Aleksandra Kukawka
300246

PROJEKT – ETAP II

Temat: Liga piłkarska

SPIS TREŚCI

KRÓTKI OPIS ROZWIĄZANIA

Tematem mojej bazy danych jest (opisana szczegółowo w etapie I) organizacja rozgrywek polskiej Ekstraklasy piłkarskiej w sezonie 2018/2019 trwających od 20.07.2018 do 19.05.2019.

Głównym celem bazy jest rozstrzygnięcie, które drużyny odpadną z ligi, a które w niej zostaną. Najważniejsze jest wyłonienie zwycięzcy całej Ekstraklasy, które odbywa się automatycznie na podstawie wyliczeń związanych ze zdobytymi bramkami klubów. Innymi celami są: wyłonienie króla strzelców poprzez możliwość aktualizowania liczby bramek poszczególnych zawodników po meczach, szybkie transfery zawodników do innych klubów oraz podejrzenie liczby meczów klubów rozegranych do tej pory.

Wszystkie procedury oraz funkcje dotyczące operacji na danych związanych z zawodnikami znajdują się w pakiecie *zawodnicy_management*. Natomiast te, które związane są z organizacją meczów, znajdują się w pakiecie *mecze_management*.

Skrypty do stworzenia schematu bazy danych

Skrypty do stworzenia schematu mojej bazy danych znajdują się w pliku *create_tables.ddl*.

Skrypty do załadowania danych

Skrypty służące do załadowania danych znajdują się w pliku *insert_data.ddl*. Imiona i nazwiska trenerów oraz zawodników i dane w tabelach Kluby oraz Stadiony są autentyczne. Pozostałe dane są fikcyjne.

UWAGI:

- W pliku *insert_data.ddl* znajduje się także wstawienie danych do tabeli Mecze. Wstawianie tych danych umożliwia procedura dodaj_mecz() oraz sekwencja mecze_seq, których definicje znajdują się w pliku *mecze_management.ddl*.
- Dodatkowo w pliku *insert_data.ddl.* znajduje się dodawanie bramek zawodnikom. Do tego służy procedura dodaj_bramki(), której definicja znajduje się w pliku *zawodnicy_management.ddl.*

Sekwencje

Podczas wstawiania danych do tabel poprzez INSERT wykorzystałam 4 stworzone przeze mnie sekwencje:

- trenerzy_seq – identyfikatory trenerów (pierwszy id: 341)
- stadiony_seq – identyfikatory stadionów (pierwszy id: 101)
- kluby_seq – identyfikatory klubów (pierwszy id: 901)
- zawodnicy_seq – identyfikatory zawodników (pierwszy id: 1001)

Sędziowie nie mają swojej własnej sekwencji dlatego, że ich identyfikator to numer licencji – losowa kombinacja 4-5 liczb/cyfr.

Dodatkowo użyłam sekwencji mecze_seq, która ułatwia dodanie meczu do tabeli mecze za pomocą opisanej w dalszej części procedury dodaj_mecz().

Wyzwalacze

Zdefiniowałam kilka wyzwalaczy, które zapobiegają wprowadzeniu błędnych danych i tym samym umożliwiają prawidłowe działanie oraz trafną interpretację i analizę wyników konkretnych zapytań. Definicje wyzwalaczy znajdują się w pliku *triggers.ddl.*

- wstaw_zawodnik_trigger
  Funkcja: Wprowadzenie do bazy nowego zawodnika z liczbą strzelonych bramek różną od 0 skutkuje zamianą tej nieprawidłowej liczby na 0. Nowy zawodnik, dopiero co wprowadzony do bazy (dokładniej – do klubu) nie może mieć liczby strzelonych bramek różnej od 0, ponieważ nie zagrał dotychczas w żadnym meczu.

```
CREATE OR REPLACE TRIGGER wstaw_zawodnik_trigger
BEFORE INSERT
ON zawodnicy
FOR EACH ROW
    WHEN (new.strzelone_bramki != 0)
BEGIN
    :new.strzelone_bramki := 0;
END;

ALTER TRIGGER wstaw_zawodnik_trigger ENABLE;
```

- zmien_zawodnik_trigger
  Funkcja: Próba zaktualizowania liczby strzelonych bramek któregoś z zawodników na liczbę ujemną skutkuje wyświetleniem komunikatu o błędzie. Zawodnik nie może zdobyć ujemnej liczby goli.

```
CREATE OR REPLACE TRIGGER zmien_zawodnik_trigger
BEFORE UPDATE OF strzelone_bramki
ON zawodnicy
FOR EACH ROW
    WHEN(new.strzelone_bramki < 0)
BEGIN
    :new.strzelone_bramki := :old.strzelone_bramki;
    dbms_output.put_line('Nie można zmienić liczby strzelonych bramek zawodnika o id ' || :old.id_zawodnika || ' na liczbę ujemna!');
END;

ALTER TRIGGER zmien_zawodnik_trigger ENABLE;
```

- zmien_klub_trigger
  Funkcja: To samo dotyczy punktów danego klubu. Próba zaktualizowania liczby punktów jakiegoś z klubów na liczbę ujemną skutkuje wyświetleniem komunikatu o błędzie. Klub nie może zdobyć ujemnej liczby punktów.

```
CREATE OR REPLACE TRIGGER zmien_klub_trigger
BEFORE UPDATE OF punkty
ON kluby
FOR EACH ROW
    WHEN(new.punkty < 0)
BEGIN
    :new.punkty := :old.punkty;
    dbms_output.put_line('Nie można zmienić liczby punktów klubu o id ' || :old.id_klubu || ' na liczbę ujemna!');
END;

ALTER TRIGGER zmien_klub_trigger ENABLE;
```

Procedury

Zdefiniowałam kilka procedur, które mają kluczowe zadanie w działaniu bazy danych. Pozwalają na wprowadzenie przez użytkownika nowego meczu, transfer zawodników między klubami, czy dodanie któremuś z zawodników strzelonych goli. Definicje procedur znajdują się w pakiecie *zawodnicy_management* i *mecze_management*.

