

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. Modello concettuale
   1. Analisi dei requisiti
   2. Class diagram
2. Ristrutturazione
   1. Fase di Ristrutturazione

2.1.1 Analisi delle ridondanze

2.1.2 Analisi degli identificativi

2.1.3 Rimozione di attributi multivalore

2.1.4 Rimozione di attributi composti

2.1.5 Partizione/Accorpamento delle associazioni

2.1.6 Rimozione delle gerarchi

* 1. Dizionari

1. Definizioni SQL
   1. Tabelle
   2. Trigger
   3. Funzioni

1. Modello Concettuale
   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. Class diagram

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

A screenshot of a computer

Description automatically generated

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, infatti, 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.

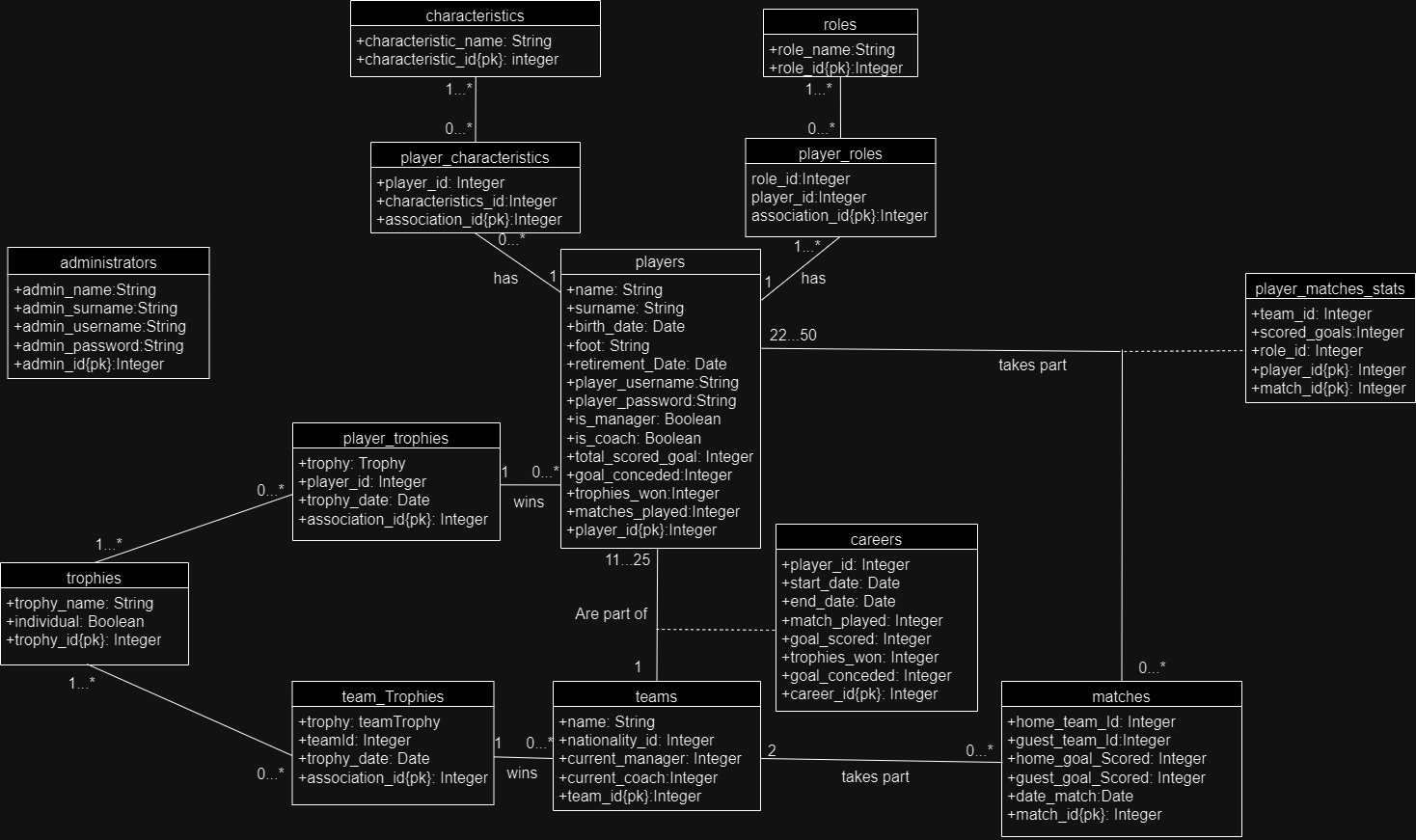
La squadra è un attributo appartenente alla militanza. Nella classe suadra si terrà conto della nazionalità e del nome della stessa. il giocatore può militare in una sola squadra per volta.

