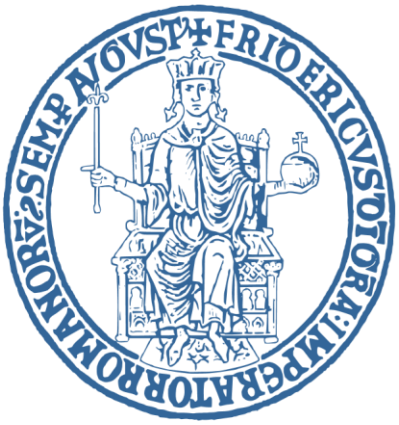


UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Documentazione progetto Basi di dati. A.A. 2023/2024

Sistema informativo dei calciatori

Autori: Orlovskyi Glib, Valentino Ferdinando, Scognamiglio Gianluca
Matricole: N86004789, N86004645, N86004467

Indice

1 Progettazione Concettuale

1.1 Analisi dei requisiti

Il primo punto da sviluppare, prima di passare alla concreta realizzazione del progetto, è quella di analizzare la richiesta del nostro cliente. Nello specifico ci è stato richiesto di realizzare una base di dati relazionale, e di un rispettivo programma applicativo, che raccolga dati relativi a dei giocatori di calcio.

"Si sviluppi un sistema informativo per la gestione di calciatori di tutto il mondo. Ogni calciatore è caratterizzato da nome, cognome, data di nascita, piede (sinistro, destro o ambidestro), uno o più ruoli di gioco (portiere, difensore, centrocampista, attaccante) e una serie di feature caratteristiche (ad esempio colpo di testa, tackle, rovesciata, etc.).

Il giocatore ha una carriera durante la quale può militare in diverse squadre di calcio. La militanza in una squadra è caratterizzata da uno o più periodi di tempo nei quali il giocatore era in quella squadra. Ogni periodo di tempo ha una data di inizio ed una data di fine.

Durante la militanza del giocatore nella squadra si tiene conto del numero di partite giocate, del numero di goal segnati e del numero di goal subiti (applicabile solo ai giocatori di ruolo portiere). Il giocatore può inoltre vincere dei trofei, individuali o di squadra.

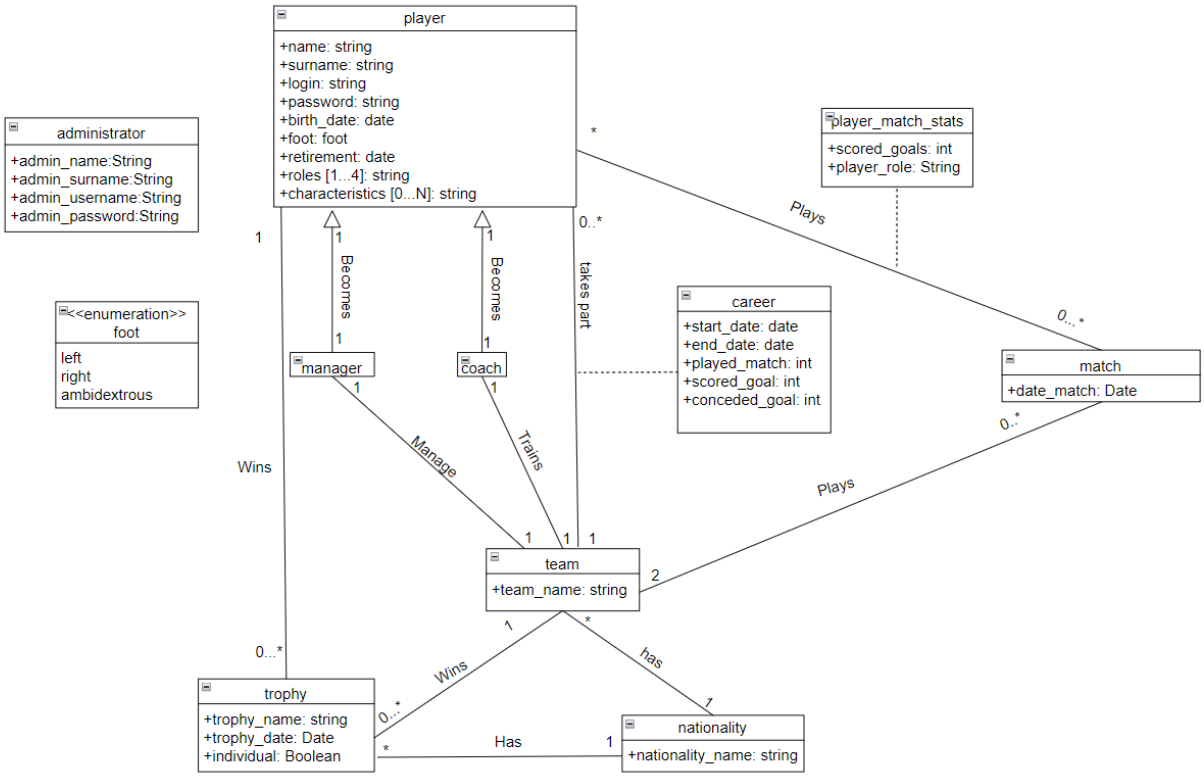
Il giocatore può avere anche una data di ritiro a seguito della quale decide di non giocare più. Le squadre di calcio sono specificate dal loro nome e nazionalità.

