nie znajdzie się na przykład wiersz, dla którego w jednej tabeli istnieje klucz główny, ale brak go w odpowiadającym mu kluczu obcym drugiej tabeli. Ta cecha klasyfikuje opisywany proces jako ***złączenie wewnętrzne*** *(inner join* lub *join).*

Składnia złączenia wewnętrznego dla dwóch tabel:

select kolumny z tabel
from tabela1, tabela2
where tabela1.kolumna_n = tabela2.kolumna_n

np.:

select imie, nazwisko, nazwa_klubu
from bd3_zawodnicy**,** bd3_kluby
where bd3_zawodnicy.nr_klubu = bd3_kluby.nr_klubu;

	IMIE	NAZWISKO	NAZWA_KLUBU
757	Michał	Lenuszyński	KB Promyk Ciechanów
758	Marek	Herman-Iżycki	KB Promyk Ciechanów
759	Piotr	Wojtyński	KB Promyk Ciechanów
760	Piotr	Bobrowski	KB Lechici Zielonka
761	Grzegorz	Protas	KB Lechici Zielonka
762	Przemysław	Augustyniak	Akvedukt Kielce
763	Mikołaj	Jakowicz	KB Trucht Warszawa
764	Tomasz	Zajkowski	KB Trucht Warszawa
765	Jakub	Bednarczyk	KB Trucht Warszawa
766	Leszek	Kuczkowski	KB Lotos Jabłonna

Na powyższym rysunku pokazano kilka ostatnich wierszy zbioru wyników. Ostatni wiersz ma numer 766, co oznacza, że tyle wierszy zwróciło tak skonstruowane zdanie *select*. Porównując tę liczbę z liczbą wierszy w tabeli BD3_ZAWODNICY wykazaną po implementacji modelu i wprowadzeniu danych do tabel można zauważyć różnicę.
Liczność tabeli BD3_ZAWODNICY wynosi 771, a powyższy *select* zwrócił 766 wierszy. Pięciu zawodników nie jest wykazanych w tym zestawie danych. Wynika to z istoty równozłączenia, która polega na tym, że dla pięciu wierszy z tabeli BD3_ZAWODNICY nie można było dobrać odpowiedniej nazwy klubu.
Wniosek: Pięciu zawodników nie ma określonego klubu, do którego należy (w tabeli BD3_ZAWODNICY kolumna nr_klubu zawiera wartości *null*):

Można to sprawdzić zdaniem:

select *
from bd3_zawodnicy
where nr_klubu *is null*;

	NR_ZAWODNIKA	NR_KLUBU	NR_KATEGORII	IMIE	NAZWISKO	PLEC	DATA_URODZENIA
1	597	(null)	6	Krzysztof	Misiejuk	M	56/12/25
2	648	(null)	6	Adam	Kostecki	M	53/04/22
3	94	(null)	22	Katarzyna	Wojciuk	K	79/04/13
4	156	(null)	23	Justyna	Małkiewicz	K	76/01/27
5	228	(null)	6	Dawid	Wereda	M	53/10/12

Istnieje druga, równoznaczna, składnia równozłączenia:

```
select imie, nazwisko, nazwa_klubu
        from bd3_zawodnicy
                join  bd3_kluby on  bd3_zawodnicy.nr_klubu = bd3_kluby.nr_klubu;
```

Zapis ten wydaje się być bardziej uniwersalnym w kontekście innych, niżej opisanych konstrukcji, ale dla równozłączenia nie ma żadnej różnicy.

