**Progetto di Basi di dati**

**2018-19**

**SOCCERDB**

**Gregorio Ghidoli, 908960**

**INDICE**

**PROGETTAZIONE CONCETTUALE 1**

**RISTRUTTURAZIONE E SCHEMA FINALE 5**

**PROGETTAZIONE LOGICA E ANALISI DELLE DIPENDENZE 6**

**ASPETTI METODOLOGICI E FUNZIONI REALIZZATE 7**

**ALTRI ASPETTI METODOLOGICI 12**

**PRODOTTI SOFTWARE E LINGUAGGI UTILIZZATI 12**

**PROGETTAZIONE CONCETTUALE**

***CONCETTI ANALIZZATI:*** *LEAGUE, SEASON, STAGE, MATCH*

Il concetto di COUNTRY è modellato come attributo di LEAGUE perchè non sono presenti altre informazioni oltre al nome.

Se si vuole rendere disponibile la possibilità di aggiungere altre informazioni relative al COUNTRY, è conveniente modellare tale concetto come un’entità.

Con questa soluzione si assume che non possono esserci occorrenze di LEAGUE con lo stesso nome in COUNTRY diversi.

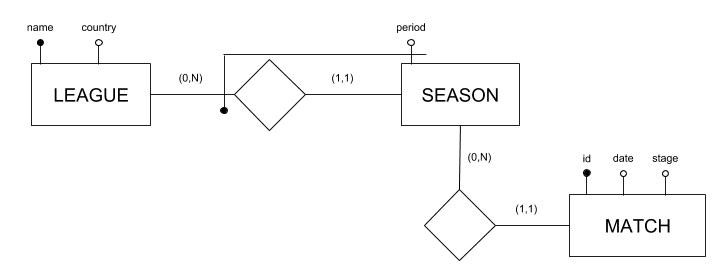
Il concetto di SEASON potrebbe essere modellato in due modi:

* come entità debole di LEAGUE
* come attributo di MATCH

***Soluzione 1***

SEASON viene modellata come entità debole di LEAGUE, in quanto una season è identificata sia dal periodo che della lega a cui appartiene (es: “La stagione 2017/2018 della Belgium Jupiler League”).

In questo modo si ha direttamente l’elenco di tutte le season e si possono inserire SEASON senza che siano stati giocati ancora dei MATCH.

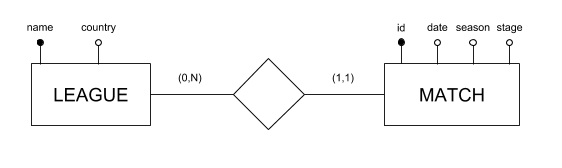


***Soluzione 2***

SEASON viene modellata come attributo di MATCH, in quanto l’unico dato disponibile sulla SEASON è il periodo identificativo e un MATCH può essere giocato solo in una season.

Con questa soluzione non si dispone direttamente dell’elenco di tutte le season ma deve essere ricavato.

Si ha il vantaggio di dover gestire una entità in meno.



Il concetto di STAGE, come quello di season, è modellato come attributo di MATCH, sia perchè viene rappresentato solo da un numero e sia perchè un MATCH non può essere giocato in più stage(giornate).

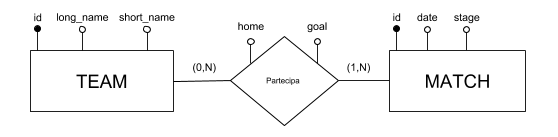
Se venisse modellato come entità si avrebbe il vantaggio di avere direttamente l’elenco di tutti gli stage al costo di dover gestire una entità in più.

***CONCETTI ANALIZZATI:*** *TEAM, MATCH*

La relazione che sussiste tra TEAM e MATCH può essere modellata in due modi.

1. un team partecipa ad un match
2. un “home team” sfida un “away team” in un match

***soluzione 1***



In questa soluzione si ha un’associazione molti a molti tra TEAM e MATCH.

L’attributo “home” è un attributo booleano che indica se il TEAM, relazionato con il MATCH, gioca in casa o meno.

