POLITECHNIKA CZĘSTOCHOWSKA
WYDZIAŁ INŻYNIERII MECHANICZNEJ I INFORMATYKI

BAZY DANYCH

PROJEKT BAZY DANYCH: ROZGRYWKI PIŁKARSKIE

AUTOR:

DARIUSZ WUJEC

2020

1. Wprowadzenie

Projekt bazy danych dla rozgrywek piłkarskich zawiera 8 tabel:

- Gole
- Kartki
- Kluby
- Ligi
- Sezony
- Sklady
- Spotkania

Przykładowe dane zawarte w tabeli Ligi dotyczą istniejących europejskich lig piłkarskich i pochodzą z Wikipedii. Miejscowości przykładowych klubów są “inspirowane” Ekstraklasą, natomiast reszta danych jest przypadkowa. Imiona i nazwiska piłkarzy w tabeli Zawodnicy są wygenerowane losowo przy użyciu 150 najpopularniejszych imion oraz 15 000 najpopularniejszych nazwisk w naszym kraju (dane dostępne na stronie dane.gov.pl). Przykładowe dane w reszcie tabel są przypadkowe.

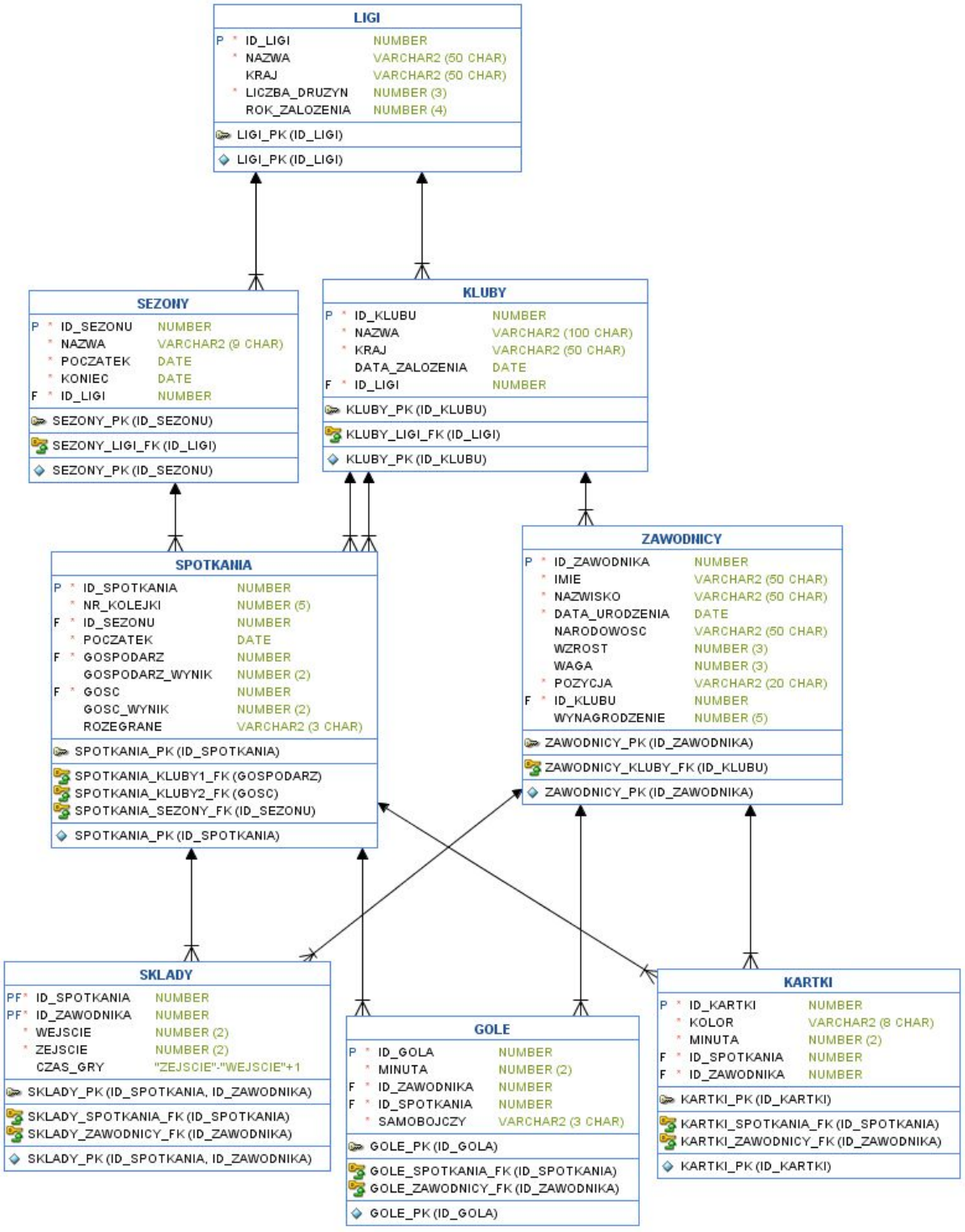
Projekt zawiera także 4 perspektywy:

- ranking goli strzelonych w jednym meczu
- ranking strzelców
- tabela wyników w sezonie
- terminarz meczy

W projekcie przyjęto kilka uproszczeń :

- mecz składa się z dwóch połów po 45 minut - nie ma dogrywek
- każdy klub gra tylko w jednej lidze (nie spada z niej ani nie awansuje)
- zawodnicy nie zmieniają klubów

2. Diagram związków encji



3. Tabele

Gole - tabela przechowuje informacje o strzelonych golach

Kolumny:

- ***id_gola*** - klucz główny tabeli
- ***minuta*** - minuta, w której został strzelony gol, przyjmuje wartości od 1 do 90, pole obowiązkowe
- ***id_zawodnika*** - klucz obcy odwołujący się do tabeli Zawodnicy, jest to identyfikator zawodnika, który strzelił gola
- ***id_spotkania*** - klucz obcy odwołujący się do tabeli Spotkania, jest to identyfikator spotkania, podczas którego strzelono gola
- ***samobojczy*** - pole tekstowe przyjmujące wartości ‘TAK’ i ‘NIE’ (domyślnie ‘NIE’), wartość tak oznacza, że zawodnik trafił do własnej bramki, pole obowiązkowe

Kartki - tabela przechowuje informacje o karach otrzymanych przez zawodników

Kolumny:

- ***id_kartki*** - klucz główny tabeli
- ***kolor*** - pole tekstowe przyjmujące wartości ‘ŻÓŁTA’ i ‘CZERWONA’, pole obowiązkowe
- ***minuta*** - minuta, w której zawodnik otrzymał karę, przyjmuje wartości od 1 do 90, pole obowiązkowe
- ***id_spotkania*** - klucz obcy odwołujący się do tabeli Spotkania, jest to identyfikator spotkania, podczas którego zawodnik otrzymał karę
- ***id_zawodnika*** - klucz obcy odwołujący się do tabeli Zawodnicy, jest to identyfikator zawodnika, który otrzymał karę

Kluby - tabela przechowuje informacje dotyczące klubów piłkarskich

Kolumny:

- ***id_klubu*** - klucz główny tabeli
- ***nazwa*** - pole tekstowe przechowujące nazwę klubu piłkarskiego, może zawierać maksymalnie 100 znaków, pole jest obowiązkowe
- ***kraj*** - pole tekstowe przechowujące nazwę kraju, w którym funkcjonuje klub piłkarski, maksymalnie 50 znaków, pole jest obowiązkowe
- ***data_zalozenia*** - data założenia klubu piłkarskiego, przyjmuje wartości pomiędzy 24.10.1857 (według Wikipedii wtedy został założony najstarszy klub piłkarski), a 01.01.2020
- ***id_ligi*** - klucz obcy odwołujący się do tabeli Ligi, jest to identyfikator ligi, w której gra klub

Ligi - tabela przechowuje informacje o ligach piłkarskich

Kolumny:

- ***id_ligi*** - klucz główny tabeli
- ***nazwa*** - pole tekstowe przechowujące nazwę ligi piłkarskiej, maksymalnie 50 znaków, pole obowiązkowe
- ***kraj*** - pole tekstowe przechowujące nazwę kraju, w którym rozgrywana jest liga piłkarska
- ***liczba_druzyn*** - maksymalna (większa od 1) liczba drużyn mogących grać w lidze, pole obowiązkowe
- ***rok_zalozenia*** - rok założenia ligi piłkarskiej

Sezony - tabela przechowuje informacje o sezonach rozgrywanych w konkretnych ligach piłkarskich

Kolumny:

- ***id_sezonu*** - klucz główny tabeli
- ***nazwa*** - krótka (9 znaków) nazwa sezonu, pole obowiązkowe
- ***poczatek*** - data rozpoczęcia sezonu, pole obowiązkowe
- ***koniec*** - data zakończenia sezonu (data późniejsza niż początek sezonu), pole obowiązkowe
- ***id_ligi*** - klucz obcy odwołujący się do tabeli Ligi, jest to identyfikator ligi w ramach której rozgrywany jest sezon

Sklady - tabela przechowuje informacje o składach drużyn piłkarskich podczas konkretnych meczów

Kolumny:

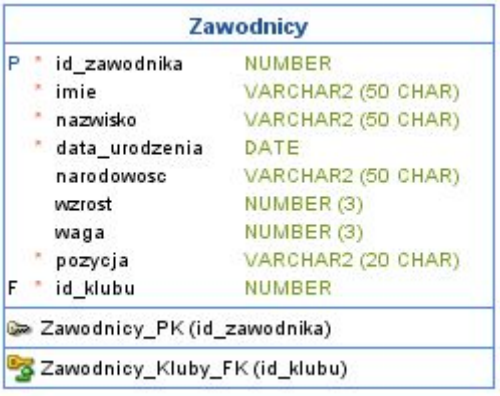
- ***id_spotkania*** - klucz obcy odwołujący się do tabeli Spotkania, razem z kolumną id_zawodnika tworzy klucz główny tabeli
- ***id_zawodnika*** - klucz obcy odwołujący się do tabeli Zawodnicy, razem z kolumną id_spotkania tworzy klucz główny tabeli
- ***wejscie*** - minuta, w której zawodnik wszedł na boisko (od 1 do 90), pole obowiązkowe
- ***zejscie*** - minuta, w której zawodnik zszedł z boiska (do wejścia do 90), pole obowiązkowe
- ***czas_gry*** - kolumna wirtualna przechowująca liczbę minut spędzonych przez zawodnika na boisku w konkretnym spotkaniu

Spotkania - tabela przechowuje informacje o rozegranych lub planowanych meczach

Kolumny:

- ***id_spotkania*** - klucz główny tabeli
- ***nr_kolejki*** - numer kolejki w ramach której rozgrywane jest spotkanie, pole obowiązkowe
- ***id_sezonu*** - klucz obcy odwołujący się do tabeli Sezony, jest to identyfikator sezonu w ramach którego rozgrywane jest spotkanie
- ***poczatek*** - data rozegrania spotkania, pole obowiązkowe
- ***gospodarz*** - klucz obcy odwołujący się do tabeli Kluby, jest to identyfikator klubu, który jest gospodarzem spotkania
- ***gospodarz_wynik*** - kolumna przechowująca liczbę (nieujemną) goli gospodarza, domyślnie 0
- ***gosc*** - klucz obcy odwołujący się do tabeli Kluby, jest to identyfikator klubu, który gra na wyjeździe
- ***gosc_wynik*** - kolumna przechowująca liczbę (nieujemną) goli gościa, domyślnie 0
- ***rozegrane*** - pole tekstowe przyjmujące wartości ‘TAK’ i ‘NIE’ (domyślnie ‘NIE’), informuje o tym czy spotkanie zostało rozegrane

Zawodnicy - tabela przechowuje informacje o piłkarzach



Kolumny:

- ***id_zawodnika*** - klucz główny tabeli
- ***imie*** - pole tekstowe przechowujące imię piłkarza (maksymalnie 50 znaków), obowiązkowe
- ***nazwisko*** - pole tekstowe przechowujące nazwisko piłkarza (maksymalnie 50 znaków), obowiązkowe
- ***data_urodzenia*** - data urodzenia piłkarza, dozwolone są daty starsze niż 01.01.2005, pole obowiązkowe
- ***narodowosc*** - pole tekstowe przechowujące informację o narodowości piłkarza (maksymalnie 50 znaków)
- ***wzrost*** - wzrost piłkarza podany w centymetrach, dozwolone są wartości od 140 do 220
- ***waga*** - waga piłkarza podana w kilogramach, dozwolone są wartości od 50 do 130
- ***pozycja*** - pozycja na boisku, na której gra piłkarz, przyjmuje wartości ‘BRAMKARZ’, ‘OBROŃCA’, ‘POMOCNIK’ i ‘NAPASTNIK’, pole obowiązkowe
- ***id_klubu*** - klucz obcy odwołujący się do tabeli Kluby, jest to identyfikator klubu w którym gra zawodnik

4. Sekwencje

SEQ_ID_ZAW - sekwencja używana podczas wprowadzania danych do tabeli Zawodnicy

```
CREATE SEQUENCE seq_id_zaw
MINVALUE 1
MAXVALUE 10000
START WITH 1
NOCYCLE
INCREMENT BY 1;
```

Wartość początkowa i minimalna to 1
Wartość maksymalna to 10 000
Brak cykliczności
Zwiększanie wartości o 1

Przykład wprowadzania danych z użyciem sekwencji seq_id_zaw:

```
INSERT INTO Zawodnicy
VALUES
  (
    seq_id_zaw.nextval,
    'ERYK',
    'SZCZEPARA',
    TO_DATE('2004-03-01', 'YYYY-MM-DD'),
    'POLSKA',
    164,
    63,
    'BRAMKARZ',
    1
  );
```

seq_id_zaw.nextval oznacza pobranie następnej wartości z sekwencji, wszystkie dane podane w kolejności odpowiadającej kolejności kolumn w tabeli (id_zawodnika, imie, nazwisko, data_urodzenia, narodowosc, wzrost, waga, pozycja i id_klubu)

Sekwencje wygenerowane za pomocą programu Oracle SQL Developer Data Modeler:

SEQ_ID_GOLA - wykorzystywana przez wyzwalacz gole_id_gola_trg podczas wprowadzania danych do tabeli Gole:

```
CREATE SEQUENCE seq_id_gola START WITH 1 MINVALUE 1 MAXVALUE 9999 NOCACHE ORDER;
```

SEQ_ID_KARTKI - wykorzystywana przez wyzwalacz kartki_id_kartki_trg podczas wprowadzania danych do tabeli Kartki:

```
CREATE SEQUENCE seq_id_kartki START WITH 1 MINVALUE 1 MAXVALUE 9999 NOCACHE ORDER;
```

Więcej o wyzwalaczach zawartych w tym projekcie w następnym rozdziale.