L'amministratore del sistema si identifica con un login ed una password e ha il diritto di inserire nuovi giocatori nella base di dati, modificarne i dati, aggiungere ulteriori informazioni oppure eliminare un giocatore. L'utente generico può vedere l'elenco dei giocatori e le loro caratteristiche e può richiedere diverse ricerche, ad esempio filtrando i giocatori per nome, per ruolo, per piede, per numero di goal segnati, per numero di goal subiti, per età, per squadre di appartenenza.

I giocatori dopo la fine della carriera possono diventare allenatori o dirigenti. Il sistema continua a mantenere una parte delle informazioni (squadra, numero di partite e trofei vinti) anche per allenatori e dirigenti.

Inoltre, può accedere al sistema anche un terzo tipo di utente, consistente nel Giocatore stesso. Egli ha un suo login e password e può modificare unicamente i dati relativi a sé stesso.

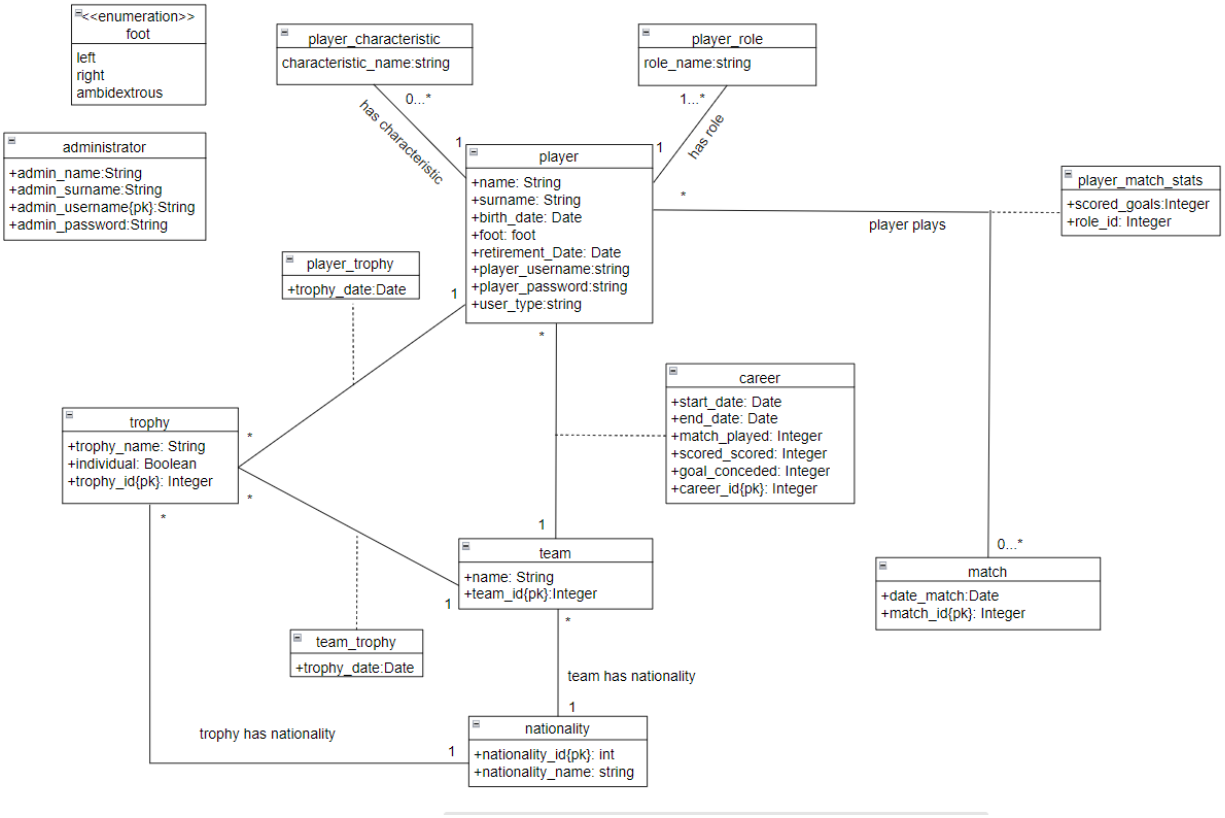
1.2 Class diagram



Schema concettuale del DB ricavato dalle informazioni viste durante l'analisi dei requisiti. La classe del giocatore è quella centrale sulla quale vertono tutte le altre. Un giocatore viene associato ad un team con una classe di associazioni carriera che ne racchiude le informazioni più importanti di quella militanza. Il giocatore possiede un piede dominante o può essere ambidestro, l'attributo piede è un enum che racchiude i valori accettabili. Il giocatore potrà possedere anche diversi ruoli (portiere, difensore, attaccante e centrocampista). Può possedere più caratteristiche. La classe di associazione carriera terrà in conto dei match giocati durante quella militanza, dei goal segnati e dei goal subiti nel caso di un portiere. Durante la sua carriera il giocatore può militare diverse volte in diverse squadre di calcio, in periodi differenti. Nella classe squadra si terrà conto della nazionalità e del nome della stessa. il giocatore può militare in una sola squadra per volta. Coach e manager sono delle sottoclassi di giocatore ed erediteranno i suoi attributi.