Składnia złączenia wewnętrznego dla wielu tabel:

W przypadku większej liczby tabel należy zarówno ich nazwy, jak i nazwy kolumn umieścić w odpowiednich frazach zdania *select*, jak również rozbudować odpowiednio warunki *where* lub *join*, Np. poniższe zdania:

```
select imie, nazwisko, nazwa_klubu, nazwa_kategorii
        from bd3_zawodnicy, bd3_kluby, bd3_kategorie
        where bd3_zawodnicy.nr_klubu = bd3_kluby.nr_klubu
          and bd3_zawodnicy.nr_kategorii = bd3_kategorie.nr_kategorii;
```

lub równoważne jemu:

```
select imie, nazwisko, nazwa_klubu, nazwa_kategorii
from bd3_zawodnicy
        join bd3_kluby on bd3_zawodnicy.nr_klubu = bd3_kluby.nr_klubu
        join bd3_kategorie on bd3_zawodnicy.nr_kategorii = bd3_kategorie.nr_kategorii;
```

dadzą wynik:

IMIE	NAZWISKO	NAZWA_KLUBU	NAZWA_KATEGORII
Piotr	Kuczkowski	KB Lotos Jabłonna	IV
Aleksander	Cieślak	Warszawianka	IV
Tadeusz	Strzałkowski	KB Trucht Warszawa	V
Jan	Kowalski	Allianz Warszawa	IV
Grzegorz	Grabowski	KB Trucht Warszawa	III
Petter	Cieślak	KB Gymnasion Warszawa	III
Leszek	Kowalczyk	KB Pułaski Strong Warka	VII
Marcin	Grzana	Allianz Warszawa	VI
Rafał	Radomski	Allianz Warszawa	II
Wojciech	Gaca	KB Gymnasion Warszawa	V
Tomasz	Jakubowski	KB Trucht Warszawa	III

……

W takich konstrukcjach zdania *select* można stosować aliasy nazw tabel w celu skrócenia zapisu.

Na przykład:

```
select imie, nazwisko, nazwa_klubu, nazwa_kategorii
from bd3_zawodnicy z , bd3_kluby kl , bd3_kategorie ka
where z.nr_klubu = kl.nr_klubu
  and  z.nr_kategorii = ka.nr_kategorii;
```

lub

```
select imie, nazwisko, nazwa_klubu, nazwa_kategorii
from bd3_zawodnicy z
        join bd3_kluby kl on z.nr_klubu = kl.nr_klubu
        join bd3_kategorie ka on z.nr_kategorii = ka.nr_kategorii;
```

W przypadku zdefiniowania takich aliasów, muszą one być używane w całym zdaniu do specyfikowania tabel, a w szczególności we frazach *where* lub *join* do definiowania złączeń. Niedopuszczalne jest w takim przypadku używanie w jednym zdaniu *select* zarówno pełnej nazwy tabeli, jak i jej aliasu.

W równozłączeniach można rozbudowywać frazę *where* o różnego rodzaju filtry według zasad obowiązujących dla pobierania danych z jednej tabeli, jak również sortować zbiór wynikowy.
Na przykład zdanie:

```
select nazwisko || '  ' || imie  as Zawodnik, nazwa_klubu as Klub
        ,data_urodzenia as "Data urodzenia"
from bd3_zawodnicy z , bd3_kluby kl
where z.nr_klubu = kl.nr_klubu
    and data_urodzenia between '87/01/01' and '87/12/31'
    and plec= 'K';
order by "Data urodzenia";
```

lub równoważne mu:

```
select nazwisko || ' ' || imie as Zawodnik, nazwa_klubu as Klub
        ,data_urodzenia  as "Data urodzenia"
from bd3_zawodnicy z join bd3_kluby kl on z.nr_klubu = kl.nr_klubu
 where data_urodzenia between '87/01/01' and '87/12/31'
        and plec= 'K'
order by "Data urodzenia";
```

utworzy zbiór wynikowy:

Zawodnik	Klub	Data urodzenia
Dobkowska Justyna	KB Lechici Zielonka	87/02/19
Butkiewicz Anna	KB Gymnasion Warszawa	87/02/23
Sawicka Jolanta	KB Trucht Warszawa	87/09/11

Gdyby w powyższym przykładzie zaszła potrzeba, aby zbiór wynikowy zawierał kolumnę nr_kategorii – to należy zauważyć, że ta kolumna występuje w dwóch tabelach: BD3_KATEGORIE i BD3_ZAWODNICY. Próba wykonania zdania:

```
select imie, nazwisko, nazwa_klubu, nazwa_kategorii, nr_kategorii
from bd3_zawodnicy z , bd3_kluby kl , bd3_kategorie ka
where z.nr_klubu = kl.nr_klubu and  z.nr_kategorii = ka.nr_kategorii;
```

zakończy się niepowodzeniem:

```
Error report -
SQL Error: ORA-00918: kolumna zdefiniowana w sposób niejednoznaczny
00918. 00000 -  "column ambiguously defined"
```

gdyż nie zostało określone, z której tabeli mają być pobierane dane dotyczące nr_kategorii.