- dodaj_mecz
  Procedura umożliwiająca dodanie nowego meczu do tabel. Użytkownik podaje:
  - id klubu, który jest gospodarzem
  - strzelone bramki gospodarza
  - id klubu, który jest gościem
  - strzelone bramki gościa
  - sędziego (jego nr licencji)
  - numer aktualnej kolejki meczu

  Dodatkowym ułatwieniem jest tutaj stworzona sekwencja mecze_seq, którą opisałam w części „Sekwencje". Sprawdzane jest czy mecz faktycznie się odbył (drużyna grała już mecz w danej kolejce), czy ten sam mecz był już wcześniej wpisany w tabeli oraz czy użytkownik nie podał ujemnej liczby bramek. Procedura znajduje się w pliku (i w pakiecie) *mecze_management.ddl*.

```
CREATE OR REPLACE PROCEDURE dodaj_mecz
(
    dm_id_gospodarza kluby.id_klubu%TYPE,
    dm_bramki_gospodarza NUMBER,
    dm_id_goscia kluby.id_klubu%TYPE,
    dm_bramki_goscia NUMBER,
    dm_nr_licencji_sedziego sędziowie.nr_licencji%TYPE,
    dm_nr_kolejki mecze.kolejka%TYPE
)
AS
    v_mecz_istnieje NUMBER := 0;
    v_id_meczu mecze.id_meczu%TYPE := mecze_seq.NEXTVAL;
    v_id_stadionu stadiony.id_stadionu%TYPE := 0;
BEGIN

    --sprawdzenie, na którym stadionie odbyl się mecz na podstawie
    --podanego id gospodarza (mecz zawsze odbywa się na stadionie
    -- gospodarza)
    SELECT id_stadionu
    INTO v_id_stadionu
    FROM kluby
    WHERE id_klubu = dm_id_gospodarza;

    --sprawdzenie, czy rekord dotyczacy danego meczu (jego nr kolejki i
    --id stadionu [tym samym id gospodarza] w zupełności wystarczy, ponieważ
    --podczas jednej kolejki klub może być gospodarzem tylko raz)
    SELECT count(*)
    INTO mecz_istnieje
    FROM mecze m
    WHERE m.kolejka = dm_nr_kolejki AND m.id_stadionu = v_id_stadionu;

   --sprawdzenie, czy podany mecz w ogóle ma prawo bytu, ponieważ każdy klub
   --w każdej kolejce powinien grać tylko jeden mecz
   SELECT count(*)
   INTO v_mecz_nie_istnieje
   FROM mecze_kluby mk
   JOIN mecze m
   ON mk.id_meczu = m.id_meczu
   WHERE m.kolejka = dm_nr_kolejki AND mk.id_klubu IN (dm_id_gospodarza, dm_id_goscia);


   IF v_mecz_nie_istnieje = 1
       THEN dbms_output.put_line('Podany mecz nie odbyl się! Któraś z podanych drużyn zagrala już mecz w tej kolejce. ');

   ELSIF v_mecz_istnieje = 1
       THEN dbms_output.put_line('Mecz jest już w tabeli.');

   ELSE
       --wstawienie calego rekordu do tabeli mecze, wraz z numerem kolejki, punktami zdobytymi przez gospodarza i gościa,
       --numerem licencji sędziego oraz stadionem
       IF dm_bramki_gospodarza >= 0 AND dm_bramki_goscia >=0
       THEN
           INSERT INTO mecze VALUES(v_id_meczu, dm_nr_kolejki, dm_bramki_gospodarza, dm_bramki_goscia, dm_nr_licencji_sedziego, v_id_stadionu, 0);

           --dodanie meczu do tabeli mecze_klubu
           INSERT INTO mecze_kluby VALUES(dm_id_gospodarza, v_id_meczu);
           INSERT INTO mecze_kluby VALUES(dm_id_goscia, v_id_meczu);

       ELSE
           dbms_output.put_line('Nie można podać ujemnej liczby bramek!');
       END IF;
   END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Sprawdz ponownie wprowadzone dane.');
END dodaj_mecz;

END mecze_management;
```