5. Wyzwalacze

SPRAWDZ_LIMIT_DRUZYN_LIGI - sprawdza czy można dodać następny klub do ligi o podanym id_ligi - w przypadku gdy został już osiągnięty limit drużyn w lidze powoduje przerwanie procedury wprowadzania nowego rekordu do tabeli Kluby.

```
CREATE OR REPLACE TRIGGER Sprawdz_limit_druzyn_ligi BEFORE INSERT ON Kluby
FOR EACH ROW
DECLARE
    limit_druzyn INTEGER;
    aktualna_ilosc INTEGER;
BEGIN
    SELECT liczba_druzyn INTO limit_druzyn FROM Ligi WHERE id_ligi = :new.id_ligi;
    SELECT COUNT(*) INTO aktualna_ilosc FROM Kluby WHERE id_ligi = :new.id_ligi;

    IF aktualna_ilosc = limit_druzyn THEN
        raise_application_error(-20001, 'ANULOWANIE procedury wprowadzania
        nowego klubu. Nie mozna dodac wiecej druzyn do ligi '||:new.id_ligi||'!');
    END IF;
END;
/
```

SPRAWDZ_NUMER_KOLEJKI - sprawdza czy numer kolejki podany podczas wprowadzania nowego rekordu do tabeli Spotkania nie jest większy od maksymalnej liczby kolejek (2 * (liczba_druzyn - 1)) w danej lidze, jeśli jest większy to przerywa procedurę wprowadzania nowego rekordu do tabeli.

```
CREATE OR REPLACE TRIGGER Sprawdz_nr_kolejki BEFORE INSERT ON Spotkania
FOR EACH ROW
DECLARE
    liczba_kolejek INTEGER;
    liga INTEGER;
BEGIN
    SELECT id_ligi, 2*(liczba_druzyn-1) INTO liga, liczba_kolejek FROM Ligi
    WHERE id_ligi = (
        SELECT id_ligi FROM Sezony WHERE id_sezonu = :new.id_sezonu
    );

    IF :new.nr_kolejki > liczba_kolejek THEN
        raise_application_error(-20001, 'ANULOWANIE procedury wprowadzania
        nowego spotkania. Podano za duzy numer kolejki - '
        ||' liczba kolejek dla ligi '||liga||' wynosi '||liczba_kolejek);
    END IF;
END;
/
```

SPRAWDZ_POCZATEK_SPOTKANIA - sprawdza czy data początku spotkania podana podczas wprowadzania rekordu do tabeli Spotkania mieści się między datą początku i końca sezonu o podanym id_sezonu, jeśli nie to przerywa procedurę wprowadzania nowego rekordu do tabeli.

```
CREATE OR REPLACE TRIGGER Sprawdz_poczatek_spotkania BEFORE INSERT ON Spotkania
FOR EACH ROW
DECLARE
    poczatek_sezonu DATE;
    koniec_sezonu DATE;
BEGIN
    SELECT poczatek, koniec INTO poczatek_sezonu, koniec_sezonu
    FROM Sezony WHERE id_sezonu = :new.id_sezonu;

    IF :new.poczatek NOT BETWEEN poczatek_sezonu AND koniec_sezonu THEN
        raise_application_error(-20001, 'ANULOWANIE procedury wprowadzania nowego
        spotkania. Podana data spotkania nie miesci sie w ramach czasowych sezonu.');
    END IF;
END;
/
```

SPRAWDZ_GOSPODARZA_SPOTKANIA - sprawdza czy klub, który jest gospodarzem spotkania należy do ligi w ramach której rozgrywane jest spotkanie, jeśli nie to przerywa procedurę wprowadzania nowego rekordu do tabeli Spotkania.

```
CREATE OR REPLACE TRIGGER Sprawdz_gospodarza_spotkania BEFORE INSERT ON Spotkania
FOR EACH ROW
DECLARE
    liga_gospodarza INTEGER;
    liga_spotkania INTEGER;
BEGIN
    SELECT id_ligi INTO liga_gospodarza FROM Kluby WHERE id_klubu = :new.gospodarz;
    SELECT id_ligi INTO liga_spotkania FROM Sezony WHERE id_sezonu = :new.id_sezonu;

    IF liga_gospodarza != liga_spotkania THEN
        raise_application_error(-20001, 'Klub o id '||:new.gospodarz||
        ' nie nalezy do ligi o id '||liga_spotkania||'!');
    END IF;
END;
/
```

SPRAWDZ_GOSCIA_SPOTKANIA - sprawdza czy klub, który gra na wyjeździe należy do ligi w ramach której rozgrywane jest spotkanie, jeśli nie to przerywa procedurę wprowadzania nowego rekordu do tabeli Spotkania.

```
CREATE OR REPLACE TRIGGER Sprawdz_goscia_spotkania BEFORE INSERT ON Spotkania
FOR EACH ROW
DECLARE
    liga_goscia INTEGER;
    liga_spotkania INTEGER;
BEGIN
    SELECT id_ligi INTO liga_goscia FROM Kluby WHERE id_klubu = :new.gosc;
    SELECT id_ligi INTO liga_spotkania FROM Sezony WHERE id_sezonu = :new.id_sezonu;

    IF liga_goscia != liga_spotkania THEN
        raise_application_error(-20001, 'Klub o id '||:new.gosc||
        ' nie nalezy do ligi o id '||liga_spotkania||'!');
    END IF;
END;
/
```

SPRAWDZ_CZY_DRUGA_ZOLTA - sprawdza czy zawodnik otrzymał drugą żółtą kartkę w trakcie spotkania, jeśli tak to zmienia wartość kolumny kolor na ‘CZERWONA’. Wyzwalacz jest wykonywany tylko wtedy, gdy kolumna kolor nowego rekordu wstawianego do tabeli Kartki ma wartość ‘ŻÓŁTA’.

```
CREATE OR REPLACE TRIGGER Sprawdz_czy_druga_zolta BEFORE INSERT ON Kartki
FOR EACH ROW
WHEN (NEW.kolor = 'ŻÓŁTA')
DECLARE
    liczba_kartek_w_meczu INTEGER;
BEGIN
    SELECT COUNT(*) INTO liczba_kartek_w_meczu FROM Kartki
    WHERE id_spotkania = :new.id_spotkania AND id_zawodnika = :new.id_zawodnika;

    IF liczba_kartek_w_meczu = 1 THEN
        :new.kolor := 'CZERWONA';
    END IF;
END;
/
```

USTAW_MINUTE_ZEJSCIA - sprawdza, czy zawodnik otrzymał czerwoną kartkę, jesli tak to ustawia wartość kolumny zejście w tabeli Sklady na wartość kolumny minuta rekordu dodawanego do tabeli Kartki. Wyzwalacz jest wykonywany dopiero po wykonaniu wyzwalacza sprawdz_czy_druga_zolta.

```
CREATE OR REPLACE TRIGGER Ustaw_minute_zejscia BEFORE INSERT ON Kartki
FOR EACH ROW
FOLLOWS Sprawdz_czy_druga_zolta
WHEN (NEW.kolor = 'CZERWONA')
BEGIN
    UPDATE Sklady SET zejscie = :new.minuta
    WHERE id_spotkania = :new.id_spotkania AND id_zawodnika = :new.id_zawodnika;
END;
/
```