Należy zmodyfikować zdanie poprzez dokładne wyspecyfikowanie tej kolumny nazwą tabeli lub jej aliasem:

```
select imie, nazwisko, nazwa_klubu, nazwa_kategorii, z.nr_kategorii
from bd3_zawodnicy z , bd3_kluby kl , bd3_kategorie ka
where z.nr_klubu = kl.nr_klubu and  z.nr_kategorii = ka.nr_kategorii;
```

W tym przypadku kolumna nr_kategorii będzie przyjmowała wartości z tabeli BD3_ZAWODNICY.

Złączenie zewnętrzne

Złączenie zewnętrzne (*outer join*) polega na dopisaniu do zbioru wynikowego wszystkich wierszy z tabel źródłowych, niezależnie czy istnieją odpowiadające im wiersze w tych tabelach. Jeśli nie można znaleźć odpowiednika danego wiersza w innej tabeli, pola te w zbiorze wynikowym przyjmują wartość *null*.

Istnieją trzy rodzaje złączeń zewnętrznych:

Złączenie zewnętrzne lewostronne:

Złączenie to (*left outer join* lub *left join*) lewostronnie umieszcza w zbiorze wynikowym wszystkie wiersze z pierwszej (lewej) z podanych tabel:

```
tabela1 left outer join tabela2
```

Poniżej dla porównania pokazano to samo zdanie SQL w dwóch sposobach łączenia tabel.

Pierwszy sposób – złączenie wewnętrzne (*join*) omówione wcześniej:

```
select nazwa_klubu, nazwisko
        from bd3_kluby kl
                join bd3_zawodnicy z on z.nr_klubu = kl.nr_klubu
 order by kl.nr_klubu desc;
```

Należy zwrócić uwagę na specyfikację kolumny, według której następuje sortowanie zbioru wynikowego. Kolumna nr_klubu występuje w obu łączonych tabelach i dlatego trzeba precyzyjnie określić, według której ma nastąpić porządkowanie zbioru. Dodatkowo kolumna ta nie występuje w zbiorze wyników (brak jej we frazie *select*)[3].

NAZWA_KLUBU	NAZWISKO
Skra Warszawa	Zatorski
KB Lechici Zielonka	Katłubaj-Domżak
KB Lechici Zielonka	Dobkowska
KB Lechici Zielonka	Karczewski
KB Lechici Zielonka	Popowski

...

[3] Gdyby na liście frazy *select* znajdowała się, odpowiednio wyspecyfikowana, kolumna nr_klubu, wtedy fraza *order by* może zawierać nazwę tej kolumny bez specyfikacji.

Zbiór wyników zawiera zestawienie zawodników, którzy przynależą do jakiegoś klubu czyli w zbiorze wyników nie znajdą się zawodnicy, dla których w tabeli BD3_ZAWODNICY w kolumnie nr_klubu występuje *null.* Liczność tego zbioru wynosi 766, mimo, że w ewidencji jest ich 771.

Drugi sposób – złączenie zewnętrzne lewostronne (*left outer join*):

```
select nazwa_klubu, nazwisko
from bd3_kluby kl
        left outer join bd3_zawodnicy z on z.nr_klubu = kl.nr_klubu
 order by nazwisko nulls first;
```

da następujący wynik:

	NAZWA_KLUBU	NAZWISKO
1	ACTION Warszawa	(null)
2	KU AZS WAT Warszawa	(null)
3	Grunwald Poznań	(null)
4	KB Galeria Warszawa	(null)
5	Flota Gdynia	(null)
6	Śląsk Wrocław	(null)
7	Allianz Warszawa	Adach
8	KB Lechici Zielonka	Adamczyk

…..