2 Ristrutturazione

2.1 Fase di ristrutturazione



2.1.1 Analisi delle Ridondanze

Una ridondanza è un dato che si ripete per più di una volta nel database o può essere derivato da altri dati.

Nello schema presentato sono presenti diverse ridondanze, legate alla necessità di memorizzare la quantità di partite giocate, goal fatti e goal subiti. Così la tabella career contiene gli attributi “match_played”, “goal_scored” e “goal_conceded”, tutti ricavabili dalla tabella “player_match_stats”. Abbiamo deciso di preservare queste ridondanze poiché le richieste presentate a noi riguardo la struttura del progetto implicano un accesso frequente per la lettura di questi dati, e una lettura frequente dalla tabella “player_match_stats” consuma una quantità di risorse decisamente maggiore rispetto a quella impiegata per l’occasionale calcolo della somma di queste informazioni per ogni singola carriera.

2.1.2 Analisi degli Identificativi

Per le entità trophy, nationality, team, match e la classe di associazione career si è deciso di usare una chiave surrogata, questo perché nessuno dei loro attributi possedeva i requisiti necessari per svolgere il ruolo della primary key.

Per le entità player e administrator siamo riusciti ad utilizzare le natural key, grazie alla presenza di attributi username.

2.1.3 Rimozione degli Attributi Multivalore

La rimozione degli attributi multivalore consiste nell'identificare all'interno delle tabelle gli attributi che contengono più di un valore per casella e normalizzarli esportandoli in una nuova tabella associata alla tabella originale, nel nostro caso lo facciamo tramite una tabella che va ad associare un attributo con l'username del giocatore.

Questo procedimento è stato fatto con i ruoli e le caratteristiche.