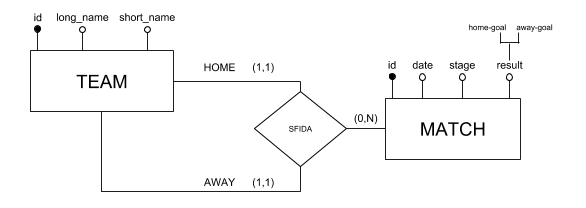
L’informazione del numero di goal effettuati è rappresentata come come attributo dell’associazione.

“Un team partecipa ad un match, in casa o meno, effettuando n goal”.

Con questa soluzione si deve controllare, tramite opportuni vincoli, che ad ogni occorrenza di MATCH corrispondano esattamente due occorrenze di TEAM.

Inoltre, l’informazione delle squadre partecipanti al match, non è direttamente accessibile ma andrà ricostruita.

***soluzione 2***



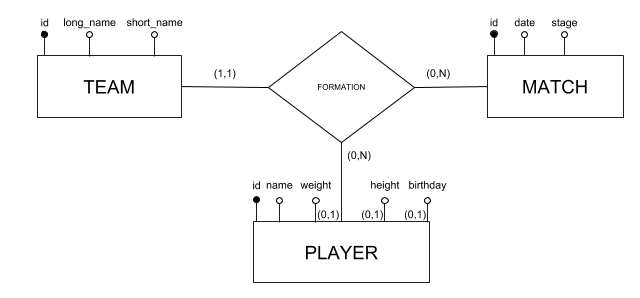
In questa soluzione, il concetto di un team (home) che sfida un altro team (away) in un match, è modellato con un’associazione ricorsiva (TEAM) e ternaria.

Le informazioni sul risultato della partita sono proposte come attributo composto di match.

Il vantaggio di questa soluzione rispetto alla ***soluzione 1*** è che, dato un match, è immediato recuperare sia il risultato della partita che le squadre coinvolte nel match e, non necessita di trigger per controllare le occorrenze.

***CONCETTI ANALIZZATI:*** *TEAM, MATCH, PLAYER*

La formazione con cui una squadra entra in campo si può modellare come un’ associazione ternaria.



Nel file csv a disposizione, per ciascun match, non sempre vengono registrati tutti i giocatori di un team, quindi la cardinalità minima di PLAYER è 0.

Nel file CSV fornito in prima istanza, ogni volta che un giocatore partecipa ad un MATCH, ne vengono forniti tutti i dati.

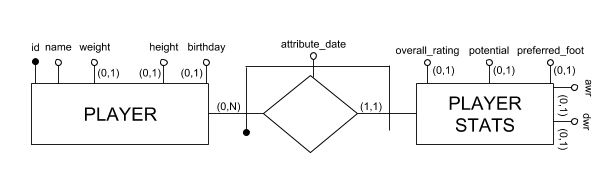
Ipotizzando che in successivi CSV, gli attributi *weight, height* e *birthday*  possano non

venire sempre registrati, sono stati resi non obbligatori.

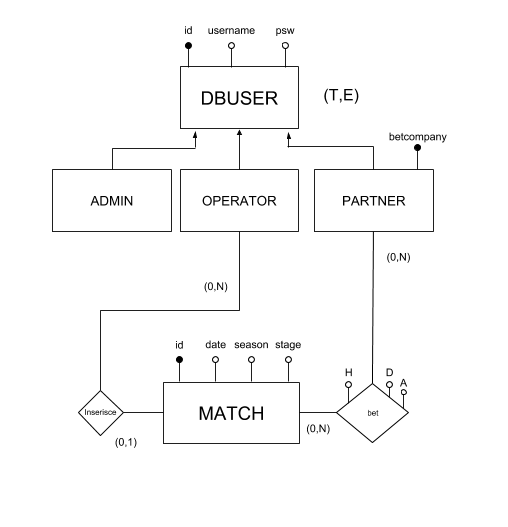
***CONCETTI ANALIZZATI:*** *PLAYER, PLAYER STATS*

Per permettere di memorizzare le caratteristiche dei giocatori nel tempo, queste sono state modellate come una serie storica (*PLAYER STATS*).

Tutti gli attributi dell’entità STATS non sono obbligatori (nello schema non sono stati rappresentati tutti i numerosi attributi presenti nel file CSV).



***CONCETTI ANALIZZATI:*** *MATCH, BET COMPANY, BET, USER, PARTNER, OPERATOR*

******

Le categorie di utenti sono tre: administrator, operator e partner.