Wyraźnie widać, że w zbiorze wynikowym pojawiło się sześć wierszy, których nie było w przypadku łączenia wewnętrznego. Dla tych wierszy kolumny pochodzące z tabeli prawej są uzupełnione wartościami *null*.

Zestawienie pokazuje, że istnieją w bazie kluby, który nie posiadają żadnego zawodnika. Liczność tego zbioru wynikowego wynosi 772 (766 zawodników przynależnych do jakiegoś klubu i dodatkowo sześć klubów bez zawodników).

Złączenie zewnętrzne prawostronne:

Złączenie to (*right outer join* lub *right join*) prawostronnie umieszcza w zbiorze wynikowym wszystkie wiersze z drugiej (prawej) z podanych tabel:

```
tabela1 right outer join tabela2
```

np. wyżej podany przykład złączenia lewostronnego, w którym zmieniono klauzulę *left* na *right* wygląda jak poniżej:

```
select nazwa_klubu, nazwisko
from bd3_kluby kl
        right outer join bd3_zawodnicy z on z.nr_klubu = kl.nr_klubu
 order by kl.nr_klubu desc;
```

i da efekt:

NAZWA_KLUBU	NAZWISKO
(null)	Misiejuk
(null)	Kostecki
(null)	Wereda
(null)	Małkiewicz
(null)	Wojciuk
Skra Warszawa	Zatorski
KB Lechici Zielonka	Misiaczek
KB Lechici Zielonka	Młotkowski

…..

czyli w zbiorze wynikowym znajdą się wszystkie wiersze tabeli BD3_ZAWODNICY (771) uzupełnione o wartości z tabeli BD3_KLUBY, a w przypadku braku odpowiadających wierszy tej tabeli w kolumnie nazwa_klubu pojawią się wartości *null*.

Warto zaznaczyć, że konstrukcja:

tabela1 *left outer join* tabela2

jest równoważna konstrukcji:

tabela2 *right outer join* tabela1

Złączenie zewnętrzne pełne:

Złączenie zewnętrzne pełne (*full outer join* lub *full join*) uwzględnia wszystkie wiersze z obu tabel składowych, wypełniając odpowiednie kolumny wartościami *null* tam, gdzie to konieczne.

tabela1 *full outer join* tabela2

Jeśli tabele składowe łączy związek między kluczem głównym a kluczem obcym, wówczas rezultat tego złączenia będzie pokrywał się z wynikami przeprowadzenia złączenia lewostronnego, w którym tabela zawierająca klucz główny znajduje się po lewej stronie operatora *join* oraz złączenia prawostronnego, w którym tabela ta leży po prawej stronie operatora.
Innymi słowy do zbioru wynikowego przejdą wszystkie wiersze z lewej tabeli oraz wszystkie wiersze z prawej tabeli. Pola wierszy, które nie mogły być połączone będą miały wartość *null*.

Zdanie:

```
select nazwa_klubu, nazwisko
from bd3_kluby kl
full outer join bd3_zawodnicy z on z.nr_klubu = kl.nr_klubu
order by nazwa_klubu nulls first;
```

utworzy zbiór:

NAZWA_KLUBU	NAZWISKO
(null)	Misiejuk
(null)	Kostecki
(null)	Wereda
(null)	Małkiewicz
(null)	Wojciuk
ACTION Warszawa	(null)
Akvedukt Kielce	Walczak
Akvedukt Kielce	Augustyniak

...

Liczność tego zbioru wynosi 777, gdyż jest 771 zawodników w ewidencji i dodatkowo sześć klubów bez zawodników.

Uwaga:
Jeśli w bazie danych relacje między dwiema tabelami (klucz główny i klucz obcy) są tak zdefiniowane, że dopuszcza się wartość *null* na pozycji klucza obcego to należy być tego świadomym konstruując zapytania *select*, aby nie utracić potencjalnie ważnych danych. Taki przypadek zachodzi w implementacji omawianego modelu między tabelami dotyczącymi ewidencji zawodników i ewidencji klubów. Natomiast nie zachodzi w relacji zawodnicy i kategorie wiekowe, gdyż w definicji tabeli

BD3_ZAWODNICY kolumna nr_kategorii nie może przyjmować wartości *null*, co oznacza, że każdy zawodnik musi mieć określoną kategorię wiekową.