2.1.4 Rimozione degli Attributi Composti

La rimozione degli attributi composti consiste nella scomposizione delle colonne contenenti più informazioni per cella all'interno della stessa tabella. All'interno del nostro schema non sono presenti attributi composti

2.1.5 Accorpamenti o partizionamenti delle entità

La partizione delle entità è un processo di scomposizione di un'entità più grande e complessa in una serie di entità più piccole e specializzate. Abbiamo applicato questo processo all'entità trophy, suddividendola in player_trophies e team_trophies, separando così i trofei vinti dai giocatori singoli dai trofei vinti dalla squadra intera.

2.1.6 Rimozione delle generalizzazioni

Il processo dell'eliminazione delle generalizzazioni consiste nell'unificazione delle entità figlie che ereditano una parte della struttura da una classe padre. Nel nostro caso le uniche relazioni figlie sono coach e manager, le quali non aggiungono nuove informazioni, se non un flag booleano che va a identificare un utente come coach or manager, per questo motivo andiamo ad unificare questa tabella con la classe padre, la tabella player, aggiungendo l'attributo user_type che va ad identificare il tipo di utente.

3 Dizionari

3.1 Dizionario delle entità

Entità	Descrizione	Attributi
administrator	Rappresenta gli amministratori del sistema che hanno privilegi di gestione e controllo.	admin_name (String) admin_surname (String) admin_username (String,PK) admin_password (String)
player	Rappresenta i giocatori con informazioni personali, statistiche di carriera e dati di autenticazione.	name (String) surname (String) birth_date (Date) foot (foot ENUM) retirement_date (Date) player_username (String,PK) player_password (String) user_type (String)
team	Rappresenta le squadre con il nome, la nazionalità e i dettagli di allenatore e manager attuali.	team_name (String) team_id (Integer, PK)
trophy	Rappresenta i trofei che possono essere vinti da giocatori o squadre, distinguendo tra trofei individuali e di squadra.	trophy_name (String) individual (Boolean) trophy_id (Integer, PK)
player_characteristic	Associa le caratteristiche a un giocatore	characteristic_name (String)
player_role	Associa i ruoli al giocatore	role_name (String)
career	Classe di associazione di una o più squadre ai giocatori che ne fanno o facevano parte, con l'aggiunta di statistiche inerenti ad ogni periodo di militanza.	start_date (Date) end_date (Date) match_played (Integer) goal_scored (Integer) goal_conceded (Integer) career_id (Integer, PK)
match	Rappresenta le informazioni relative alle partite giocate tra squadre	date_match (Date) match_id (Integer PK)
player_match_stats	Classe di associazione di più giocatori alle partite nelle quali hanno partecipato, con l'aggiunta di	scored_goals (Integer) role_id (Integer, FK)
player_trophy	Classe di associazione tra il giocatore e il trofeo vinto	trophy_date (Date)
team_trophy	Classe di associazione tra la squadra e il trofeo vinto	trophy_date (Date)
nationality	Rappresenta le nazionalità associabili alle squadre e ai trofei	nationality_id (Integer,pk) nationality_name (String)

3.2 Dizionario delle associazioni

Associazione	Classi coinvolte	Descrizione
Trophy has nationality	Nationality[1] Trophy[*] Questa è una relazione uno a molti	Associa le caratteristiche ai giocatori
has characteristic	Player [1] Player_characteristics [0..*] Questa è una relazione uno a molti	Associa le caratteristiche ai giocatori
Team has nationality	Nationality[1] Trophy[*] Questa è una relazione uno a molti	Associa i trofei alle nazioni di appartenenza
has role	Player [1] Player_roles [1..*] Questa è una relazione uno a molti	Associa i ruoli ai giocatori

3.3 Dizionario dei Vincoli

Vincolo	Tipologia	Descrizione
end_check_date	Interrelazionale	Verifica che la fine della carriera sia minore o uguale della data corrente
unique_player_characteristic	Interrelazionale	Si assicura che ogni combinazione di player_username e characteristic_name sia unica
unique_player_role	Interrelazionale	Si assicura che ogni combinazione di player_username e role_name sia unica
user_type_check	Interrelazionale	Si assicura che il valore inserito in user_type sia uno dei valori ammissibili
role_name_check	Interrelazionale	Si assicura che il valore inserito in role_name sia uno dei valori ammissibili
characteristic_name_check	Interrelazionale	Si assicura che il valore inserito in characteristic_name sia uno dei valori ammissibili