Gli utenti (administrator, operator, partner) della base di dati sono stati modellati con una gerarchia di generalizzazione.

La gerarchia è TOTALE ED ESCLUSIVA in quanto un utente non può ricoprire simultaneamente più ruoli.

Gli operatori possono inserire, modificare e cancellare i match che loro stessi hanno inserito.

L’appartenenza di un partner ad una società di scommesse può essere modellata come entità ma, essendoci solo a disposizione l’informazione del nome della società, è preferibile rappresentarla come attributo obbligatorio di PARTNER.

L’associazione BET permette al partner di inserire scommesse relative ad un match.

Gli attributi H,A,D di BET rappresentano le singole quote della scommessa.

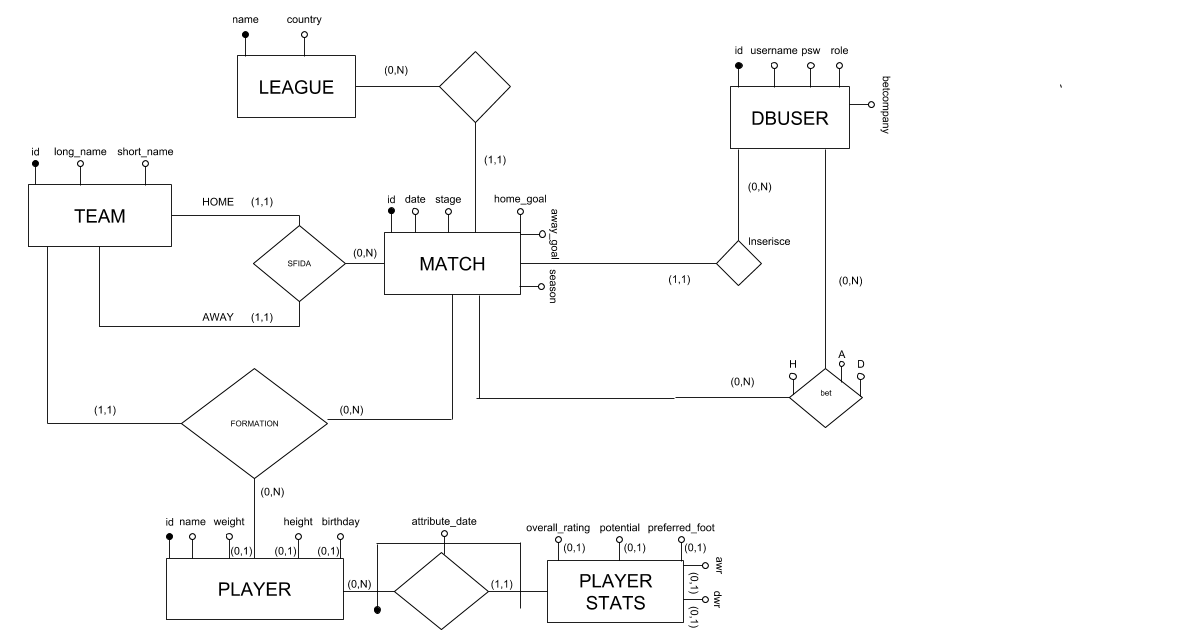
**RISTRUTTURAZIONE E SCHEMA FINALE**

Per modellare il concetto di season è stata scelta la **soluzione 2** perchè da specifiche non viene mai chiesto di visualizzare l’elenco di tutte le stagioni e quindi è meglio avere la possibilità di gestire un’entità in meno.

Per modellare la relazione tra team e match è stata scelta la ***soluzione 2*** perchè è molto utile per l’applicazione poter avere direttamente le informazioni dei team partecipanti ad un match e i loro risultati.

Nella ristrutturazione della gerarchia di generalizzazione di DBUSER sono state eliminate le entità figlie apportando le seguenti modifiche alle cardinalità e alle associazioni:

* si è tenuta una sola associazione *INSERISCE*
* è stato aggiunto l’attributo obbligatorio *role* a DBUSER
* l’attributo betcompany è stato reso non obbligatorio.