Operacja unii

Operacja unii (*union*) tworzy zbiór wynikowy na podstawie dwóch lub większej liczby zdań *select.* W przeciwieństwie do złączenia, które zwiększa liczbę kolumn w zbiorze wynikowym, operacja unii zmienia liczbę wierszy.

Np.:

```
select nazwisko || ' ' || imie as "Nazwisko i imię", rezultat_min, rezultat_sek, nazwa_klubu, plec
from bd3_zawodnicy z, bd3_wyniki w, bd3_kluby k
where z.nr_klubu = k.nr_klubu and z.nr_zawodnika = w.nr_zawodnika
        and w.nr_zawodow = 1 and plec = 'M' and k.nr_klubu = 4

union

select nazwisko || ' ' || imie as "Nazwisko i imię", rezultat_min, rezultat_sek, nazwa_klubu, plec
from bd3_zawodnicy z, bd3_wyniki w, bd3_kluby k
where z.nr_klubu = k.nr_klubu and z.nr_zawodnika = w.nr_zawodnika
        and w.nr_zawodow = 1 and plec = 'K' and k.nr_klubu = 20

order by 5 desc, 2,3;
```

Powyższa konstrukcja składa się z czterech części.

Pierwsze zdanie *select* tworzy zbiór wynikowy zawierający wyspecyfikowane kolumny w celu pokazania rezultatów zawodników w zawodach o numerze 1 będących mężczyznami należącymi do klubu o identyfikatorze 4. Pozostałe warunki ograniczające we frazie *where* służą do zdefiniowania odpowiedniego połączenia tabel.

Drugim elementem jest fraza *union* lub *union all*.
W dalszej części występuje drugie zdanie *select* tworzące zbiór wynikowy zawierający te same kolumny jak w zdaniu pierwszym, ale z innymi ograniczeniami. Tym razem wybrane zostaną kobiety biorące udział w zawodach o numerze 1, które należą do klubu o numerze 20.

Końcowym elementem, który może (ale nie musi) wystąpić jest sposób sortowania. W przykładzie zastosowano skróconą formę zapisu frazy *order by* zamieniając nazwy kolumn na numery określające miejsce ich występowania w zdaniu *select*.

Nazwisko i imię	REZULTAT_MIN	REZULTAT_SEK	NAZWA_KLUBU	PLEC
Ławecki Tomasz	37	22	Legia Warszawa	M
Harkot Marta	59	48	KB Lotos Jabłonna	K
Gołąbek Agnieszka	63	5	KB Lotos Jabłonna	K

Uwagi:

1. W przypadku, gdy zbiory wynikowe obu zdań *select* mają powtarzające się wiersze (co nie zachodzi w powyższym przykładzie, gdyż mamy do czynienia z dwoma rozłącznymi zbiorami z uwagi na płeć) fraza *union* automatycznie je wyeliminuje czyli w tle zostanie zastosowana klauzula *distinct*. Jeżeli chcemy, aby w końcowym zbiorze wynikowym występowały powtarzające się wiersze należy zastosować frazę *union all*.
2. W obu zdaniach *select* liczba kolumn musi być taka sama, jak również musi zachodzić zgodność typów lub przynajmniej możliwość konwersji na jeden rodzaj typu.
3. Frazy *order by* można użyć tylko raz na końcu unii i nazwy kolumn w niej zawarte muszą występować na liście kolumn pierwszego zdania *select*.
4. Unia może zawierać większą liczbę zdań *select* połączonych frazą *union* lub *union all*.

