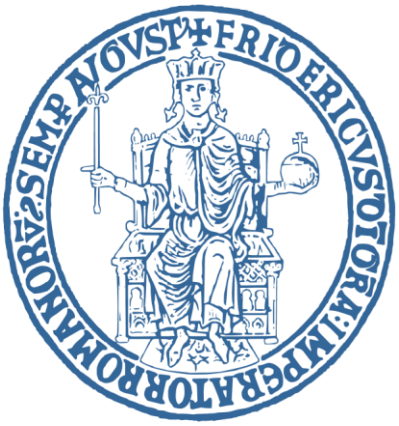


UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

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.

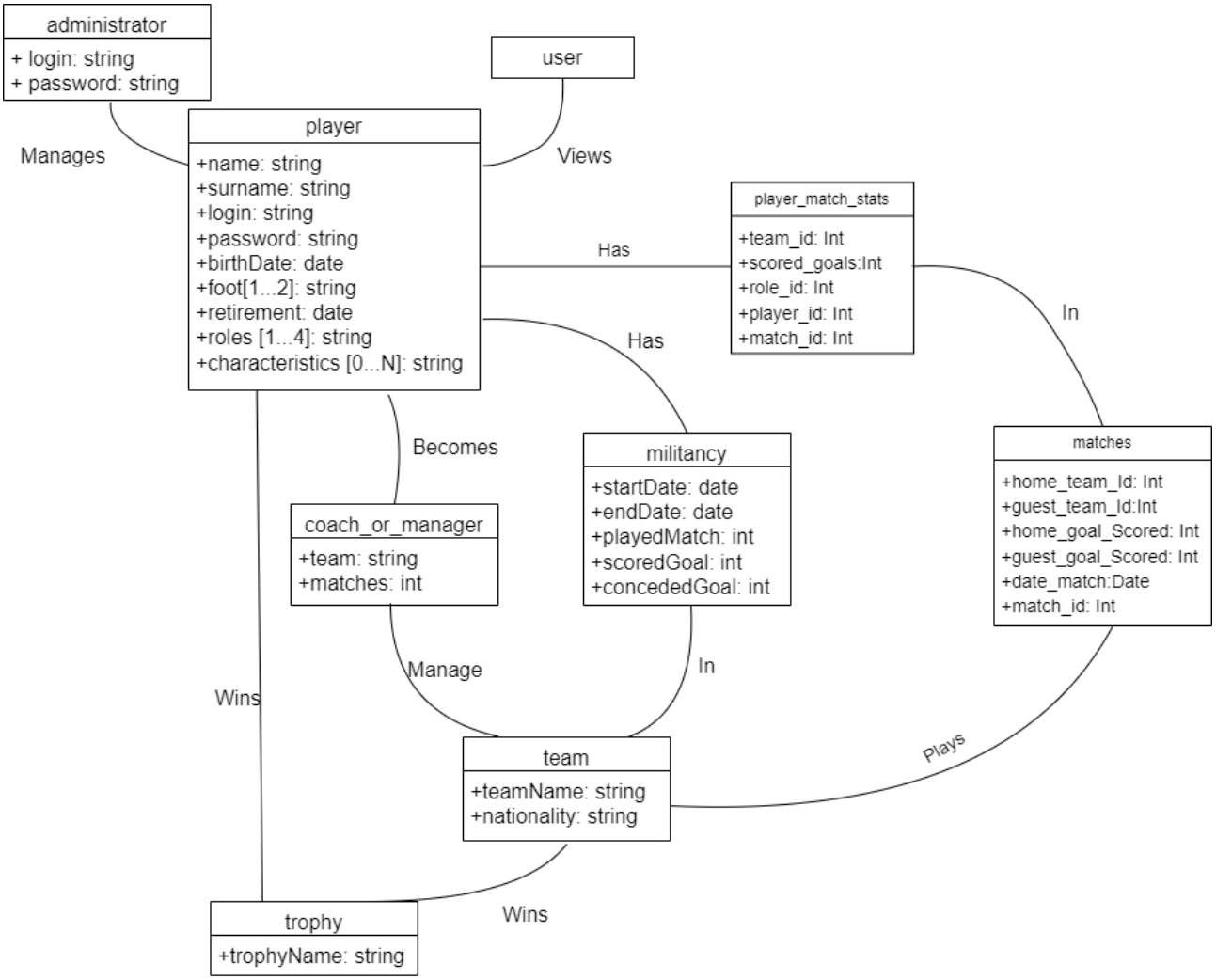
L'amministratore del sistema si identifica con un login ed una password e ha il diritto di inserire nuovi giocatori nella base di dati, modificarne i dati, aggiungere ulteriori informazioni oppure eliminare un giocatore. L'utente generico può vedere l'elenco dei giocatori e le loro caratteristiche e 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