**PROGETTAZIONE LOGICA E ANALISI DELLE DIPENDENZE**

**league**(*name*, country)

*Dipendenze:* name→country

**match**(*id*,date,stage, season, home, away, home\_goal,away\_goal, league,dbuser)

[league è chiave esterna obbligatoria di league]

[home è chiave esterna obbligatoria di team]

[away è chiave esterna obbligatoria di team]

[dbuser è chiave esterna obbligatoria di team]

*Dipendenze:*

id→date,stage,season,home,away,home\_goal,away\_goal,league

*Osservazione: l’associazione* sfida *dava origine ad una relazione* ***sfida****(match, home, away) con gli attributi home,away dipendenti dall’id del match, per questo motivo sono stati uniti alla relazione* ***match****.*

**team**(*id*,long\_name,short\_name)

*Dipendenze:* id→long\_name,short\_name

**player**(*id*,name,weight\*,height\*,birthday\*)

*Dipendenze:* id→name,weight,height,birthday

**formation**(*match,player*,team)

[match è chiave esterna non nullabile di match]

[player è chiave esterna non nullabile di player]

[team è chiave esterna non nullabile di team]

**playerstats** (*player, attribute\_date*, overall\_rating\*, potential\*, preferred\_foot\*, attacking\_work\_rate\*, defensive\_work\_rate\*, crossing\*, finishing\*, heading\_accuracy\*, short\_passing\*, volleys\*, dribbling\*, curve\*, free\_kick\_accuracy\*, long\_passing\*, ball\_control\*, acceleration\*, sprint\_speed\*, agility\*, reactions\*, balance\*, shot\_power\*, jumping\*, stamina\*, strength\*, long\_shots\*, aggression\*, interceptions\*, positioning\*, vision\*, penalties\*, marking\*, standing\_tackle\*, sliding\_tackle\*, gk\_diving\*, gk\_handling\*, gk\_kicking\*, gk\_positioning\*, gk\_reflexes\*)

[player è chiave esterna non nullabile di player]

*Dipendenze:* tutti gli attributi dipendono dalla chiave primaria (player, attribute\_date)

**bet**(*match, user*, h, d, a)

[match è chiave esterna non nullabile di match]

[user è chiave esterna non nullabile di user]

*Dipendenze:* match,user →h,d,a

Lo schema è in ***2FN*** perchè non ci sono attributi che dipendono solo parzialmente dalla chiave della relazione.

Lo schema è in ***3FN*** perchè non sussistono dipendenze transitive tra gli attributi.

**ASPETTI METODOLOGICI E FUNZIONI REALIZZATE**

Ad ogni tabella corrispondono tre funzioni: una per l’inserimento, una per la cancellazione e una per l’aggiornamento.

Per garantire la verifica dei permessi, ad ogni funzione viene passato come argomento l’identificativo dell’utente che compie l’operazione.

Se l’operazione riguarda i singoli match (***MATCH***) o le formazioni (***FORMATION***), l’utente deve essere un operatore (***role = ‘operator’***).

Per le operazioni di cancellazione e aggiornamento viene verificato pure che l’operatore che sta compiendo l’operazione sia anche l’utente che ha inserito tali tuple.

Se l’operazione riguarda le scommesse (***BET***), l’utente deve essere un partner (***role = ‘partner’***).

Per l’operazione di inserimento si controlla che nessun altro partner della stessa società di scommesse del partner che sta eseguendo l’operazione abbia già inserito una scommessa per il match in questione.

Per le operazioni di cancellazione e aggiornamento viene verificato che il partner che sta compiendo l’operazione sia anche il partner che ha inserito tali tuple.

Per le restanti tabelle, solo gli utenti amministratore (***role = ‘administrator’)*** possono effettuare operazioni di inserimento, aggiornamento e cancellazione.

**ELENCO FUNZIONI**

**INSERIMENTO**

Le funzioni di inserimento restituiscono un codice che indica se l’operazione è andata a buon fine oppure la causa che non ha permesso di completare l’inserimento.