SPRAWDZ_CZY_MA_CZERWONA - sprawdza czy zawodnik o podanym id_zawodnika w ciągu aktualnego spotkania lub 2 poprzednich otrzymał czerwoną kartkę, jeśli tak to przerywa procedurę wprowadzania nowego rekordu do tabeli Sklady.

```
CREATE OR REPLACE TRIGGER Sprawdz_czy_ma_czerwona BEFORE INSERT ON Sklady
FOR EACH ROW
DECLARE
    liczba_czerwonych INTEGER;
    sezon INTEGER;
    kolejka INTEGER;
BEGIN
    SELECT id_sezonu, nr_kolejki INTO sezon, kolejka
    FROM Spotkania
    WHERE id_spotkania = :new.id_spotkania;

    SELECT COUNT(*) INTO liczba_czerwonych
    FROM kartki
    NATURAL JOIN zawodnicy
    NATURAL JOIN spotkania
    WHERE id_zawodnika = :new.id_zawodnika AND id_sezonu = sezon
    AND nr_kolejki IN (kolejka, kolejka-1, kolejka-2) AND kolor = 'CZERWONA';

    IF liczba_czerwonych > 0 THEN
        raise_application_error(-20001, 'Zawodnik o id '||:new.id_zawodnika
        ||' w ciagu 2 ostatnich spotkan otrzymal czerwona kartke i nie moze
        uczestniczyc w tym spotkaniu');
    END IF;
END;
/
```

ZAKTUALIZUJ_WYNIK - wyzwalacz po wstawieniu rekordu do tabeli Gole aktualizuje liczbę goli odpowiedniej drużyny w tabeli Spotkania.

```
CREATE OR REPLACE TRIGGER Zaktualizuj_wynik AFTER INSERT ON Gole
FOR EACH ROW
DECLARE
    klub_strzelajacego INTEGER;
    gospodarz INTEGER;
    gosc INTEGER;
BEGIN
    SELECT id_klubu INTO klub_strzelajacego
    FROM Zawodnicy WHERE id_zawodnika = :new.id_zawodnika;

    SELECT gospodarz, gosc INTO gospodarz, gosc
    FROM Spotkania WHERE id_spotkania = :new.id_spotkania;

    IF :new.samobojczy = 'NIE' AND klub_strzelajacego = gospodarz
    OR :new.samobojczy = 'TAK' AND klub_strzelajacego = gosc THEN
        UPDATE Spotkania SET gospodarz_wynik = gospodarz_wynik + 1
        WHERE id_spotkania = :new.id_spotkania;
    END IF;

    IF :new.samobojczy = 'NIE' AND klub_strzelajacego = gosc
    OR :new.samobojczy = 'TAK' AND klub_strzelajacego = gospodarz THEN
        UPDATE Spotkania SET gosc_wynik = gosc_wynik + 1
        WHERE id_spotkania = :new.id_spotkania;
    END IF;

END;
/
```

FORMATUJ_DANE_ZAWODNIKA - zmienia sposób zapisu danych wprowadzanych do kolumn imie, nazwisko i narodowosc tabeli Zawodnicy - imię i nazwisko mają pierwszą literę dużą i resztę liter małych (initcap), narodowość jest zapisana dużymi literami (upper).

```
CREATE OR REPLACE TRIGGER Formatuj_dane_zawodnika BEFORE INSERT ON Zawodnicy
FOR EACH ROW
BEGIN
    :new.imie := initcap(:new.imie);
    :new.nazwisko := initcap(:new.nazwisko);

    IF :new.narodowosc IS NOT NULL THEN
        :new.narodowosc := upper(:new.narodowosc);
    END IF;
END;
/
```

SPRAWDZ_CZY_DWIE_ROZNE_DRUZYNY - sprawdza czy gospodarz i gość spotkania to nie ta sama drużyna - jeśli tak to przerywa procedurę wprowadzania nowego rekordu do tabeli Spotkania.

```
CREATE OR REPLACE TRIGGER sprawdz_czy_dwie_rozne_druzyny BEFORE INSERT ON spotkania
FOR EACH ROW
BEGIN
    IF :NEW.gospodarz = :NEW.gosc THEN
        raise_application_error(-20001, 'ANULOWANIE procedury wprowadzania nowego spotkania -
        gospodarz i gosc spotkania nie moze byc ta sama druzyna.');
    END IF;
END;
/
```

SPRAWDZ_LICZBE_MECZY_W_KOLEJCE - sprawdza czy drużyny podane jako gospodarz i gość spotkania nie rozgrywały już jakiegoś spotkania w danej kolejce danego sezonu - jeśli tak to przerywa procedurę wprowadzania nowego rekordu do tabeli Spotkania.

```
CREATE OR REPLACE TRIGGER sprawdz_liczbe_meczy_w_kolejce BEFORE INSERT ON spotkania
FOR EACH ROW
DECLARE
    rozegrane_spotkania_gospodarz INTEGER;
    rozegrane_spotkania_gosc INTEGER;
BEGIN
    SELECT COUNT(*) INTO rozegrane_spotkania_gospodarz
    FROM spotkania
    WHERE (gospodarz = :NEW.gospodarz OR gosc = :NEW.gospodarz)
    AND id_sezonu = :NEW.id_sezonu AND nr_kolejki = :NEW.nr_kolejki;

    SELECT COUNT(*) INTO rozegrane_spotkania_gosc
    FROM spotkania
    WHERE (gospodarz = :NEW.gosc OR gosc = :NEW.gosc)
    AND id_sezonu = :NEW.id_sezonu AND nr_kolejki = :NEW.nr_kolejki;

    IF rozegrane_spotkania_gospodarz > 0 THEN
        raise_application_error(-20001, 'ANULOWANIE procedury wprowadzania nowego spotkania -
        gospodarz spotkania rozegral juz jeden mecz w kolejce numer '||:NEW.nr_kolejki||' sezonu
         o id '||:NEW.id_sezonu);
    END IF;

    IF rozegrane_spotkania_gosc > 0 THEN
        raise_application_error(-20001, 'ANULOWANIE procedury wprowadzania nowego spotkania -
        gosc spotkania rozegral juz jeden mecz w kolejce numer '||:NEW.nr_kolejki||' sezonu
         o id '||:NEW.id_sezonu);
    END IF;
END;
/
```

Wyzwalacze wygenerowane za pomocą programu Oracle SQL Developer Data Modeler:

GOLE_ID_GOLA_TRG - jeśli nie podano id_gola podczas wstawiania nowego rekordu do tabeli Gole to ustawia wartość tej kolumny na następną wolną wartość z sekwencji seq_id_gola.

```
CREATE OR REPLACE TRIGGER gole_id_gola_trg BEFORE
    INSERT ON gole
    FOR EACH ROW
    WHEN ( new.id_gola IS NULL )
BEGIN
    :new.id_gola := seq_id_gola.nextval;
END;
/
```