- dodaj_bramki

  Procedura dodająca bramki, które zawodnik uzyskał w danym meczu. Potrzebne jest to, aby wyłonić króla strzelców. Procedura dokonuje aktualizacji pola „strzelone_bramki" w tabeli zawodnicy. Na początku sprawdzane jest, czy zawodnik faktycznie jest w jednej z drużyn, która brała udział w meczu o podanym id. Następnie sprawdzana jest poprawność wprowadzonej liczby bramek (czy jest ona większa od 0). Procedura znajduje się w pliku (i w pakiecie) *zawodnicy_management.ddl*.

```
CREATE OR REPLACE PROCEDURE dodaj_bramki
(
    db_id_zawodnika zawodnicy.id_zawodnika%TYPE,
    db_strzelone_bramki zawodnicy.strzelone_bramki%TYPE,
    db_id_meczu mecze.id_meczu%TYPE
)
AS
    v_informacje_prawidlowe NUMBER := 0;
    v_id_klubu_zawodnika kluby.id_klubu%TYPE;
BEGIN
    --sprawdzenie, z którego klubu powinien być dany zawodnik
    --na podstawie jego identyfikatora
    SELECT k.id_klubu
    INTO v_id_klubu_zawodnika
    FROM kluby k
    JOIN zawodnicy z
    ON k.id_klubu = z.id_klubu
    WHERE z.id_zawodnika = db_id_zawodnika;

    --sprawdzenie, czy zawodnik faktycznie gra w danym meczu
    SELECT count(*)
    INTO v_informacje_prawidlowe
    FROM mecze_kluby
    WHERE id_meczu = db_id_meczu AND id_klubu = v_id_klubu_zawodnika;


    IF v_informacje_prawidlowe = 1
        --zaktualizowanie rekordu w tabeli zawodnicy, dodanie do aktualnej liczby
        --strzelonych bramek tych, które zostały wprowadzone przez użytkownika
        --poprzez procedurę

        THEN
            --można tylko dodać bramki, nie można ich odjąć
            IF db_strzelone_bramki > 0
                THEN
                    UPDATE zawodnicy
                    SET strzelone_bramki = strzelone_bramki + db_strzelone_bramki
                    WHERE id_zawodnika = db_id_zawodnika;
            ELSE
                dbms_output.put_line('Nie można podać ujemnej liczby strzelonych bramek!');
            END IF;
    ELSE
        dbms_output.put_line('Zawodnik nie należy do żadnego z klubów, które grały w podanym meczu.');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Sprawdz ponownie wprowadzone dane.');
END dodaj_bramki;
```

- transfer_zawodnika

  Procedura umożliwiająca transfer zawodnika do innego klubu, poprzez podanie jego id i id klubu, do którego ma zostać przetransferowany. Jeśli zawodnik już jest w danym klubie, zostanie wyświetlony komunikat. Procedura znajduje się w pliku (i w pakiecie) *zawodnicy_management.ddl*.

```
CREATE OR REPLACE PROCEDURE transfer_zawodnika
(
    tz_id_zawodnika zawodnicy.id_zawodnika%TYPE,
    tz_dokad kluby.id_klubu%TYPE
)
AS
    v_zawodnik_istnieje NUMBER := 0;
BEGIN
    --sprawdzenie czy dany zawodnik jest już w danym klubie
    SELECT count (*)
    INTO v_zawodnik_istnieje
    FROM zawodnicy
    WHERE id_klubu = tz_dokad AND id_zawodnika = tz_id_zawodnika;


    IF v_zawodnik_istnieje = 1
        THEN dbms_output.put_line('Zawodnik już należy do tego klubu.');

    ELSE
        UPDATE zawodnicy
        SET id_klubu = tz_dokad
        WHERE id_zawodnika = tz_id_zawodnika;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Sprawdz ponownie wprowadzone dane.');
END;
```

Funkcje

Dodatkowo zdefiniowałam kilka funkcji. Nie wszystkie z nich są niezbędne do poprawnego działania mojej bazy danych, ale zapewniają kilka dodatkowych i ciekawych możliwości Definicje funkcji znajdują się w pakietach *zawodnicy_management* i *mecze_management*.

- król_strzelców()

  Funkcja, która wyszukuje zawodnika z największą ilością strzelonych bramek. Jest to bardzo prosta i podstawowa funkcja, gdyż głównie bazuje na procedurze dodaj_bramki() – bez niej próba wyłonienia króla strzelców nie miałaby najmniejszego sensu. Funkcja znajduje się w pliku (i w pakiecie) *zawodnicy_management.ddl*.

```
CREATE OR REPLACE FUNCTION krol_strzelcow
RETURN zawodnicy.id_zawodnika%TYPE
AS
v_krol zawodnicy.id_zawodnika%TYPE;
BEGIN
    SELECT id_zawodnika
    INTO v_krol
    FROM (SELECT *
          FROM zawodnicy
          ORDER BY strzelone_bramki DESC, id_zawodnika)
    WHERE ROWNUM <=1;

    return(v_krol);
END;
```

- wylicz_punkty()