4 Modello Logico

4.1 Schema logico

Legenda: Primary key, Foreign Key, Derived attribute

player_trophy (player_username, trophy_id, trophy_date)
player_username →player.player_username, trophy_id→trophy.trophy_id

team_trophy (team_id, trophy_id, trophy_date)
team_id →team.team_id, trophy_id→trophy.trophy_id

trophy (trophy_name, individual, trophy_id, nationality_id)
nationality_id→nationality.nationality_id

team (name, team_id, nationality_id, current_manager, current_coach)
nationality_id→nationality.nationality_id
current_manager → player.player_username
current_coach → player.player_username

player_characteristic (characteristic_name, player_username)
player_username → player.player_username

player_role (role_name, player_username)
player_username→player.player_username

career (start_date, end_date, match_played, scored_goals, goal_conceded, career_id, player_username, team_id)
player_username→player.player_username , team_id→team.team_id

player_match_stats (scored_goals, role, player_username, match_id)
player_username→player.player_username, match_id→match.match_id

match (date_match, match_id)

nationality (nationality_id, nationality_name)

player (name, surname, birth_date, foot, retierement_date, player_username, player_password, user_type)

administrator (admin_name, admin_surname, admin_username, admin_password)

5 Definizioni SQL

-- Creazione di un ENUM che contiene i valori accettabili per il piede principale del giocatore

```
CREATE TYPE foot AS ENUM ('left', 'right', 'ambidextrous');
```

5.1 Tabelle

-- Creazione della tabella amministratore

```
CREATE TABLE administrator
(
   admin_name     VARCHAR(64) NOT NULL,
   admin_surname  VARCHAR(64) NOT NULL,
   admin_username VARCHAR(64) PRIMARY KEY,
   admin_password VARCHAR(64) NOT NULL
)
```

-- Creazione della tabella carriera

Il constraint su end_date si assicura che la data della fine della carriera non possa essere più tardi della data odierna

```
CREATE TABLE CAREER
(
   player_username VARCHAR(64) REFERENCES player NOT NULL,
   start_date      DATE             NOT NULL,
   end_date        DATE,
   match_played    integer          NOT NULL DEFAULT 0,
   scored_goals    integer          NOT NULL DEFAULT 0,
   goal_conceded   integer          NOT NULL DEFAULT 0,
   career_id       serial PRIMARY KEY NOT NULL,
   team_id         integer REFERENCES team

)
```

-- Creazione della tabella caratteristiche

```
CREATE TABLE characteristic
(
   characteristic_id SERIAL UNIQUE PRIMARY KEY,
   characteristic  VARCHAR(255) NOT NULL
);
```

-- Creazione della tabella caratteristiche giocatore

Il constraint previene l'associazione di una caratteristica ad un giocatore per più di una volta

```
CREATE TABLE player_characteristic
(
   player_username     VARCHAR(64) REFERENCES player NOT NULL,
   characteristic_name VARCHAR(64) NOT NULL,
   UNIQUE (player_username, characteristic_name),
   CONSTRAINT characteristic_name_check CHECK ( characteristic_name = 'head shot' OR
                                    characteristic_name = 'overhead kick' OR
                                    characteristic_name = 'tackle' OR characteristic_name = 'cross')
```

```
)
```

-- Creazione della tabella match
Il constraint previene l'inserimento di valori negativi alle colonne dei goal

```
CREATE TABLE match
(
   match_id serial PRIMARY KEY NOT NULL,
   date_match DATE NOT NULL
)
```

-- Creazione della tabella nazionalità

```
CREATE TABLE nationality
(
   nationality_id serial primary key not null,
   nationality_name VARCHAR(64)
)
```