KARTKI_ID_KARTKI_TRG - jeśli nie podano id_kartki podczas wstawiania nowego rekordu do tabeli Kartki to ustawia wartość tej kolumny na następną wolną wartość z sekwencji seq_id_kartki.

```
CREATE OR REPLACE TRIGGER kartki_id_kartki_trg BEFORE
    INSERT ON kartki
    FOR EACH ROW
    WHEN ( new.id_kartki IS NULL )
BEGIN
    :new.id_kartki := seq_id_kartki.nextval;
END;
/
```

6. Aktualizacje

Dane testowe wprowadzone do tabeli Gole zawierają informacje dotyczące goli strzelonych w kolejce numer 1 sezonu o identyfikatorze 10. Instrukcja aktualizująca zawartość kolumny rozegrane (ustawienie wartości ‘TAK’) w tabeli Spotkania:

```
UPDATE Spotkania SET rozegrane = 'TAK' WHERE id_sezonu = 10 AND nr_kolejki = 1;
```

Instrukcje dodające do tabeli Zawodnicy kolumnę wynagrodzenie (maksymalnie 5 cyfrowa liczba) i ustawiające jej wartość według zasad:

- jeśli zawodnik nie spędził na boisku ani jednej minuty
  3000
- w przeciwnym przypadku
  3000 + 10 * liczba_minut_na_boisku +
  200 * liczba_wygranych_jako_gospodarz +
  400 * liczba_wygranych_jako_gosc +
  500 * liczba_goli

```
ALTER TABLE zawodnicy ADD wynagrodzenie NUMBER(5);

UPDATE zawodnicy z SET wynagrodzenie =
CASE
    WHEN NOT EXISTS(SELECT * FROM sklady WHERE id_zawodnika = z.id_zawodnika) THEN 3000
    ELSE 3000 +
    10 * (
        SELECT SUM(czas_gry)
        FROM sklady
        WHERE id_zawodnika = z.id_zawodnika
        GROUP BY id_zawodnika
    ) +
    200 * (
        SELECT COUNT(*)
        FROM spotkania
        JOIN sklady USING(id_spotkania)
        WHERE gospodarz = z.id_klubu AND id_zawodnika = z.id_zawodnika
        AND gospodarz_wynik > gosc_wynik
    ) +
    400 * (
        SELECT COUNT(*)
        FROM spotkania
        JOIN sklady USING(id_spotkania)
        WHERE gosc = z.id_klubu AND id_zawodnika = z.id_zawodnika
        AND gospodarz_wynik < gosc_wynik
    ) +
    500 * (SELECT COUNT(*) FROM gole WHERE id_zawodnika = z.id_zawodnika)
END;
```

7. Zapytania

Zapytanie wyświetlające informacje o meczach rozegranych dnia 01.11.2019 (drużyny i wyniki spotkań):

```
SELECT (SELECT nazwa FROM kluby WHERE id_klubu = gospodarz) gospodarz,
(gospodarz_wynik||'-'||gosc_wynik) wynik,
(SELECT nazwa FROM kluby WHERE id_klubu = gosc) gosc,
poczatek data_spotkania
FROM spotkania
WHERE rozegrane = 'TAK' AND poczatek = TO_DATE('01.11.2019', 'DD.MM.YYYY');
```

Przykładowy wynik zapytania:

	GOSPODARZ	WYNIK	GOSC	DATA_SPOTKANIA
1	Odra Wrocław	0-2	Czerwoni Łódź	19/11/01
2	Bałtyk Gdańsk	0-0	Żółci Lubin	19/11/01

Zapytanie wyświetlające informacje o liczbie goli, goli samobójczych, żółtych i czerwonych kartek poszczególnych zawodników. Dane posortowane malejąco według liczby czerwonych kartek, oraz malejąco według liczby żółtych kartek:

```
SELECT imie, nazwisko, nazwa klub, nvl(SUM(gole), 0) gole,
nvl(SUM(gole_samobojcze), 0) gole_samobojcze,
nvl(SUM(zolta), 0) zolte_kartki,
nvl(SUM(czerwona), 0) czerwone_kartki
FROM zawodnicy
JOIN kluby USING(id_klubu)
LEFT JOIN (
    SELECT id_zawodnika,
    CASE WHEN samobojczy = 'TAK' THEN 1 ELSE 0 END gole_samobojcze,
    CASE WHEN samobojczy = 'NIE' THEN 1 ELSE 0 END gole
    FROM gole
) USING(id_zawodnika)
LEFT JOIN (
    SELECT id_zawodnika,
    CASE WHEN kolor = 'ŻÓŁTA' THEN 1 ELSE 0 END zolta,
    CASE WHEN kolor = 'CZERWONA' THEN 1 ELSE 0 END czerwona
    FROM kartki
) USING(id_zawodnika)
GROUP BY id_zawodnika, imie, nazwisko, nazwa, gole, zolta, czerwona, gole_samobojcze
ORDER BY 7 DESC, 6 DESC;
```

Przykładowy wynik zapytania (fragment):

	IMIE	NAZWISKO	KLUB	GOLE	GOLE_SAMOBOJCZE	ZOLTE_KARTKI	CZERWONE_KARTKI
1	Kacper	Myszka	Warta Częstochowa	0	0	0	1
2	Gabriel	Kotaś	Żółci Lubin	0	0	0	1
3	Sebastian	Korkus	Bałtyk Gdańsk	0	0	0	1
4	Oleksandr	Rezler	Bałtyk Gdańsk	0	0	1	0
5	Kajetan	Likus	Czerwoni Łódź	0	0	1	0
6	Rafał	Bis	Czarni Kraków	0	0	1	0
7	Bruno	Staszak	Śląsk Gliwice	0	0	1	0
8	Filip	Leśnikowski	Wisła Warszawa	0	0	1	0
9	Przemysław	Rechnio	Podlasie Białystok	1	0	1	0

Zapytanie zwracające informacje o średnim wieku, wzroście i wadze piłkarzy w poszczególnych klubach, ligach i ogólnie:

```
SELECT nvl(ligi.nazwa, ' ') liga, nvl(kluby.nazwa, ' ') klub,
sredni_wiek, sredni_wzrost, srednia_waga, komentarz
FROM kluby
RIGHT JOIN (
    SELECT id_ligi, id_klubu,
    round(AVG(months_between(sysdate, data_urodzenia)/12),1) sredni_wiek,
    round(AVG(wzrost)) sredni_wzrost,
    round(AVG(waga)) srednia_waga,
    CASE
        WHEN id_ligi IS NOT NULL AND id_klubu IS NOT NULL THEN 'Statystyki dla klubu'
        WHEN id_ligi IS NOT NULL AND id_klubu IS NULL THEN 'Statystyki dla ligi'
        WHEN id_ligi IS NULL AND id_klubu IS NULL THEN 'Statystyki ogólne'
    END komentarz
    FROM zawodnicy
    JOIN kluby USING(id_klubu)
    GROUP BY ROLLUP(id_ligi, id_klubu)
) USING(id_klubu, id_ligi)
LEFT JOIN ligi USING(id_ligi);
```

Przykładowy wynik zapytania (fragment):