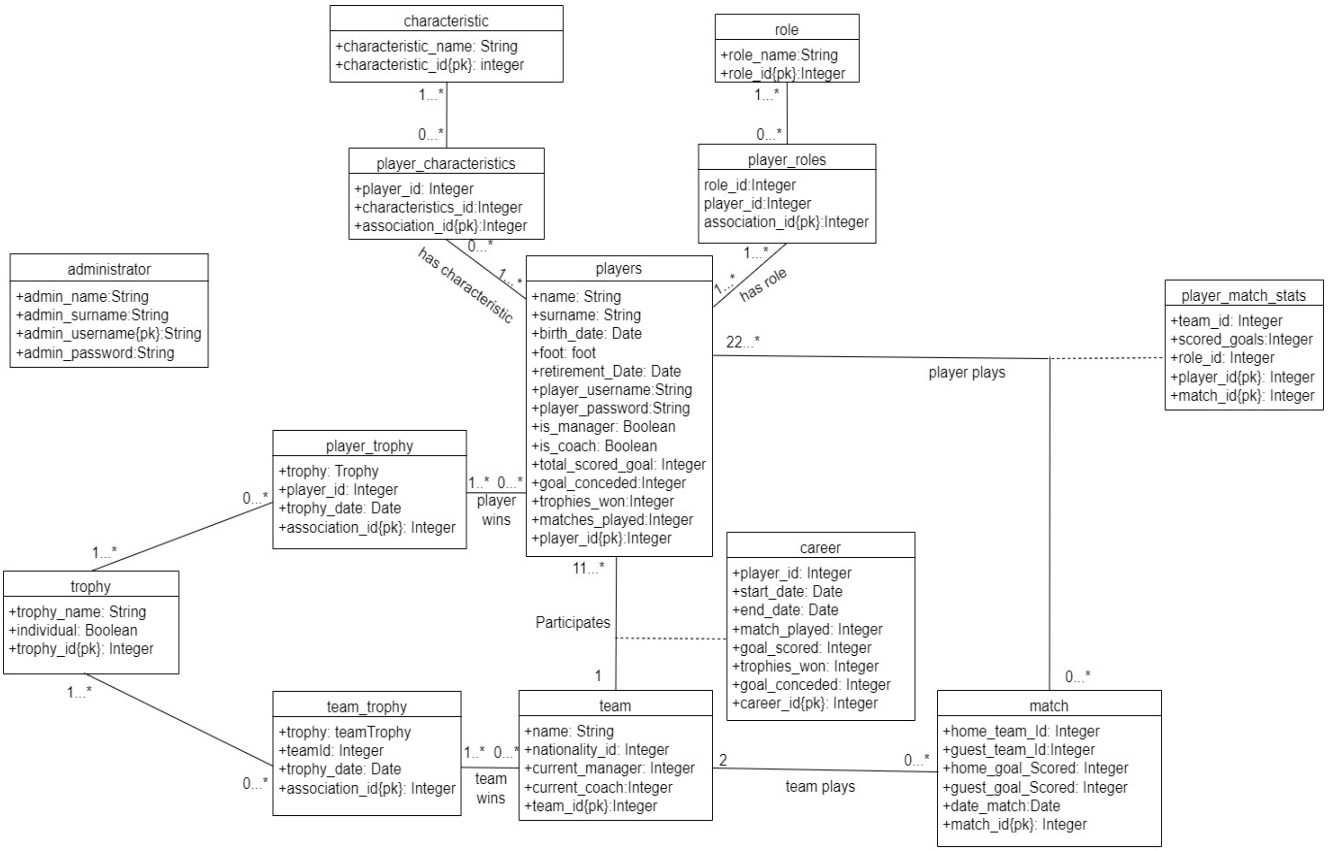
Schema concettuale del DB ricavato dalle informazioni viste durante l'analisi dei requisiti.