***insert\_league*** (id\_user INTEGER, new\_name VARCHAR, country VARCHAR)

valori di ritorno:

1 se l'inserimento va a buon fine

-10 se non vengono specificati tutti i parametri

-20 se utente (id\_user) non è presente a sistema

-30 se l'utente (id\_user) non è autorizzato (solo amministratore)

-40 se lega (new\_name) è già presente a sistema

***insert\_team*** (id\_user INTEGER, id\_team INTEGER, long\_name VARCHAR, short\_name VARCHAR(10))

valori di ritorno:

1 se team viene inserito correttamente

-10 se non vengono specificati tutti i parametri

-20 se utente non è presente a sistema

-30 se utente non è autorizzato

-40 se team è già presente a sistema

***insert\_player*** (id\_user INTEGER, id\_player INTEGER, name VARCHAR, birthday DATE, weight INTEGER, height FLOAT)

valori di ritorno:

1 se l'inserimento va a buon fine

-10 se non vengono specificati tutti i parametri

-20 se utente (id\_user) non è presente a sistema

-30 se utente (id\_user) non è autorizzato (solo amministratore)

-40 se player (id\_player) già presente a sistema

-50 se i parametri non sono validi (weight e height devono essere >=0)

***insert\_playerstats*** (id\_user INTEGER, new\_player INTEGER,new\_attribute\_date DATE,overall\_rating INTEGER,potential INTEGER,preferred\_foot VARCHAR(10),

attacking\_work\_rate VARCHAR(10),defensive\_work\_rate VARCHAR(10),crossing INTEGER,finishing INTEGER,heading\_accuracy INTEGER,short\_passing INTEGER,

volleys INTEGER,dribbling INTEGER,curve INTEGER,free\_kick\_accuracy INTEGER,long\_passing INTEGER,ball\_control INTEGER,acceleration INTEGER,sprint\_speed INTEGER,

agility INTEGER,reactions INTEGER,balance INTEGER,shot\_power INTEGER,jumping INTEGER,stamina INTEGER,strength INTEGER,long\_shots INTEGER, aggression INTEGER, interceptions INTEGER,

positioning INTEGER,vision INTEGER,penalties INTEGER,marking INTEGER,standing\_tackle INTEGER,sliding\_tackle INTEGER,gk\_diving INTEGER,gk\_handling INTEGER,gk\_kicking INTEGER,

gk\_positioning INTEGER,gk\_reflexes INTEGER)

valori di ritorno:

1 se l'inserimento va a buon fine

-10 se non vengono specificati i parametri obbligatori (id\_user, new\_player, new\_attribute\_date)

-20 se utente (id\_user) non presente a sistema

-30 se utente non è autorizzato (solo amministratore)

-40 se player (new\_player) non è presente a sistema

-50 se la statistica è già presente a sistema

***insert\_match*** (id\_user INTEGER, new\_id INTEGER, new\_date DATE, new\_stage INTEGER, new\_season VARCHAR, new\_home INTEGER, new\_away INTEGER, new\_home\_goal INTEGER, new\_away\_goal INTEGER, new\_league VARCHAR, include\_id INTEGER)

valori di ritorno:

>=0 se match viene inserito correttamente (corrisponde ad id match inserito)

-10 se non vengono specificati tutti i parametri

-20 se utente (id\_user) non presente a sistema

-30 se utente non è autorizzato ad inserire match (solo operator)

-40 se match (new\_id) è già presente a sistema

-50 se viene riconosciuta uguaglianza del match con altri match non utilizzando id

-60 se lega (new\_league) non è presente a sistema

-70 se home team (new\_home) non è presente a sistema

-80 se away team (new\_away) non è presente a sistema

-90 se parametri non validi (home\_goal e away\_goal devono essere positivi e new\_home e new\_away devono essere diversi)

Nota 1: la funzione per inserire i match viene usata sia per inserire lo specifico match (non specificando il nuovo id) sia durante il caricamento massivo (dove l’id è già specificato).

Nel primo caso include\_id deve essere 0 e bisogna passare un qualunque new\_id non nullo che non verrà poi considerato durante l’inserimento perchè verrà utilizzato il seriale.

Nel secondo caso include\_id deve essere 1 e new\_id dovrà essere valorizzato correttamente perchè verrà preso in considerazione durante l’inserimento.

Nota 2: per verificare che operatori diversi non inseriscano dati relativi al medesimo match viene fatto un confronto fra tutti i campi del match che si sta inserendo con quelli già a sistema non considerando l’identificativo.

Nota 3: quando l’amministratore carica il file match.csv, dovrà scegliere a quale operatore associare i match.