  Najbardziej skomplikowana funkcja, która na podstawie tabel Mecze i Mecze_kluby wylicza punkty uzyskane do tej pory dla każdej z drużyn. Korzysta z kursora. Na początku sprawdzane jest, czy mecz jest już rozliczony. Właśnie do tego celu służy pole „Rozliczony" w tabeli Mecze, które przyjmuje wartość 0 – gdy mecz jeszcze nie został rozliczony. Funkcja zwraca klub z aktualnie najwyższą ilością punktów. Funkcja znajduje się w pliku (i w pakiecie) *mecze_management.ddl*.

```
FUNCTION wylicz_punkty
RETURN kluby.id_klubu%TYPE
AS
v_zwyciezca kluby.id_klubu%TYPE;
id_gościa kluby.id_klubu%TYPE;

--Aby użyć operacji UPDATE w funkcji i użyć tej funkcji testujac ja w zapytaniu SELECT
--trzeba użyć PRAGMA AUTONOMOUS_TRANSACTION, a następnie "zcommitować" zmiany
pragma autonomous_transaction;

CURSOR c IS
    SELECT m.id_meczu, m.punkty_gospodarza, m.punkty_gościa, m.rozliczony, k.id_klubu as id_gospodarza
    FROM mecze m, kluby k
    WHERE m.id_stadionu = k.id_stadionu;
BEGIN
    FOR i in c
    LOOP
        --jeśli wartość pola rozliczony jest różna od zera, to znaczy, że drużyny dostały
        --już punkty za ten mecz
        IF i.rozliczony != 0
        THEN
            dbms_output.put_line('Mecz o id' || i.id_meczu || 'zostal juz rozliczony');

        --w przeciwnym wypadku rozpoczyna się wyliczanie punktów dla danego meczu
        --zwycięska drużyna otrzymuje 3 punkty, przegrana drużyna 0
        --jeśli jest remis - każda z drużyn otrzymuje po 1 punkcie
        ELSE
            --jeżeli wygral gospodarz, to sprawa jest prosta,
            --bo do id_gospodarza jest bezpośredni dostęp z kursora
            IF i.punkty_gospodarza > i.punkty_gościa
            THEN
                UPDATE kluby
                SET punkty = punkty + 3
                WHERE id_klubu = i.id_gospodarza;

            --w przeciwnym wypadku wiadomo, że gościowi też trzeba dopisać
            --punkt lub 3, dlatego teraz zajmę się wyluskaniem z moich danych
            --(a dokladniej z tabeli Mecze_kluby) id_gościa
            ELSE
```

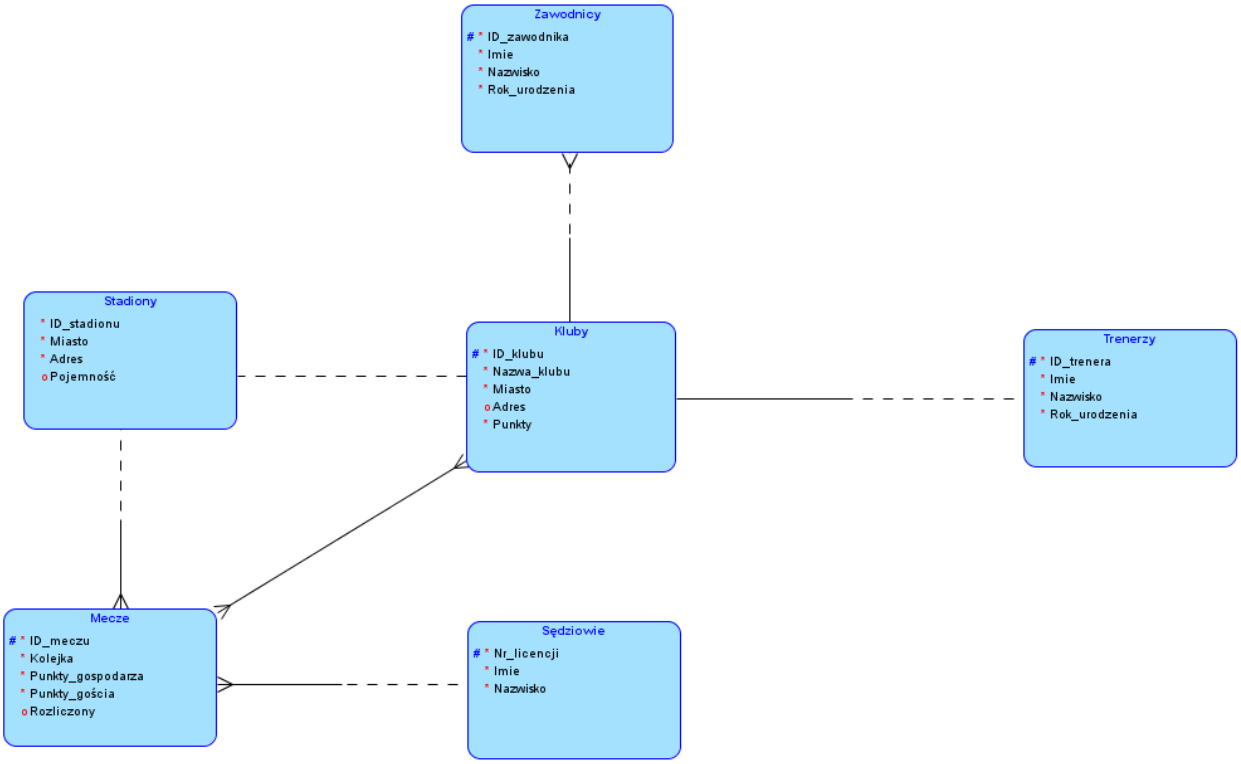
```
            ELSE
                SELECT id_klubu
                INTO id_gościa
                FROM mecze_kluby
                WHERE id_meczu = i.id_meczu AND id_klubu != i.id_gospodarza;

                --teraz, majac zarówno id_gospodarza, jak i id_gościa, można
                --przejść do sprawdzenia kolejnych opcji wyniku meczu

                --jeśli jest remis
                IF i.punkty_gospodarza = i.punkty_gościa
                THEN
                    UPDATE kluby
                    SET punkty = punkty + 1
                    WHERE id_klubu IN (i.id_gospodarza, id_gościa);

                --zostaje tylko opcja, że wygral gość
                ELSE
                    UPDATE kluby
                    SET punkty = punkty + 3
                    WHERE id_klubu = id_gościa;
                END IF;

            END IF;
        END IF;
    END LOOP;

    --niezależnie od wyniku wszystkie mecze zostay on obsłużone, mecz można uznać
    --wszystkie za rozliczone
    UPDATE mecze
    SET rozliczony = 1
    WHERE rozliczony = 0;
    commit;

    --zwrocenie id_klubu z najwieksza ilościa dotychczas zebranych punktów
    SELECT id_klubu
    INTO v_zwyciezca
    FROM (SELECT *
          FROM kluby
          ORDER BY punkty DESC, id_klubu)
    WHERE ROWNUM <=1;

    return v_zwyciezca;

END wylicz_punkty;
```