La classe del giocatore è forse quella più importante dove vertono tutte le altre. Un giocatore viene associato ad una carriera, con una data d'inizio e una data di fine, la relazione tra il giocatore e la carriera è molti a una. Il giocatore possiede un piede dominante o può essere ambidestro, il piede è posseduto da un giocatore, l'attributo piede è di tipo piede e verrà specificato proprio se usa il destro, il sinistro o entrambi. Il giocatore potrà possedere anche diversi ruoli (portiere, difensore, attaccante e centrocampista), anche qui l'attributo ruoli è di tipo ruoli. Infine, potrebbe possedere anche diverse caratteristiche, oppure neanche una. Caratteristiche è un attributo di tipo caratteristiche. La classe della carriera terrà in conto dei trofei che il giocatore ha vinto individualmente e dei trofei che ha vinto con le squadre in cui ha militato. Durante la sua carriera il giocatore può militare diverse volte in diverse squadre di calcio, in periodi differenti. Nella classe della militanza, quindi, terremo conto del periodo ti tempo in cui il giocatore è stato nella squadra. Periodo è un attributo multivalore, il giocatore potrebbe anche cambiare squadra e ritornare nella vecchia successivamente, il giocatore può militare minimo 1 volta fino ad un numero "n" non definito di volte in una squadra(1: [1…*]), ma ad ogni periodo è associata una militanza. La squadra è un attributo appartenente alla militanza. 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 alcuni dei suoi attributi. Stessa cosa vale per le classi giocatore e amministratore, sono delle sottoclassi di utente.

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 l'unica ridondanza presente è la colonna totalgoalscored in player, nonostante sia possibile ricavare lo stesso dato facendo la somma di tutti i goal fatti da un giocatore registrati nella tabella player_match_stats, questa operazione con il riempimento della database diventerà sempre più costosa dal punto di vista della performance, poiché dovrà andare a sommare sempre più record di goal fatti, con il sistema attuale invece questo calcolo viene fatto solamente quando c'è un insert, un update o un delete alla tabella player_match_stats, ottimizzando così il carico sulla database

2.1.2 Analisi degli Identificativi

Per la maggior parte delle tabelle si è deciso di utilizzare una chiave primaria surrogata a causa dell'assenza di unicità o invariabilità nelle colonne. L'unica eccezione è la tabella

player_match_stats, nella quale viene usata una chiave primaria composta (player_id e match_id), questo perché nel nostro database non si fa mai riferimento alla singola riga della tabella, ma solo alla somma di esse, e a causa di questo non abbiamo bisogno di un id univoco per ogni riga

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 tramite una chiave esterna alla tabella originale, nel nostro caso lo facciamo tramite una tabella intermedia che va ad associare un ID dell'attributo con l'ID del giocatore, o della squadra, nel caso dei trofei di squadra.

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

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 l'unica entità figlio presente è la tabella CoachOrManager la quale non aggiunge nuove informazioni se non un semplice flag booleano che va a determinare se la persona presente nella tabella è un coach oppure un manager, per questo motivo andiamo ad unificare questa tabella con la classe padre, la tabella player, aggiungendo i flag is_coach e is_manager

2.2 Dizionari

Dizionario delle entità

Entità	Descrizione	Attributi
administrator	Rappresenta gli amministratori del sistema che hanno privilegi di gestione e controllo.	admin_name (String, PK) admin_surname (String) admin_username (String) 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) player_password (String) is_manager (Boolean) is_coach (Boolean) total_scored_goal (Integer) goal_conceded (Integer) matches_played (Integer) player_id (Integer, PK)
Team	Rappresenta le squadre con il nome, la nazionalità e i dettagli di allenatore e manager attuali.	team_name (String) nationality_id (Integer, FK) current_manager (Integer, FK) current_coach (integer,FK) 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)
characteristic	Definisce le caratteristiche che un giocatore può avere	characteristic_name (String) characteristic_id (Integer, PK)
rolelist	Rappresentazione I ruoli che un giocatore può ricoprire in campo	role_name (String) role_id (Integer, PK)
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.	player_id (Integer, FK) team_id (Integer, FK) 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	home_team_id (Integer, FK) guest_team_id (Integer, FK) home_goal_scored(Integer) guest_goal_scored (Integer) 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	team_id (Integer, FK) scored_goals (Integer) role_id (Integer, FK)