***insert\_formation*** (id\_user INTEGER, new\_match INTEGER, new\_player INTEGER, new\_team INTEGER)

valori di ritorno:

1 se formation viene inserita correttamente

-10 se non vengono specificati tutti i parametri

-20 se utente (id\_user) non presente a sistema

-30 se utente non è autorizzato (solo operator)

-40 se formazione già presente

-50 se match (new\_match) non presente

-60 se operator non può inserire formation per questo match (solo operator che ha inserito new\_match)

-70 se player (new\_player) non presente

-80 se team (new\_team) non presente

-90 se il team è al già al completo per questo match (11 giocatori)

-100 se il player gioca già in un team per il match dato

Nota: nel file CSV è possibile che ci siano righe in cui uno stesso giocatore gioca è associato in tutte e due le squadre per lo stesso match.

In questo caso il sistema inserisce il giocatore nella squadra di casa.

Questa informazione potrà essere modificata in seguito usando la funzione update\_match.

***insert\_dbuser*** (id\_user INTEGER, new\_username VARCHAR, new\_password VARCHAR, new\_role VARCHAR, new\_betcompany VARCHAR(10))

valori di ritorno:

1 se nuovo utente viene inserito correttamente

-10 se non vengono specificati tutti i parametri

-20 se utente (id\_user) che sta eseguendo l'inserimento non è presente a sistema

-30 se utente che sta eseguendo l'inserimento non è autorizzato (solo amministratore)

-40 se utente (new\_username) che si sta inserendo è già presente a sistema

-50 se ruolo specificato non è valido

-60 se nuovo utente è partner deve essere specificato anche new\_betcompany

***insert\_bet*** (new\_match INTEGER, new\_dbuser INTEGER, new\_h FLOAT, new\_d FLOAT, new\_a FLOAT)

valori di ritorno:

1 se bet viene inserita correttamente

-10 se non vengono specificati tutti i parametri

-20 se new\_h o new\_d o new\_a sono negativi

-30 se utente non presente a sistema

-40 se utente non è abilitato ad inserire bet (solo partner)

-50 se il match (new\_match) non è presente a sistema

-60 se è già stata inserita una bet per questo match da un'altro partner appartenente alla stessa società di scommesse

**UPDATE**

Le funzioni di update sono funzioni senza valore di ritorno.

Nel caso ci siano problemi con l’aggiornamento vengono sollevate delle eccezioni con specifici messaggi di errore.

In tutte le funzioni, per i parametri non obbligatori, se passati come NULL, vengono aggiornati prendendo i valori già presenti a sistema.

***update\_league*** (id\_user INTEGER, leagueToUpdate VARCHAR, new\_name VARCHAR, new\_country VARCHAR)

Parametri obbligatori: id\_user e leagueToUpdate

Permessi: concesso solo ad amministratore.

***update\_team*** (id\_user INTEGER, teamToUpdate INTEGER, new\_id\_team INTEGER, new\_long\_name VARCHAR, new\_short\_name VARCHAR(10))

Parametri obbligatori: id\_user e teamToUpdate

Permessi: concesso solo ad amministratore.

***update\_player*** (id\_user INTEGER, playerToUpdate INTEGER, new\_id\_player INTEGER, new\_name VARCHAR, new\_birthday DATE, new\_weight FLOAT, new\_height FLOAT)

Parametri obbligatori: id\_user e playerToUpdate

Permessi: concesso solo ad amministratore.

***update\_playerstats***(id\_user INTEGER, player INTEGER, attribute\_date DATE,new\_player INTEGER,new\_attribute\_date DATE,new\_overall\_rating INTEGER,new\_potential INTEGER,new\_preferred\_foot VARCHAR(10),new\_attacking\_work\_rate VARCHAR(10),new\_defensive\_work\_rate VARCHAR(10),new\_crossing INTEGER,new\_finishing INTEGER,new\_heading\_accuracy INTEGER,new\_short\_passing INTEGER,new\_volleys INTEGER,new\_dribbling INTEGER,new\_curve INTEGER,new\_free\_kick\_accuracy INTEGER,new\_ball\_control INTEGER,new\_acceleration INTEGER,new\_sprint\_speed INTEGER,new\_agility INTEGER,new\_reactions INTEGER,new\_shot\_power INTEGER,new\_jumping INTEGER,new\_stamina INTEGER,new\_long\_shots INTEGER,new\_aggression INTEGER,new\_interceptions INTEGER,new\_positioning INTEGER,new\_vision INTEGER,new\_penalties INTEGER,new\_marking INTEGER,new\_standing\_tackle INTEGER,new\_sliding\_tackle INTEGER,new\_gk\_diving INTEGER,new\_gk\_handling INTEGER,new\_gk\_kicking INTEGER,new\_gk\_positioning INTEGER,new\_gk\_reflexes INTEGER)

Parametri obbligatori: id\_user,player,attribute\_date

Permessi: concesso solo ad amministratore.