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.

1. Ristrutturazione

2.1 Fase di ristrutturazione

In questa fase andremo a cambiare alcuni aspetti del diagramma concettuale con il fine di renderlo più adatto per una traduzione al livello logico.



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

Gli identificativi sono stati razionalizzati nella versione ristrutturata:

- Chiavi Primarie (PK): Ogni entità ora ha un identificativo chiaro e univoco. Ad esempio, player\_id in players, team\_id in teams e match\_id in matches.

- Chiavi Esterne (FK): L'uso delle chiavi esterne è chiaramente indicato per rappresentare le associazioni tra entità, migliorando l'integrità e la coerenza dei dati. Ad esempio, team\_id e player\_id sono utilizzati come chiavi esterne in careers per collegare i giocatori e le squadre alle loro rispettive carriere.

- Chiavi Composte: Le chiavi primarie composte sono utilizzate nelle entità associative come player\_roles e player\_matches\_stats per identificare univocamente i record combinando più campi.

2.1.3 Rimozione di 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 di Attributi Composti

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

2.1.5 Analisi di 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 trophies, 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 Gerarchie

- 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 classi

|  |  |  |
| --- | --- | --- |
| Entità | Descrizione | Attributi |
| administrators | Rappresenta gli amministratori del sistema che hanno privilegi di gestione e controllo. | admin\_name (String), admin\_surname (String), admin\_username (String), admin\_password (String), admin\_id (Integer, PK) |
| players | Rappresenta i giocatori con informazioni personali, statistiche di carriera e dati di autenticazione. | name (String)  surname (String)  birth\_date (Date)  foot (String)  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) |
| Teams | Rappresenta le squadre con il nome, la nazionalità e i dettagli di allenatore e manager attuali. | name (String)  nationality\_id (Integer) current\_manager (String) current\_coach (String) team\_id (Integer, PK) |
| trophies | 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) |
| characteristics | Definisce le caratteristiche che un giocatore può avere | characteristic\_name (String), characteristic\_id (Integer, PK) |
| Roles | Rappresentazione I ruoli che un giocatore può ricoprire in campo | role\_name (String), role\_id (Integer, PK) |
| careers | Rappresenta la carriera di un giocatore in una squadra con statistiche temporali | 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) |
| matches | 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\_matches\_stats | Raccoglie le statistiche delle prestazioni di un giocatore in ogni partita | team\_id (Integer, FK), scored\_goals (Integer), role\_id (Integer, FK), player\_id (Integer, FK), match\_id (Integer, FK) |

Dizionario delle associazione

|  |  |  |
| --- | --- | --- |
| Associazione | Descrizione | Attributi |
| player\_characteristics | Rappresenta l'associazione tra i giocatori e le loro caratteristiche. | player\_id (Integer, FK), characteristics\_id (Integer, FK), association\_id (Integer, PK) |
| player\_roles | Definisce I ruoli che i giocatori possono avere collegando i giocatori ai ruoli | role\_id (Integer, FK), player\_id (Integer, FK), association\_id (Integer, PK) |
| player\_trophies | Associa I trofei vinti ai giocatori | trophy (String, FK), player\_id (Integer, FK), trophy\_date (Date), association\_id (Integer, PK) | |
| Team\_trophy | Associa I trofei vinti alle squadre | trophy (String, FK), team\_id (Integer, FK), trophy\_date (Date), association\_id (Integer, PK) |
| careers | Rappresenta l'associazione tra un giocatore e una squadra per un certo periodo di tempo, registrando le statistiche durante quella carriera. | 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) | |
| Player\_matches\_stats | Associa le statistiche di un giocatore durante una partita specifica. | team\_id (Integer, FK), scored\_goals (Integer), role\_id (Integer, FK), player\_id (Integer, FK), match\_id (Integer, FK) |

Definizioni SQL

3.1 Tabelle

-- Creazione della tabella amministratore

create table administrator

(

admin\_id serial

primary key,

admin\_name varchar(255) not null,

admin\_surname varchar(255) not null,

admin\_username varchar(255) not null

unique,

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

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

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

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

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

primary key,

nationality\_name varchar(255) not null

);

-- Creazione della tabella giocatore

create table player

(

player\_id serial

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 varchar(50),

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

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

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

primary key,

role\_name varchar(255) not null

);

-- Creazione della tabella delle squadre

create table team

(

team\_id serial

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

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

primary key,

trophy\_name varchar(255) not null,

individual boolean

);

* 1. 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  
 on 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  
execute procedure *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*();

* 1. 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;

$$;