-- Creazione della tabella giocatore

```
CREATE TABLE player
(
   player_name     VARCHAR(64) NOT NULL,
   player_surname  VARCHAR(64) NOT NULL,
   birth_date      DATE        NOT NULL,
   foot            foot,
   retirement_date DATE,
   player_username VARCHAR(64) NOT NULL PRIMARY KEY,
   player_password VARCHAR(64) NOT NULL,
   user_type       VARCHAR(64) NOT NULL,
   CONSTRAINT user_type_check CHECK ( user_type = 'player' OR user_type = 'coach' OR user_type = 'manager' )
);
```

-- Creazione della tabella statistiche dei match dei giocatori

```
CREATE TABLE player_match_stats
(
   player_username VARCHAR(64) REFERENCES player NOT NULL,
   match_id        integer REFERENCES match,
   role_name       VARCHAR(64) NOT NULL,
   scored_goals    integer     NOT NULL DEFAULT 0
      CONSTRAINT role_name_check CHECK ( role_name = 'goalkeeper' OR
                                 role_name = 'midfielder' OR
                                 role_name = 'forward' OR role_name = 'defender')
)
```

-- Creazione della tabella ruolo giocatore
Il constraint previene l'associazione di un ruolo ad un giocatore per più di una volta

```
CREATE TABLE player_role
(
   player_username VARCHAR(64) REFERENCES player NOT NULL,
   role_name       VARCHAR(64) NOT NULL,
   UNIQUE (player_username, role_name),
   CONSTRAINT role_name_check CHECK ( role_name = 'goalkeeper' OR
```

```
                        role_name = 'midfielder' OR
                        role_name = 'forward' OR role_name = 'defender')
)
```

-- Creazione della tabella trofei giocatore

```
CREATE TABLE player_trophy
(
   trophy_date     DATE NOT NULL,
   player_username VARCHAR(64) REFERENCES player,
   trophy_id integer REFERENCES trophy,
   UNIQUE (player_username, trophy_id)
)
```

-- Creazione della tabella ruoli

```
 CREATE TABLE rolelist
(
   role_id SERIAL UNIQUE PRIMARY KEY,
   role_name VARCHAR(255)NOT NULL
);
```

-- Creazione della tabella delle squadre

```
CREATE TABLE team
(
   team_name VARCHAR(64) NOT NULL,
   team_id  SERIAL PRIMARY KEY,
   nationality_id  INTEGER REFERENCES nationality,
   current_manager VARCHAR(64) REFERENCES player NOT NULL,
   current_coach   VARCHAR(64) REFERENCES player NOT NULL
);
```

-- Creazione della tabella trofei di squadra

```
CREATE TABLE team_trophy
(
   trophy_date     DATE NOT NULL,
   team_id integer REFERENCES team,
   trophy_id integer REFERENCES trophy,
   UNIQUE (team_id, trophy_id)
)
```

-- Creazione della tabella trofei

```
CREATE TABLE trophy
(
   trophy_name VARCHAR(64) NOT NULL,
   individual boolean NOT NULL,
   trophy_id serial PRIMARY KEY NOT NULL,
   nationality_id integer REFERENCES nationality
)
```

5.2 Trigger

Trigger sulla tabella career che attiva *check_start_date();*

```
CREATE TRIGGER career_start_date_check
BEFORE INSERT OR UPDATE ON career
FOR EACH ROW EXECUTE PROCEDURE check_start_date();
```

Trigger sulla tabella career che attiva *update_goalkeeper_stats();*

```
CREATE TRIGGER career_update_goalkeeper_conceded_trigger
AFTER INSERT OR UPDATE ON career
FOR EACH ROW EXECUTE PROCEDURE update_goalkeeper_stats();
```

Trigger sulla tabella player che attiva *reset_player_stats_on_retirement();*

```
CREATE TRIGGER reset_stats_on_retirement_trigger
AFTER UPDATE ON player
FOR EACH ROW EXECUTE PROCEDURE reset_player_stats_on_retirement();
```

Trigger sulla tabella player_match_stats che attiva *update_career_goals*();

```
CREATE TRIGGER update_career_goals_trigger
AFTER INSERT OR UPDATE ON player_match_stats
FOR EACH ROW EXECUTE PROCEDURE update_career_goals();
```