	statistiche personali inerenti alla partita giocata.	player_id (Integer, FK) match_id (Integer, FK)

Dizionario delle associazioni

Associazione	Tipo di relazione	Descrizione
Participates	Molti a uno	Rappresenta l'associazione tra i giocatori e la loro attuale squadra
has characteristic	Molti a molti	Associa le caratteristiche ai giocatori
has role	Molti a molti	Associa i ruoli ai giocatori
player wins	Molti a molti	Associa i trofei individuali ai giocatori
team wins	Molti a molti	Associa I trofei di squadra alle squadre
player plays	Molti a molti	Associa i giocatori alle partite da loro giocate
team plays	Molti a molti	Associa le squadre alle partite da loro giocate

3 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');
```

3.1 Tabelle

-- Creazione della tabella amministratore

```
CREATE TABLE administrator
(
admin_name     varchar(255) not null,
admin_surname  varchar(255) not null,
admin_username varchar(255) not null,
unique primary key,
admin_password varchar(255) 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
(
career_id SERIAL UNIQUE PRIMARY KEY,
player_id INTEGER REFERENCES PLAYER,
team_id INTEGER REFERENCES TEAM,
start_date DATE NOT NULL,
end_date DATE CONSTRAINT end_date_check CHECK (end_date <= CURRENT_DATE),
match_played INTEGER,
goal_scored INTEGER,
trophies_won INTEGER,
goal_conceded INTEGER
);
```

-- 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 characteristic_player
(
association_id SERIAL UNIQUE PRIMARY KEY,
player_id INTEGER REFERENCES PLAYER,
characteristic_id INTEGER REFERENCES CHARACTERISTIC,
CONSTRAINT unique_player_characteristic,
UNIQUE(player_id, characteristic_id)
);
```

-- Creazione della tabella match

Il constraint previene l'inserimento di valori negativi alle colonne dei goal

```
CREATE TABLE match
(
match_id SERIAL UNIQUE PRIMARY KEY,
home_team_id INTEGER NOT NULL REFERENCES team,
guest_team_id INTEGER NOT NULL REFERENCES team,
home_goals_scored INTEGER CONSTRAINT match_home_goals_scored_check CHECK (home_goals_scored >= 0),
guest_goals_scored INTEGER CONSTRAINT match_guest_goals_scored_check CHECK (guest_goals_scored >= 0),
date_match DATE NOT NULL
);
```

-- Creazione della tabella nazionalità

```
CREATE TABLE nationality
(
nationality_id SERIAL UNIQUE PRIMARY KEY,
nationality_name VARCHAR(255) NOT NULL
);
```

-- Creazione della tabella giocatore

```
CREATE TABLE player
(
player_id SERIAL UNQUE PRIMARY KEY,
player_username VARCHAR(255) NOT NULL UNIQUE,
player_password VARCHAR(255) NOT NULL,
player_name VARCHAR(255) NOT NULL,
player_surname VARCHAR(255) NOT NULL,
birth_date DATE NOT NULL,
retirement_date DATE,
foot  FOOT,
iscoach BOOLEAN DEFAULT FALSE,
ismanager BOOLEAN DEFAULT FALSE,
totalscoredgoal INTEGER,
goals_conceded INTEGER,
trophies_won INTEGER,
matches_played INTEGER
);
```

-- Creazione della tabella statistiche dei match dei giocatori

```
CREATE TABLE player_match_stats
(
player_id INTEGER NOT NULL REFERENCES player,
team_id INTEGER REFERENCES team,
match_id INTEGER NOT NULL REFERENCES match,
scored_goals INTEGER,
role_id INTEGER REFERENCES rolelist,
PRIMARY KEY (player_id, match_id)
);
```