***update\_match*** (id\_user INTEGER, matchToUpdate INTEGER, new\_id\_match INTEGER, new\_date DATE, new\_stage INTEGER,new\_season INTEGER,new\_home INTEGER,new\_away INTEGER,new\_home\_goal INTEGER,new\_away\_goal INTEGER, new\_league VARCHAR,new\_db\_user INTEGER)

Parametri obbligatori: id\_user e matchToUpdate

Permessi: concesso solo ad operatore che ha inserito match (matchToUpdate).

***update\_formation*** (id\_user INTEGER, id\_match INTEGER, id\_player INTEGER, id\_team INTEGER, new\_match INTEGER, new\_player INTEGER, new\_team INTEGER)

Parametri obbligatori: id\_user, id\_match, id\_player, id\_team

Permessi: concesso solo ad operatore che ha inserito match (id\_match).

***update\_user*** (id\_user INTEGER, userToUpdate INTEGER, new\_username VARCHAR, new\_password VARCHAR, new\_role VARCHAR, new\_betcompany VARCHAR(10))

Parametri obbligatori: id\_user e userToUpdate

Permessi: concesso solo ad amministratore o all’utente stesso.

***update\_bet*** (id\_user INTEGER, id\_match INTEGER, new\_h FLOAT, new\_d FLOAT, new\_a FLOAT)

Parametri obbligatori: id\_user e id\_match

Permessi: concesso solo ad amministratore, non si può fare update dell’id.

**CANCELLAZIONE**

Le funzioni di cancellazione sono funzioni senza valore di ritorno.

Nel caso ci siano problemi con l’aggiornamento vengono sollevate delle eccezioni con specifici messaggi di errore.

***delete\_league*** (id\_user INTEGER, leagueToDelete VARCHAR)

Parametri obbligatori: vanno specificati tutti i parametri.

Permessi: concesso solo ad amministratore.

***delete\_team*** (id\_user INTEGER, teamToDelete INTEGER)

Parametri obbligatori: vanno specificati tutti i parametri.

Permessi: concesso solo ad amministratore.

**delete\_player** (id\_user INTEGER, playerToDelete INTEGER)

Parametri obbligatori: vanno specificati tutti i parametri.

Permessi: concesso solo ad amministratore.

***delete\_playerstats*** (id\_user INTEGER, playerToDelete INTEGER, attributeDateToDelete DATE)

Parametri obbligatori: vanno specificati tutti i parametri.

Permessi: concesso solo ad amministratore.

***delete\_match*** (id\_user INTEGER, idMatchToDelete INTEGER)

Parametri obbligatori: vanno specificati tutti i parametri.

Permessi: concesso solo ad operatore che ha inserito il match.

***delete\_formation*** (id\_user INTEGER, matchToDelete INTEGER, playerToDelete INTEGER, teamToDelete INTEGER)

Parametri obbligatori: vanno specificati tutti i parametri.

Permessi: concesso solo ad operatore che ha inserito il match.

**ALTRI ASPETTI METODOLOGICI**

La classifica si può visualizzare con la vista materializzata **ranking.**

Per mantenerla aggiornata è stato aggiunto un trigger alla tabella match che ad ogni inserimento, aggiornamento e cancellazione, effettua il refresh della vista (funzione **functgr\_refresh\_ranking**).

Tutte le chiavi esterne delle tabelle sono state messe ON UPDATE CASCADE ON DELETE CASCADE.

Per visualizzare i giocatori migliori di ogni match è stata utilizzata una vista perché la query non richiede un tempo di esecuzione oneroso.

Se fosse stata usata una vista materializzata si sarebbero dovuti inserire dei trigger in più avrebbero complicato la gestione della base di dati.

**PRODOTTI SOFTWARE E LINGUAGGI UTILIZZATI**

**DBMS:** PostgreSQL 10.6

**Web server:** apache2

**Linguaggio server-side:** PHP 7.1.26