Trigger sulla tabella player_match_stats che attiva *update_matches_played*();

```
CREATE TRIGGER update_matches_played_trigger
AFTER INSERT OR UPDATE ON player_match_stats
FOR EACH ROW EXECUTE PROCEDURE update_matches_played();
```

Trigger sulla tabella team che attiva *check_manager_coach_validity*();

```
CREATE TRIGGER check_manager_coach_constraint
BEFORE INSERT OR UPDATE ON team
FOR EACH ROW EXECUTE PROCEDURE check_manager_coach_validity();
```

Trigger sulla tabella team_trophy che attiva validate_trophy_nationality();

```
CREATE TRIGGER validate_trophy_nationality_trigger
AFTER INSERT OR UPDATE ON team_trophy
```

```
FOR EACH ROW EXECUTE PROCEDURE validate_trophy_nationality();
```

Trigger sulla tabella player_trophy che attiva validate_player_trophy_nationality();

```
CREATE TRIGGER validate_player_trophy_nationality_trigger
AFTER INSERT OR UPDATE ON player_trophy
FOR EACH ROW EXECUTE PROCEDURE validate_player_trophy_nationality();
```

5.3 Funzioni

– La funzione verifica se il giocatore che vuole diventare coach o manager si sia già ritirato e se svolge già un ruolo simile

```
CREATE FUNCTION check_manager_coach_validity()
  RETURNS trigger
  LANGUAGE plpgsql AS
$$
BEGIN
  IF (NEW.current_manager IS NOT NULL AND
    NOT EXISTS (SELECT 1
          FROM player
          WHERE player.player_username = NEW.current_manager
           AND player.retirement_date IS NOT NULL
           AND (player.user_type = 'Manager'))
    )
  THEN
    RAISE EXCEPTION 'current_manager must be retired and must be a manager';
  END IF;

  IF (NEW.current_coach IS NOT NULL AND
    NOT EXISTS (SELECT 1
          FROM player
          WHERE player.player_username = NEW.current_coach
           AND player.retirement_date IS NOT NULL
           AND (player.user_type = 'Coach' ))
    )
  THEN
    RAISE EXCEPTION 'current_coach must be retired and must be a coach';
  END IF;

  RETURN NEW;
END;
$$;
```

– La funzione si assicura che la data d'inizio di una nuova militanza non sia pari o maggiore al valore della data della fine della sua carriera

```
CREATE FUNCTION check_start_date()
   RETURNS trigger
   LANGUAGE plpgsql AS
$$
BEGIN
   IF NEW.start_date >= (SELECT retirement_date
                  FROM player
                  WHERE player_username = NEW.player_username)
   THEN
      RAISE EXCEPTION 'start_date cannot be later or equal to the retirement_date of the same player';
   END IF;

   RETURN NEW;
END;
$$;
```

– La funziona azzera i valori giù indicati del giocatore che si è appena ritirato

```
CREATE FUNCTION reset_player_stats_on_retirement()
RETURNS trigger
LANGUAGE plpgsql AS $$
BEGIN
   IF OLD.retirement_date IS NULL AND NEW.retirement_date IS NOT NULL THEN
      UPDATE player
      SET foot = NULL
      WHERE player_username = NEW.player_username;

      DELETE FROM career
      WHERE player_username = NEW.player_username;
   END IF;

   RETURN NEW;
END;
$$;
```

– La funzione aggiorna la quantità di goal effettuati durante un determinato periodo di militanza

```
CREATE FUNCTION update_career_goals()
RETURNS TRIGGER
LANGUAGE plpgsql AS $$
DECLARE
   match_date DATE;
   player_career_id INTEGER;
BEGIN
   SELECT date_match INTO match_date
   FROM match
   WHERE match_id = NEW.match_id;
```

```
    SELECT career_id INTO player_career_id
    FROM career
    WHERE player_username = NEW.player_username
      AND start_date <= match_date
      AND (end_date IS NULL OR end_date >= match_date)
    LIMIT 1;

    IF player_career_id IS NULL THEN
        RETURN NEW;
    END IF;

    UPDATE career
    SET scored_goals = scored_goals + NEW.scored_goals
    WHERE career_id = player_career_id;

    RETURN NEW;
END;
$$;
```