-- 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
(
association_id SERIAL UNIQUE PRIMARY KEY,
player_id INTEGER REFERENCES player,
role_id INTEGER REFERENCES rolelist,
CONSTRAINT unique_player_rolelist,
UNIQUE (player_id, role_id)
);
```

-- Creazione della tabella trofei giocatore

```
CREATE TABLE player_trophy
(
association_id SERIAL UNIQUE PRIMARY KEY,
player_id INTEGER REFERENCES player,
trophy_id INTEGER REFERENCES trophy,
trophy_year DATE
);
```

-- 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_id SERIAL UNIQUE PRIMARY KEY,
team_name VARCHAR(255) NOT NULL,
nationality INTEGER DEFAULT 2 NOT NULL REFERENCES nationality,
current_manager INTEGER CONSTRAINT fk_team_current_manager REFERENCES player,
current_coach INTEGER CONSTRAINT fk_team_current_coach
REFERENCES player
);
```

-- Creazione della tabella trofei di squadra

```
CREATE TABLE team_trophy
(
association_id SERIAL UNIQUE PRIMARY KEY,
trophy_id INTEGER REFERENCES trophy,
team_id INTEGER REFERENCES team,
trophy_date DATE
);
```

-- Creazione della tabella trofei

```
create table trophy (
trophy_id SERIAL UNIQUE PRIMARY KEY,
trophy_name VARCHAR(255) NOT NULL,
individual BOOLEAN
);
```

3.2 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_player_goals_conceded();*

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

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

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

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

```
CREATE TRIGGER update_totalscoredgoal_trigger
AFTER INSERT OR UPDATE OF goal_scored ON career
FOR EACH ROW EXECUTE PROCEDURE update_totalscoredgoal();
```

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_goals_conceded();*

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

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_goals_scored*();

```
CREATE TRIGGER update_goals_trigger
AFTER INSERT OR UPDATE player_match_stats
FOR EACH ROW EXECUTE PROCEDURE update_goals_scored();
```

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 player_trophy che attiva *update_player_trophies*();

```
CREATE TRIGGER player_trophy_trigger
AFTER INSERT OR UPDATE ON player_trophy
FOR EACH ROW EXECUTED update_player_trophies();
```

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();
```

3.3 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_id = NEW.current_manager AND
          (player.retirement_date IS NOT NULL AND
              (player.iscoach = TRUE OR player.ismanager = TRUE)
              )
           )
          )
  THEN
       RAISE EXCEPTION 'current_manager must be retired, a coach, or a manager';
END IF;

IF (NEW.current_coach IS NOT NULL AND NOT EXISTS
         (SELECT 1 FROM player WHERE player.player_id = NEW.current_coach AND
             (player.retirement_date IS NOT NULL AND
                 (player.iscoach = TRUE OR player.ismanager = TRUE)
                  )
                 )
                )
THEN
       RAISE EXCEPTION 'current_coach must be retired, a coach, or a manager';
ENDIF;
     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  $$
 BEGIN
   BEGIN
      IF NEW.start_date >= (SELECT retirement_date FROM player WHERE player_id =
NEW.player_id) THEN
      RAISE EXCEPTION 'start_date cannot be later or equal to the retirement_date of the same
player;    END IF;
 RETURN NEW;
   END;
END;
$$;
```

– 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  $$
 BEGIN
    IF OLD.retirement_date IS NULL AND NEW.retirement_date IS NOT NULL THEN-- Check if
retirement status changed
      UPDATE player
      SET foot = NULL, totalscoredgoal = NULL, goals_conceded = NULL
      WHEREplayer_id = NEW.player_id;
   END IF;
   RETURN NEW;
 END;
 $$;
```

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

```
 create function update_career_goals() returns trigger
language plpgsql  as  $$
 BEGIN
    UPDATE career
   SET goal_scored = (
       SELECT SUM(scored_goals)
      FROM player_match_stats pms
      JOIN match m ON pms.match_id = m.match_id
      WHERE pms.player_id = NEW.player_id
        AND m.date_match BETWEEN career.start_date AND COALESCE(career.end_date,
  CURRENT_DATE)
          AND career.team_id = NEW.team_id
     )
      WHERE player_id = NEW.player_id
AND team_id = NEW.team_id;