	LIGA	KLUB	SREDNI_WIEK	SREDNI_WZROST	SREDNIA_WAGA	KOMENTARZ
11	Ekstraklasa	Niebiescy Kielce	30,6	177	71	Statystyki dla klubu
12	Ekstraklasa	Zieloni Gdynia	32,4	182	75	Statystyki dla klubu
13	Ekstraklasa	Czarni Kraków	28,8	181	73	Statystyki dla klubu
14	Ekstraklasa	Pomarańczowi Płock	33,8	184	75	Statystyki dla klubu
15	Ekstraklasa	Śląsk Gliwice	30,4	177	71	Statystyki dla klubu
16	Ekstraklasa	Warta Częstochowa	27,8	181	75	Statystyki dla klubu
17			30,7	180	73	Statystyki ogólne
18	Ekstraklasa		30,7	180	73	Statystyki dla ligi

Zapytanie zwracające informacje o zawodnikach, którzy zagrali więcej niż 1 ale mniej niż 45 minut na boisku i w tym czasie strzelili gola przeciwnej drużynie:

```
SELECT imie, nazwisko, nazwa klub, pozycja, SUM(czas_gry) czas_gry
FROM sklady
JOIN zawodnicy USING(id_zawodnika)
JOIN kluby USING(id_klubu)
JOIN gole USING(id_zawodnika)
WHERE samobojczy = 'NIE'
GROUP BY id_zawodnika, imie, nazwisko, nazwa, pozycja
HAVING SUM(czas_gry) BETWEEN 1 AND 45;
```

Przykładowy wynik zapytania:

	IMIE	NAZWISKO	KLUB	POZYCJA	CZAS_GRY
1	Przemysław	Rechnio	Podlasie Białystok	POMOCNIK	36
2	Karol	Chabasiński	Śląsk Gliwice	NAPASTNIK	35

Zapytanie zwracające liczbę goli poszczególnych drużyn w poszczególnych spotkaniach (rozdzielenie kolumn gospodarz i gość z tabeli Spotkania), oraz kolumnę rezultat zawierającą wynik spotkania:

```
SELECT id_spotkania, nr_kolejki, id_sezonu, gospodarz id_klubu, nazwa klub,
gospodarz_wynik gole,
CASE
WHEN gospodarz_wynik > gosc_wynik THEN 'WYGRANA'
WHEN gospodarz_wynik < gosc_wynik THEN 'PRZEGRANA'
WHEN gospodarz_wynik = gosc_wynik THEN 'REMIS'
END || '('||gospodarz_wynik||':'||gosc_wynik||')' rezultat
FROM spotkania
JOIN kluby ON (id_klubu = gospodarz)
WHERE rozegrane = 'TAK'
UNION
SELECT id_spotkania, nr_kolejki, id_sezonu, gosc, nazwa, gosc_wynik,
CASE
WHEN gosc_wynik > gospodarz_wynik THEN 'WYGRANA'
WHEN gosc_wynik < gospodarz_wynik THEN 'PRZEGRANA'
WHEN gospodarz_wynik = gosc_wynik THEN 'REMIS'
END || '('||gosc_wynik||':'||gospodarz_wynik||')'
FROM spotkania
JOIN kluby ON (id_klubu = gosc)
WHERE rozegrane = 'TAK';
```

Przykładowy wynik zapytania (fragment):

	ID_SPOTKANIA	NR_KOLEJKI	ID_SEZONU	ID_KLUBU	KLUB	GOLE	REZULTAT
1	1	1	10	1	Odra Wrocław	0	PRZEGRANA(0:2)
2	1	1	10	9	Czerwoni Łódź	2	WYGRANA(2:0)
3	2	1	10	2	Bałtyk Gdańsk	0	REMIS(0:0)
4	2	1	10	10	Żółci Lubin	0	REMIS(0:0)
5	3	1	10	3	Wielkopolska Poznań	1	PRZEGRANA(1:3)
6	3	1	10	11	Niebiescy Kielce	3	WYGRANA(3:1)

Zapytanie zwracające najwyższych zawodników w poszczególnych drużynach, dane posortowane malejąco według wzrostu:

```
SELECT imie, nazwisko, wzrost, pozycja, nazwa klub
FROM zawodnicy
JOIN kluby ON (zawodnicy.id_klubu = kluby.id_klubu)
WHERE NOT EXISTS(
    SELECT *
    FROM zawodnicy z
    WHERE z.wzrost > zawodnicy.wzrost
    AND z.id_klubu = zawodnicy.id_klubu
)
ORDER BY wzrost DESC;
```

Przykładowy wynik zapytania (fragment):

	IMIE	NAZWISKO	WZROST	POZYCJA	KLUB
1	Lech	Henke	209	POMOCNIK	Bałtyk Gdańsk
2	Krystian	Wajer	209	BRAMKARZ	Warta Częstochowa
3	Edward	Gola	209	OBROŃCA	Żółci Lubin
4	Wojciech	Kulus	209	POMOCNIK	Wisła Warszawa
5	Jan	Petrenko	209	POMOCNIK	Warta Częstochowa
6	Lesław	Kiełbasiński	208	POMOCNIK	Podlasie Białystok
7	Volodymyr	Blukacz	207	NAPASTNIK	Pomarańczowi Płock
8	Marcel	Zmitrowicz	207	POMOCNIK	Czerwoni Łódź
9	Norbert	Malarczyk	207	NAPASTNIK	Odra Wrocław
10	Włodzimierz	Jastrząbek	207	POMOCNIK	Śląsk Zabrze
11	Marcel	Franczuk	206	OBROŃCA	Małopolska Kraków
12	Jan	Freda	206	OBROŃCA	Małopolska Kraków
13	Gabriel	Murański	206	OBROŃCA	Czarni Kraków
14	Oliwier	Drop	205	OBROŃCA	Zieloni Gdynia

Zapytanie zwracające informacje o ustawieniu zawodników na boisku w poszczególnych meczach:

```
SELECT nazwa klub,
(SELECT nazwa FROM kluby WHERE id_klubu = gospodarz)||' - '||
(SELECT nazwa FROM kluby WHERE id_klubu = gosc) mecz, poczatek data_meczu,
SUM(o)||'-'||SUM(P)||'-'||SUM(N) ustawienie
FROM (
    SELECT id_klubu, id_spotkania,
    CASE WHEN pozycja = 'OBROŃCA' THEN 1 ELSE 0 END o,
    CASE WHEN pozycja = 'POMOCNIK' THEN 1 ELSE 0 END P,
    CASE WHEN pozycja = 'NAPASTNIK' THEN 1 ELSE 0 END N
    FROM sklady
    JOIN zawodnicy USING(id_zawodnika)
    WHERE wejscie = 1
)
JOIN spotkania USING(id_spotkania)
JOIN kluby USING(id_klubu)
GROUP BY id_klubu, nazwa, id_spotkania, gospodarz, gosc, poczatek;
```

Przykładowy wynik zapytania:

	KLUB	MECZ	DATA_MECZU	USTAWIENIE
1	Bałtyk Gdańsk	Bałtyk Gdańsk - Żółci Lubin	19/11/01	4-4-2
2	Niebiescy Kielce	Wielkopolska Poznań - Niebiescy Kielce	19/11/02	5-3-2
3	Czarni Kraków	Wisła Warszawa - Czarni Kraków	19/11/02	4-4-2
4	Wielkopolska Poznań	Wielkopolska Poznań - Niebiescy Kielce	19/11/02	4-4-2
5	Podlasie Białystok	Podlasie Białystok - Śląsk Gliwice	19/11/03	4-4-2
6	Czerwoni Łódź	Odra Wrocław - Czerwoni Łódź	19/11/01	4-4-2
7	Wisła Warszawa	Wisła Warszawa - Czarni Kraków	19/11/02	4-4-2
8	Żółci Lubin	Bałtyk Gdańsk - Żółci Lubin	19/11/01	4-4-2
9	Odra Wrocław	Odra Wrocław - Czerwoni Łódź	19/11/01	4-4-2
10	Małopolska Kraków	Małopolska Kraków - Zieloni Gdynia	19/11/02	4-4-2
11	Pomarańczowi Płock	Śląsk Zabrze - Pomarańczowi Płock	19/11/03	4-4-2
12	Śląsk Gliwice	Podlasie Białystok - Śląsk Gliwice	19/11/03	5-4-1
13	Zieloni Gdynia	Małopolska Kraków - Zieloni Gdynia	19/11/02	4-4-2
14	Śląsk Zabrze	Śląsk Zabrze - Pomarańczowi Płock	19/11/03	4-4-2
15	Warta Częstochowa	Odra Szczecin - Warta Częstochowa	19/11/03	4-3-3
16	Odra Szczecin	Odra Szczecin - Warta Częstochowa	19/11/03	3-5-2

8. Perspektywy

EKSTRAKL_RANK_STRZELCOW_19_20 - perspektywa prezentuje ranking strzelców w przykładowej lidze “Ekstraklasa” w sezonie “2019/2020”, zawiera kolumny ranking, pilkarz, klub i gole.

```
CREATE OR REPLACE VIEW ekstrakl_rank_strzelcow_19_20 ( ranking, pilkarz, klub, gole ) AS
SELECT RANK() OVER(ORDER BY COUNT(*) DESC) ranking,
(imie||' '||nazwisko) pilkarz,
kluby.nazwa klub,
COUNT(*) gole
FROM gole
NATURAL JOIN zawodnicy
NATURAL JOIN spotkania
JOIN kluby USING(id_klubu)
WHERE id_sezonu = (
    SELECT id_sezonu
    FROM sezony
    JOIN ligi USING(id_ligi)
    WHERE ligi.nazwa = 'Ekstraklasa' AND sezony.nazwa = '2019/2020'
) AND samobojczy = 'NIE'
GROUP BY id_zawodnika, imie, nazwisko, kluby.nazwa ;
```

Fragment zawartości perspektywy:

	RANKING	PILKARZ	KLUB	GOLE
1	1	Bronisław Szustakowski	Niebiescy Kielce	3
2	2	Daniel Blajer	Czarni Kraków	2
3	2	Czesław Bronowski	Czarni Kraków	2
4	4	Mykola Popa	Wielkopolska Poznań	1
5	4	Jacek Deluga	Czerwoni Łódź	1
6	4	Witold Ochota	Wisła Warszawa	1

EKSTRAKLASA_TABELA_19_20 - perspektywa zawierająca ranking drużyn piłkarskich w przykładowej lidze “Ekstraklasa” w sezonie “2019/2020” (ze względu na spory rozmiar kodu zrezygnowano z dodatkowego podzapytania i wpisano id_sezonu = 10). Zawiera kolumny miejsce, klub, wygrane, remisy, przegrane, zdobyte_bramki, stracone_bramki, roznica_bramek i punkty. Miejsca poszczególnych drużyn ustalane są kolejno według kolumn: punkty, roznica_bramek i zdobyte_bramki.

```
CREATE OR REPLACE VIEW ekstraklasa_tabela_19_20
( miejsce, klub, wygrane, remisy, przegrane, zdobyte_bramki, stracone_bramki, roznica_bramek, punkty ) AS
SELECT RANK() OVER(ORDER BY SUM(punkty) DESC, SUM(zdobyte)-SUM(stracone) DESC, SUM(zdobyte) DESC) miejsce,
nazwa klub,
SUM(wygrana) wygrane, SUM(remis) remisy, SUM(przegrana) przegrane,
SUM(zdobyte) zdobyte_bramki, SUM(stracone) stracone_bramki, (SUM(zdobyte)-SUM(stracone)) roznica_bramek,
SUM(punkty) punkty
FROM (
    SELECT gospodarz id_klubu, gospodarz_wynik zdobyte, gosc_wynik stracone,
    CASE
    WHEN gospodarz_wynik > gosc_wynik THEN 3
    WHEN gospodarz_wynik < gosc_wynik THEN 0
    ELSE 1 END punkty,
    CASE WHEN gospodarz_wynik > gosc_wynik THEN 1 ELSE 0 END wygrana,
    CASE WHEN gospodarz_wynik = gosc_wynik THEN 1 ELSE 0 END remis,
    CASE WHEN gospodarz_wynik < gosc_wynik THEN 1 ELSE 0 END przegrana
    FROM spotkania WHERE id_sezonu = 10 AND rozegrane = 'TAK'
    UNION
    SELECT gosc klub, gosc_wynik zdobyte, gospodarz_wynik stracone, CASE
    WHEN gosc_wynik > gospodarz_wynik THEN 3
    WHEN gosc_wynik < gospodarz_wynik THEN 0
    ELSE 1 END punkty,
    CASE WHEN gosc_wynik > gospodarz_wynik THEN 1 ELSE 0 END wygrana,
    CASE WHEN gospodarz_wynik = gosc_wynik THEN 1 ELSE 0 END remis,
    CASE WHEN gosc_wynik < gospodarz_wynik THEN 1 ELSE 0 END przegrana
    FROM spotkania WHERE id_sezonu = 10 AND rozegrane = 'TAK'
)
JOIN kluby USING(id_klubu)
GROUP BY id_klubu, nazwa;
```

Fragment zawartości perspektywy:

	MIEJSCE	KLUB	WYGRANE	REMISY	PRZEGRANE	ZDOBYTE_BRAMKI	STRACONE_BRAMKI	ROZNICA_BRAMEK	PUNKTY
1	1	Czarni Kraków	1	0	0	5	1	4	3
2	2	Niebiescy Kielce	1	0	0	3	1	2	3
3	3	Małopolska Kraków	1	0	0	2	0	2	3
4	3	Czerwoni Łódź	1	0	0	2	0	2	3
5	5	Pomarańczowi Płock	1	0	0	1	0	1	3
6	6	Odra Szczecin	0	1	0	1	1	0	1
7	6	Śląsk Gliwice	0	1	0	1	1	0	1
8	6	Podlasie Białystok	0	1	0	1	1	0	1
9	6	Warta Częstochowa	0	1	0	1	1	0	1
10	10	Bałtyk Gdańsk	0	1	0	0	0	0	1
11	10	Żółci Lubin	0	1	0	0	0	0	1
12	12	Śląsk Zabrze	0	0	1	0	1	-1	0