–La funzione aggiorna la quantità di goal subiti durante un determinato periodo di militanza

```
CREATE FUNCTION update_goalkeeper_stats()
    RETURNS TRIGGER
    LANGUAGE plpgsql AS
$$
DECLARE
    opponent_goals      INTEGER;
    goalkeeper_username VARCHAR(64);
    goalkeeper_team_id  INTEGER;
BEGIN
    SELECT c.team_id, pms.player_username
    INTO goalkeeper_team_id, goalkeeper_username
    FROM player_match_stats pms
            JOIN career c ON pms.player_username = c.player_username
    WHERE pms.match_id = NEW.match_id
      AND pms.role_name = 'goalkeeper'
    LIMIT 1;

    IF goalkeeper_team_id IS NULL THEN
        RETURN NEW;
    END IF;
    SELECT COALESCE(SUM(pms.scored_goals), 0)
    INTO opponent_goals
    FROM player_match_stats pms
            JOIN career c ON pms.player_username = c.player_username
    WHERE pms.match_id = NEW.match_id
      AND c.team_id != goalkeeper_team_id;
    UPDATE career
    SET goal_conceded = goal_conceded + opponent_goals
    WHERE player_username = goalkeeper_username;

    RETURN NEW;
END;
$$;
```

– La funzione calcola la quantità di partite giocate da un giocatore durante un determinato periodo di militanza

```
CREATE FUNCTION update_matches_played()
RETURNS trigger
LANGUAGE plpgsql AS $$
BEGIN
   UPDATE career
   SET match_played = (
      SELECT COUNT(*)
      FROM player_match_stats
      JOIN match ON player_match_stats.match_id = match.match_id
      WHERE player_match_stats.player_username = NEW.player_username
        AND match.date_match BETWEEN career.start_date AND COALESCE(career.end_date, CURRENT_DATE)
   )
   WHERE career.player_username = NEW.player_username;

   RETURN NULL;
END;
$$;
```

--Verifica che la nazionalità del trofeo assegnato alla squadra combaci con la nazionalità della squadra

```
CREATE FUNCTION validate_trophy_nationality()
   RETURNS TRIGGER
   LANGUAGE plpgsql AS
$$
DECLARE
   team_nationality   INTEGER;
   trophy_nationality INTEGER;
BEGIN

   SELECT nationality_id
   INTO team_nationality
   FROM team
   WHERE team_id = NEW.team_id;

   SELECT nationality_id
   INTO trophy_nationality
   FROM trophy
   WHERE trophy_id = NEW.trophy_id;

   IF trophy_nationality IS NOT NULL AND trophy_nationality <> team_nationality THEN
      RAISE EXCEPTION 'The trophy nationality does not match the team nationality';
   END IF;
```

```
    RETURN NEW;
END;
$$;
```

--Verifica che la nazionalità del trofeo assegnato al giocatore combaci con la nazionalità della squadra di cui faceva parte

```
CREATE FUNCTION validate_player_trophy_nationality()
RETURNS TRIGGER
LANGUAGE plpgsql AS $$
DECLARE
    player_team_id INTEGER;
    team_nationality INTEGER;
    trophy_nationality INTEGER;
BEGIN

    SELECT team_id INTO player_team_id
    FROM career
    WHERE player_username = NEW.player_username
      AND start_date <= NEW.trophy_date
      AND (end_date IS NULL OR end_date >= NEW.trophy_date)
    LIMIT 1;

    IF player_team_id IS NULL THEN
        RAISE EXCEPTION 'No active team found for player at the given trophy date';
    END IF;

    SELECT nationality_id INTO team_nationality
    FROM team
    WHERE team_id = player_team_id;

    SELECT nationality_id INTO trophy_nationality
    FROM trophy
    WHERE trophy_id = NEW.trophy_id;

    IF trophy_nationality IS NOT NULL AND trophy_nationality <> team_nationality THEN
        RAISE EXCEPTION 'The trophy nationality does not match the team nationality';
    END IF;

    RETURN NEW;
END;
$$;
```