- ile_meczów()

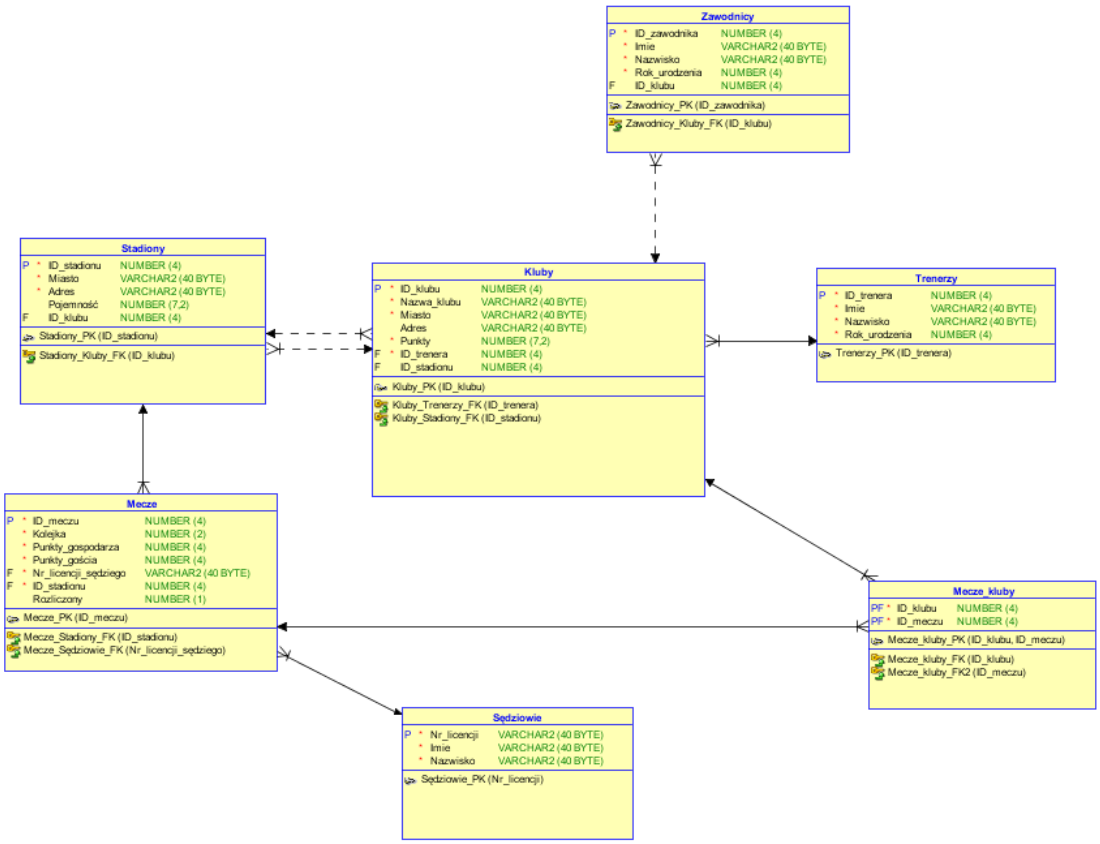
  Funkcja, która na podstawie tabeli Mecze_kluby wyszukuje klub, który zagrał największą ilość meczów do tej pory. Jeśli kilka klubów zagrało taką samą ilość meczów, to zostanie zwrócony ten, który ma mniejsze id. Dodatkowo po kolei wyświetlana jest liczba meczów każdego klubu. Funkcja znajduje się w pliku (i w pakiecie) *mecze_management.ddl*.

```
FUNCTION ile_meczow
(
    im_id_klubu kluby.id_klubu%TYPE
)
RETURN NUMBER
AS
v_najwiecej_meczow_klub kluby.id_klubu%TYPE;

CURSOR c IS
    SELECT k.id_klubu, k.nazwa_klubu, count(UNIQUE mk.id_meczu) as liczba_meczow
    FROM mecze_kluby mk
    JOIN kluby k
    ON k.id_klubu = mk.id_klubu
    GROUP BY k.id_klubu, k.nazwa_klubu
    ORDER BY nazwa_klubu;

BEGIN
    FOR i in c
    LOOP
        dbms_output.put_line('Klub ' || i.nazwa_klubu || ' zagral do tej pory ' || i.liczba_meczow || ' meczów.');
    END LOOP;

    SELECT id_klubu
    INTO v_najwiecej_meczow_klub
    FROM (
            SELECT id_klubu, count(UNIQUE id_meczu) as liczba_meczow
            FROM mecze_kluby
            GROUP BY id_klubu
            ORDER BY liczba_meczow DESC)
    WHERE ROWNUM <= 1;

    return v_najwiecej_meczow_klub;

END ile_meczow;
```