     RETURNNEW;
   END;
$$;
```

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

```
 create function update_goals_conceded() returns trigger
language plpgsql  as  $$
 BEGIN
    UPDATE career
    SET goal_conceded = goal_conceded + (
       SELECT SUM(scored_goals)
       FROM player_match_stats pms
       JOIN match m ON pms.match_id = m.match_id
       WHERE pms.team_id = NEW.team_id
       AND pms.match_id = NEW.match_id
```

```
        AND m.date_match BETWEEN career.start_date AND COALESCE(career.end_date,
CURRENT_DATE)
      )
     WHERE player_id IN (
       SELECT player_id
       FROM player_match_stats
       WHERE match_id = NEW.match_id
        AND role_id = 1
        AND team_id <> NEW.team_id
   );
    RETURN NEW;
END;
$$;
```

– La funzione aggiorna la quantità di goal fatti da ogni squadra durante una determinata partita

```
create function update_goals_scored() returns trigger
language plpgsql  as  $$
BEGIN
    UPDATE match
    SET home_goals_scored = (
        SELECT SUM(scored_goals)
        FROM player_match_stats
        WHERE match_id = NEW.match_id
        AND team_id = NEW.team_id
      )
     WHERE match_id = NEW.match_id AND home_team_id = NEW.team_id;
     UPDATE match
     SET guest_goals_scored = (
         SELECT SUM(scored_goals)
         FROM player_match_stats
         WHERE match_id = NEW.match_id
         AND team_id = NEW.team_id
       )
      WHERE match_id = NEW.match_id AND guest_team_id = NEW.team_id;

       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_id = NEW.player_id
       AND match.date_match BETWEEN career.start_date AND COALESCE(career.end_date,
CURRENT_DATE)
```

```
      )
     WHERE career.player_id = NEW.player_id;
    RETURN NULL;
  END; $$;
```

– La funzione calcola la quantita' di goal che un giocatore ha fatto passare nella porta se occupava il ruolo del portiere

```
 create function update_player_goals_conceded() returns trigger
language plpgsql  as  $$
 BEGIN
   UPDATE player
   SET goals_conceded = COALESCE((
       SELECT SUM(c.goal_conceded)
       FROM career c
       WHERE c.player_id = player.player_id
    ), 0);
    RETURN NULL;
  END;
$$;
```

– Calcola la quantità di trofei vinti da parte del giocatore durante la sua intera carriera

```
 create function update_player_total_trophies() returns trigger
language plpgsql  as  $$
 BEGIN
   UPDATE player
   SET trophies_won = (
      SELECT SUM(trophies_won)
      FROM career
      WHERE career.player_id = NEW.player_id
    )
    WHERE player_id = NEW.player_id;

    RETURN NEW;
  END;
$$;
```

– Calcola la quantità di trofei vinti da parte di un giocatore durante un periodo di militanza

```
 create function update_player_trophies() returns trigger
language plpgsql  as  $$
 BEGIN
   UPDATE career
     SET trophies_won = (
       SELECT COUNT(*)
       FROM player_trophy
       WHERE player_trophy.player_id = NEW.player_id
       AND player_trophy.trophy_year BETWEEN career.start_date AND
  COALESCE(career.end_date, CURRENT_DATE)
    )
    WHERE player_id = NEW.player_id;

    RETURN NEW;
 END;
 $$;
```

– Calcola la quantità totale di goal fatti da parte del giocatore nel corso della sua carriera

```
create function update_totalscoredgoal() returns trigger
      language plpgsql
as  $$
 BEGIN
    IF (SELECT retirement_date FROM player p WHERE p.player_id = NEW.player_id )IS NULL
THEN
    UPDATE player
    SET totalscoredgoal = (
       SELECT SUM(goal_scored)
       FROM career
       WHERE player_id = NEW.player_id
      )
     WHERE player_id = NEW.player_id;
     ENDIF;
     RETURN NEW;
  END;
 $$;
```