Zadania do samodzielnego wykonania:

Poniższe zadania wykonać przy następujących założeniach:

- We wszystkich zadaniach pobierać dane z minimum trzech tabel,
- Kolumny Imię i Nazwisko łączyć w jedną kolumnę,
- Stosować aliasy do zmiany nazw kolumn,
- Zestawienia powinny zawierać takie kolumny jak Nazwisko i imię, Nazwa klubu, Nazwa kategorii.

1. Opracować raport zawierający zestawienie mężczyzn, którzy brali udział w zawodach numer 1 i 3 i uzyskali wynik między 44 i 48 min (kolumny Rezultat_Min i Rezultat_Sek). Posortować zbiór według numeru zawodów i osiągniętego czasu.

2. Opracować zestawienie kobiet, które należą do klubów o numerach od 5 do 40, urodzone są w latach między 1975 i 1984 i które nie zdobyły punktów w klasyfikacji generalnej (kolumna Punkty_Globalne).

3. Opracować komunikat końcowy zawodów numer 3 pokazujący klasyfikację zawodników (mężczyzn) na mecie według osiągniętych czasów. Wykorzystać kolumnę Lokata_W_Biegu.

4. Przy pomocy unii zaprojektować raport pokazujący zestawienie mężczyzn należących do kategorii IV i kobiet urodzonych w latach 1970 – 1985, których nazwiska zaczynają się na literę K i kończą na ‘ska’. Posortować zbiór według roku urodzenia i nazwiska.

5. Wykonać zadanie z pkt. 4 korzystając z jednego zdania *select*.

6. Dokonać modyfikacji modelu bazy danych. O każdym klubie należy przechowywać informacje teleadresowe, rok założenia klubu, nazwisko prezesa klubu itp. Dane te należy zapisywać w osobnej tabeli. Dokonać odpowiedniej modyfikacji relacji między tabelami poprzez napisanie skryptu realizującego te modyfikacje. Wprowadzić przykładowe dane dla kilku klubów i napisać zdanie wybierające informacje z bazy przy uwzględnieniu nowej tabeli.
7. Dokonać modyfikacji modelu bazy danych. O każdym zawodniku trzeba przechowywać informacje o jego osiągnięciach w postaci wpisów do osobnej tabeli (np. rok i zdarzenie) przy założeniu, że jeden zawodnik może mieć kilka wpisów. Wpisy dotyczą historii zawodnika (np. zmiana barw klubowych, wyniki sportowe). Dokonać odpowiedniej modyfikacji relacji między tabelami poprzez napisanie skryptu realizującego te modyfikacje. Wprowadzić przykładowe dane dla kilku zawodników i napisać zdanie wybierające informacje z bazy przy uwzględnieniu nowej tabeli.

8. Dokonać modyfikacji modelu bazy danych tak, aby niemożliwe było zarejestrowanie zawodnika bez przynależności klubowej. Utworzyć fikcyjny klub o nazwie Niezrzeszeni i w nim umieścić dotychczasowych zawodników bez przydziału.

9. Dokonać modyfikacji modelu bazy danych polegającej na konieczności automatycznego rejestrowania osoby i czasu wprowadzenia danych o zawodniku do ewidencji oraz ewentualnej ich modyfikacji. Należy wykorzystać systemowe funkcje USER i SYSDATE.

10. W procesie wczytywania danych do tabel z plików płaskich przy pomocy SQL Developer można było utworzyć skrypt zawierający kompletne zdania *insert* wybierając metodę *Insert Script*. Skrypty te były tworzone automatycznie w odpowiednim folderze systemu operacyjnego (c:\users\%user%\AppData\Local\Temp). Należy, na ich podstawie utworzyć jeden zbiorczy skrypt zawierający wszystkie zdania *insert* niezbędne do wprowadzenia danych do wszystkich tabel, zakończyć go zatwierdzeniem transakcji oraz zdaniem sprawdzającym zawartość tabel. Nazwać tak powstały skrypt *lab_BD3_populate.sql*, a następnie dokonać globalnego testowania poprawności wdrożenia przy pomocy sekwencji poleceń:
    @c:\temp\lab_BD3_drop.sql
    @c:\temp\lab_BD3_create.sql
    @c:\temp\lab_BD3_populate.sql