MODEL ER



MODEL RELACYJNY



SKRYPTY TESTUJĄCE

Aby włączyć konkretny wyzwalacz należy użyć:

```
ALTER TRIGGER nazwa_wyzwalacza ENABLE;
```

- Testowanie wyzwalacza wstaw_zawodnik_trigger
  - Wstawienie zawodnika z liczbą strzelonych bramek równą 0:

```
INSERT INTO zawodnicy VALUES(zawodnicy_seq.NEXTVAL, 'Rafal', 'Adamski', 1988, 912, 0);
```

1427	Rafal	Adamski	1988	912	0

  - Próba wstawienia zawodnika z liczbą strzelonych bramek różną od 0 – wyzwalacz wyzerował pole strzelone_bramki:

```
INSERT INTO zawodnicy VALUES(zawodnicy_seq.NEXTVAL, 'Patryk', 'Makuch', 1988, 912, 20);
```

1428	Patryk	Makuch	1988	912	0

- Testowanie wyzwalacza zmien_zawodnik_trigger
  - Zaktualizowanie liczby strzelonych bramek zawodnika na 20:

```
UPDATE zawodnicy
SET strzelone_bramki = 20
WHERE id_zawodnika = 1001;
```

1001	Krystian	Stepniowski	1992	904	20

  - Próba zaktualizowania liczby strzelonych bramek zawodnika na -20 – wyświetlony zostaje komunikat o błędzie, liczba bramek nie została zmieniona:

```
UPDATE zawodnicy
SET strzelone_bramki = -20
WHERE id_zawodnika = 1002;
```

```
Nie można zmienić liczby strzelonych bramek zawodnika o id 1002 na liczbę ujemna!
```

1002	Kacper	Chorazka	1999	904	0

- Testowanie wyzwalacza zmien_klub_trigger
  - Zaktualizowanie liczby punktów klubu na 10:

```
UPDATE kluby
SET punkty = 10
WHERE id_klubu = 901;
```

901	Arka Gdynia	Gdynia	Olimpijska 5/9	10	353	113

- Próba zaktualizowania liczby punktów klubu na liczbę ujemną – wyświetlony zostaje komunikat o błędzie, liczba punktów nie została zmieniona:

```
UPDATE kluby
SET punkty = -10
WHERE id_klubu = 902;
```

```
Nie można zmienić liczby punktów klubu o id 902 na liczbę ujemna!
```

902	Korona Kielce	Kielce	Nowowiejska 1	0	350	110

- Testowanie procedury dodaj_mecz()
  - Dodanie meczu, który odbył się naprawdę i którego nie ma jeszcze w tabeli:

```
BEGIN
    dodaj_mecz(913, 0, 906, 1, 'h36d', 1);
END;
```

    - Tabela Mecze:

2008	1	0	1	h36d	105

    - Tabela Mecze_kluby:

913	2008
906	2008

  - Próba dodania meczu, który już istnieje w tabeli:

```
BEGIN
    dodaj_mecz(913, 0, 906, 1, 'h36d', 1);
END;
```

```
Mecz jest już w tabeli.
```

  - Próba dodania meczu, który nie miał prawa się odbyć w danej kolejce (tj. jakiś klub gra mecz po raz drugi w danej kolejce):

```
BEGIN
    dodaj_mecz(913, 0, 911, 1, 'h36d', 1);
END;
```

```
Podany mecz nie odbyl się! Któraś z podanych drużyn zagrala już mecz w tej kolejce.
```

  - Próba dodania meczu z ujemną liczbą bramek gospodarza i/lub gościa:

```
BEGIN
    mecze_management.dodaj_mecz(910, -2, 905, 1, 'h36d', 5);
END;
```

```
Nie można podać ujemnej liczby bramek!
```

- Testowanie procedury dodaj_bramki()
  - Dodanie bramek zawodnikowi, który grał w danym meczu:

```
BEGIN
    zawodnicy_management.dodaj_bramki(1428, 3, 2000);
END;
```

1428	Patryk	Makuch	1988	912	3

  - Próba dodania bramek zawodnikowi, który nie grał w danym meczu:

```
BEGIN
    dodaj_bramki(1234, 2, 2023);
END;
```

```
Zawodnik nie należy do żadnego z klubów, które graly w podanym meczu.
```

  - Próba dodania ujemnej liczby bramek zawodnikowi

```
BEGIN
    zawodnicy_management.dodaj_bramki(1428, -5, 2000);
END;
```

```
Nie można podać ujemnej liczby strzelonych bramek!
```

- Testowanie procedury transfer_zawodnika()
  - Transfer zawodnika do rzeczywiście innego klubu:

1016	Dominik	Furman	1984	909	0

```
BEGIN
    transfer_zawodnika(1016,912);
END;
```

1016	Dominik	Furman	1984	912	0

  - Próba transferu zawodnika do klubu, w którym jest aktualnie:

```
BEGIN
    transfer_zawodnika(1016,912);
END;
```

```
Zawodnik już należy do tego klubu.
```

- Testowanie funkcji król_strzelców()

```
SELECT *
FROM zawodnicy
WHERE id_zawodnika = (SELECT krol_strzelcow()
                      FROM dual);
```

ID_ZAWODNIKA	IMIE	NAZWISKO	ROK_URODZENIA	ID_KLUBU	STRZELONE_BRAMKI
1428	Patryk	Makuch	1988	912	3

- Testowanie funkcji wylicz_punkty()

- Tabela Kluby przed, wywołanie funkcji, tabela Kluby po oraz wyłoniony zwycięzca:

ID_KLUBU	NAZWA_KLUBU	MIASTO	ADRES	PUNKTY	ID_TRENERA	ID_STADIONU
901	Arka Gdynia	Gdynia	Olimpijska 5/9	0	353	113
902	Korona Kielce	Kielce	Nowowiejska 1	0	350	110
903	Wisła Kraków	Kraków	Reymonta 15	0	349	109
904	Zagłębie Sosnowiec	Sosnowiec	Kresowa 3	0	356	116
905	KGHM Zagłębie Lubin	Lubin	Marii Skłodowskiej-Curie 98	0	356	106
906	Lechia Gdańsk	Gdańsk	Pokoleń Lechii Gdańsk 3	0	343	104
907	Lech Poznań	Poznań	Bułgarska 17	0	348	108
908	Pogoń Szczecin	Szczecin	Biernacka 7	0	347	107
909	Wisła Płock	Płock	Łukasiewicza 34	0	354	114
910	Cracovia	Kraków	Nieznana 18	0	344	103
911	Legia Warszawa	Warszawa	Łazienkowska 3	0	342	102
912	Miedź Legnica	Legnica	Hetmańska 4	0	355	115
913	Jagiellonia Białystok	Białystok	Słoneczna 1	0	345	105
914	Górnik Zabrze	Zabrze	Roosevelta 83	0	351	111
915	Śląsk Wrocław	Wrocław	al. Śląska 1	0	352	112
916	Piast Gliwice	Gliwice	Okrzei 22	0	341	101

```
SELECT nazwa_klubu
FROM kluby
WHERE id_klubu = (SELECT mecze_management.wylicz_punkty()
                  FROM dual);
```

ID_KLUBU	NAZWA_KLUBU	MIASTO	ADRES	PUNKTY	ID_TRENERA	ID_STADIONU
901	Arka Gdynia	Gdynia	Olimpijska 5/9	3	353	113
902	Korona Kielce	Kielce	Nowowiejska 1	4	350	110
903	Wisła Kraków	Kraków	Reymonta 15	5	349	109
904	Zagłębie Sosnowiec	Sosnowiec	Kresowa 3	3	356	116
905	KGHM Zagłębie Lubin	Lubin	Marii Skłodowskiej-Curie 98	6	356	106
906	Lechia Gdańsk	Gdańsk	Pokoleń Lechii Gdańsk 3	5	343	104
907	Lech Poznań	Poznań	Bułgarska 17	12	348	108
908	Pogoń Szczecin	Szczecin	Biernacka 7	1	347	107
909	Wisła Płock	Płock	Łukasiewicza 34	2	354	114
910	Cracovia	Kraków	Nieznana 18	2	344	103
911	Legia Warszawa	Warszawa	Łazienkowska 3	7	342	102
912	Miedź Legnica	Legnica	Hetmańska 4	3	355	115
913	Jagiellonia Białystok	Białystok	Słoneczna 1	12	345	105
914	Górnik Zabrze	Zabrze	Roosevelta 83	3	351	111
915	Śląsk Wrocław	Wrocław	al. Śląska 1	4	352	112
916	Piast Gliwice	Gliwice	Okrzei 22	9	341	101

NAZWA_KLUBU
Lech Poznań

Kluby Lech Poznań i Jagiellonia Białystok mają taką samą ilość punktów, lecz został wyświetlony pierwszy z nich, ponieważ (zgodnie z założeniami) ma on mniejszy identyfikator.

- Testowanie funkcji ile_meczów() (ponieważ wiele klubów ma tę samą liczbę meczy, funkcja zwróciła ten, który miał najniższe id)

```
SELECT nazwa_klubu
FROM kluby
WHERE id_klubu = (SELECT mecze_management.ile_meczow()
                  FROM dual);
```

	NAZWA_KLUBU
1	Zagłębie Sosnowiec

```
Klub Arka Gdynia zagral do tej pory 4 meczów.
Klub Cracovia zagral do tej pory 4 meczów.
Klub Górnik Zabrze zagral do tej pory 4 meczów.
Klub Jagiellonia Białystok zagral do tej pory 4 meczów.
Klub KGHM Zagłębie Lubin zagral do tej pory 4 meczów.
Klub Korona Kielce zagral do tej pory 3 meczów.
Klub Lech Poznań zagral do tej pory 4 meczów.
Klub Lechia Gdańsk zagral do tej pory 4 meczów.
Klub Legia Warszawa zagral do tej pory 4 meczów.
Klub Miedź Legnica zagral do tej pory 3 meczów.
Klub Piast Gliwice zagral do tej pory 4 meczów.
Klub Pogoń Szczecin zagral do tej pory 4 meczów.
Klub Śląsk Wrocław zagral do tej pory 2 meczów.
Klub Wisła Kraków zagral do tej pory 4 meczów.
Klub Wisła Płock zagral do tej pory 4 meczów.
Klub Zagłębie Sosnowiec zagral do tej pory 4 meczów.
```