EKSTRAKLASA_TERMINARZ_19_20 - perspektywa prezentuje terminarz spotkań w przykładowej lidze “Ekstraklasa” w sezonie “2019/2020” (ze względu na spory rozmiar kodu zrezygnowano z dodatkowego podzapytania i wpisano id_sezonu = 10). Zawiera kolumny gospodarz, wynik, gosc, data_spotkania, gole i kartki. Jeśli spotkanie zostało rozegrane to prezentowane są informacje o golach i kartkach w meczu, oraz wyniku.

```
CREATE OR REPLACE VIEW ekstraklasa_terminarz_19_20 ( gospodarz, wynik, gosc, data_spotkania, gole, kartki ) AS
SELECT (SELECT nazwa FROM kluby WHERE id_klubu = gospodarz) gospodarz,
CASE
WHEN rozegrane = 'NIE' THEN '-:-'
WHEN rozegrane = 'TAK' THEN (gospodarz_wynik||':'||gosc_wynik)
END wynik,
(SELECT nazwa FROM kluby WHERE id_klubu = gosc) gosc,
poczatek data_spotkania, nvl(gole, ' ') gole, nvl(kartki, ' ') kartki
FROM spotkania
LEFT JOIN (
    SELECT id_spotkania,
    LISTAGG(nazwisko||' '||minuta||'''', ', ') WITHIN GROUP(ORDER BY minuta) gole
    FROM gole
    JOIN zawodnicy USING(id_zawodnika)
    JOIN kluby USING(id_klubu)
    GROUP BY id_spotkania
) USING(id_spotkania)
LEFT JOIN (
    SELECT id_spotkania,
    LISTAGG(nazwisko||'('||substr(kolor,1,1)||') '||minuta||'''', ', ') WITHIN GROUP(ORDER BY minuta) kartki
    FROM kartki
    JOIN zawodnicy USING(id_zawodnika)
    JOIN kluby USING(id_klubu)
    GROUP BY id_spotkania
) USING(id_spotkania)
WHERE id_sezonu = 10
ORDER BY data_spotkania ;
```

Fragment zawartości perspektywy:

	GOSPODARZ	WYNIK	GOSC	DATA_SPOTKANIA	GOLE	KARTKI
1	Bałtyk Gdańsk	0:0	Żółci Lubin	19/11/01		Korkus(Ż) 12', Rezler(Ż) 43', Korkus(C) 87', Kotaś(C) 88'
2	Odra Wrocław	0:2	Czerwoni Łódź	19/11/01	Deluga 79', Starmach 87'	Turczyk(Ż) 12', Starmach(Ż) 37', Likus(Ż) 89'
3	Wielkopolska Poznań	1:3	Niebiescy Kielce	19/11/02	Popa 4', Szustakowski 14', Szustakowski 27', Szusta...	Szałek(Ż) 54'
4	Wisła Warszawa	1:5	Czarni Kraków	19/11/02	Fiałkowski 17', Ochota 51', Blajer 55', Bronowski 5...	Leśnikowski(Ż) 32', Bis(Ż) 53', Kulus(Ż) 57', Dreszler(Ż) 76', G
5	Małopolska Kraków	2:0	Zieloni Gdynia	19/11/02	Frąszczak 59', Tonder 71'	
6	Śląsk Zabrze	0:1	Pomarańczowi Płock	19/11/03	Karkoszka 62'	Świdrak(Ż) 3'
7	Podlasie Białystok	1:1	Śląsk Gliwice	19/11/03	Rechnio 60', Chabasiński 67'	Rechnio(Ż) 54', Staszak(Ż) 61'
8	Odra Szczecin	1:1	Warta Częstochowa	19/11/03	Deć 79', Kusy 83'	Ronowicz(Ż) 2', Adamiuk(Ż) 13', Myszka(C) 50'
9	Odra Wrocław	-:-	Żółci Lubin	19/11/08		
10	Bałtyk Gdańsk	-:-	Niebiescy Kielce	19/11/08		
11	Małopolska Kraków	-:-	Czarni Kraków	19/11/09		
12	Wielkopolska Poznań	-:-	Zieloni Gdynia	19/11/09		
13	Wisła Warszawa	-:-	Pomarańczowi Płock	19/11/09		
14	Śląsk Zabrze	-:-	Śląsk Gliwice	19/11/10		
15	Odra Szczecin	-:-	Czerwoni Łódź	19/11/10		

RANKING_GOLI_W_JEDNYM_MECZU - perspektywa prezentuje ranking piłkarzy pod względem liczby goli strzelonych w trakcie jednego meczu. Zawiera kolumny ranking, pilkarz, mecz, data_spotkania, liczba_goli i gole (zawiera minuty, w których zostały strzelone gole).

```
CREATE OR REPLACE VIEW ranking_goli_w_jednym_meczu
( ranking, pilkarz, mecz, data_spotkania, liczba_goli, gole ) AS
SELECT RANK() OVER(ORDER BY COUNT(*) DESC) ranking, (imie||' '||nazwisko) pilkarz,
((SELECT nazwa FROM kluby WHERE id_klubu = gospodarz)
||' - '||(SELECT nazwa FROM kluby WHERE id_klubu = gosc)) mecz,
poczatek data_spotkania, COUNT(*) liczba_goli,
LISTAGG(minuta, ''' ') WITHIN GROUP(ORDER BY minuta)||'''' gole
FROM gole
NATURAL JOIN zawodnicy
NATURAL JOIN spotkania
WHERE samobojczy = 'NIE'
GROUP BY id_spotkania, id_zawodnika, imie, nazwisko, gospodarz, gosc, poczatek ;
```

Fragment zawartości perspektywy:

	RANKING	PILKARZ	MECZ	DATA_SPOTKANIA	LICZBA_GOLI	GOLE
1	1	Bronisław Szustakowski	Wielkopolska Poznań - Niebiescy Kielce	19/11/02	3	14' 27' 88'
2	2	Czesław Bronowski	Wisła Warszawa - Czarni Kraków	19/11/02	2	57' 90'
3	2	Daniel Blajer	Wisła Warszawa - Czarni Kraków	19/11/02	2	55' 77'
4	4	Zbigniew Tonder	Małopolska Kraków - Zieloni Gdynia	19/11/02	1	71'
5	4	Kamil Kusy	Odra Szczecin - Warta Częstochowa	19/11/03	1	83'
6	4	Volodymyr Karkoszka	Śląsk Zabrze - Pomarańczowi Płock	19/11/03	1	62'
7	4	Przemysław Rechnio	Podlasie Białystok - Śląsk Gliwice	19/11/03	1	60'
8	4	Karol Chabasiński	Podlasie Białystok - Śląsk Gliwice	19/11/03	1	67'

9. Usuwanie danych

```
DROP VIEW Ekstrakl_rank_strzelcow_19_20 CASCADE CONSTRAINTS ;

DROP VIEW Ekstraklasa_tabela_19_20 CASCADE CONSTRAINTS ;

DROP VIEW Ekstraklasa_terminarz_19_20 CASCADE CONSTRAINTS ;

DROP VIEW Ranking_goli_w_jednym_meczu CASCADE CONSTRAINTS ;

DROP TABLE gole CASCADE CONSTRAINTS;

DROP TABLE kartki CASCADE CONSTRAINTS;

DROP TABLE kluby CASCADE CONSTRAINTS;

DROP TABLE ligi CASCADE CONSTRAINTS;

DROP TABLE sezony CASCADE CONSTRAINTS;

DROP TABLE sklady CASCADE CONSTRAINTS;

DROP TABLE spotkania CASCADE CONSTRAINTS;

DROP TABLE zawodnicy CASCADE CONSTRAINTS;

DROP SEQUENCE seq_id_zaw;

DROP SEQUENCE seq_id_gola;

DROP SEQUENCE seq_id_kartki;
```