- Dodatkowe zapytania sprawdzające poprawność wpisywanych danych i relacji:

  - Lista wszystkich meczów wraz z klubami i wynikami (pierwsze 15 rekordów):

```
SELECT m.id_meczu, k.id_klubu as gospodarz, m.punkty_gospodarza,
    (SELECT mk.id_klubu
     FROM mecze_kluby mk
     WHERE mk.id_klubu != k.id_klubu AND mk.id_meczu = m.id_meczu) as gosc,
     m.punkty_gościa
FROM mecze m
JOIN kluby k
ON m.id stadionu = k.id stadionu;
```

ID_MECZU	GOSPODARZ	PUNKTY_GOSPODARZA	GOSC	PUNKTY_GOŚCIA
2000	908	0	912	1
2001	913	1	906	0
2002	915	3	910	1
2003	903	0	901	0
2004	911	1	905	3
2005	914	1	902	1
2006	909	1	907	2
2007	904	1	916	2
2008	906	1	915	1
2009	903	2	912	1
2010	905	2	904	1
2011	902	1	911	2
2012	914	1	909	1
2013	901	0	913	2
2014	907	2	910	0

  - Ile meczów sędziował do tej pory każdy z sędziów?

```
SELECT s.nr_licencji, count(m.id_meczu) as ilosc_meczow
FROM sędziowie s
JOIN mecze m
ON m.nr_licencji_sędziego = s.nr_licencji
GROUP BY s.nr_licencji;
```

NR_LICENCJI	ILOSC_MECZOW
43ae	1
23fg	1
zaw3	2
t2ew	1
aj47	1
4er4	1
wey5	1
yt5r	8
76tr	3
29df	1
iul2	2
st54	1
ai37	2
jd63	2
54e4	1
5r54	1
erwl	1

- Zestawienie wszystkich sędziów i wszystkich klubów wraz z ilością ich spotkań podczas meczów (15 pierwszych rekordów)

```
SELECT s.nr_licencji, mk.id_klubu, count(mk.id_klubu)
FROM sędziowie s
JOIN mecze m
ON m.nr_licencji_sędziego = s.nr_licencji
JOIN mecze_kluby mk
ON m.id_meczu = mk.id_meczu
GROUP BY s.nr_licencji, mk.id_klubu
ORDER BY mk.id_klubu;
```

	NR_LICENCJI	ID_KLUBU	COUNT(MK.ID_KLUBU)
1	43ae	901	1
2	54e4	901	1
3	yt5r	901	1
4	zaw3	901	1
5	76tr	902	1
6	wey5	902	1
7	yt5r	902	1
8	43ae	903	1
9	jd63	903	1
10	t2ew	903	1
11	yt5r	903	1
12	29df	904	1
13	ai37	904	1
14	erwl	904	1
15	yt5r	904	1

ANALIZA ROZWIĄZANIA

Nie wprowadziłam istotnych zmian względem pierwotnej koncepcji. Wszystko zostało tak, jak było to zaplanowane w etapie I projektu. Istnieje jednak kilka ograniczeń dotyczących całej bazy, m.in.:

- Do meczu może być dołączony jedynie sędzia główny (dwaj liniowi nie są uwzględnieni).
- Nie ma limitu dolnego i górnego, który określa liczbę zawodników w danym klubie.
- Do tabeli „Kluby" można dodać większą ilość klubów, nawet jeśli nie brały one udziału w Ekstraklasie.

W późniejszym czasie można pokusić się na zamianę wszystkich fikcyjnych danych na autentyczne i w ten sposób uzyskać rzeczywistą, rzetelną bazę danych dotyczącą rozgrywek polskiej Ekstraklasy w sezonie 2018/2019. Wtedy użytkownik mógłby spokojnie poruszać się po bazie, bazować na prawidłowych danych i wysuwać odpowiednie wnioski z analizy wszystkich tabel. Dodatkowo można dodać pozostałe odbyte już mecze, aby wyłonić prawdziwą zwycięską drużynę i realnego króla strzelców na koniec sezonu.

Baza danych została zaprojektowana w taki sposób, aby umożliwić fanom piłki nożnej organizację i prowadzenie własnego „dziennika" Ekstraklasy w danym sezonie. Oczywiście poprzez transfery, dodawanie, usuwanie zawodników/klubów łatwo można przenieść bazę na kolejny sezon, a historię poprzedniego – oznaczyć w tabelach i dopisać do konkretnych dat, jeśli zajdzie taka potrzeba. Wystarczy stworzyć w tabeli Mecze dodatkowe pole, przykładowo o nazwie „Sezon". Ja zdecydowałam się na organizację sezonu 2018/2019 ze względu na to, że chciałam wprowadzić do testów autentyczne dane.