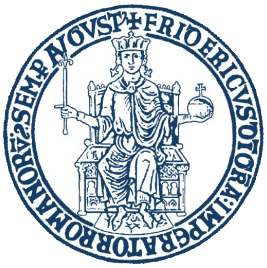


UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Documentazione database

Sistema informativo dei calciatori
Traccia 3

N86004535 Domenico Mautone - N86004521 Mario Russo

18 marzo 2024

Indice

1 Traccia progetto

Si sviluppi un sistema informativo, composto da una base di dati relazionale e da un applicativo Java dotato di GUI (Swing o JavaFX), 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 una 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.

1.1 Analisi traccia

Questo database relazionale si occupa della gestione dei calciatori di tutto il mondo.

Un **amministratore** è caratterizzato da:

- **Username**
- **Password**

Un **calciatore** è caratterizzato da:

- **Nome**
- **Cognome**

- **Piede** (può essere sinistro, destro o ambidestro)
- **Sesso**
- **Ruolo** (può essere multiplo (portiere, difensore, centrocampista e attaccante), perciò è un attributo multivalore)
- **Data_Nascita**
- **Feature** (può essere multiplo (tackle, rovesciata, etc..), quindi multivalore)
- **Nazionalità** (Siccome esistono calciatori con più di una nazionalità l'attributo è multivalore)

In aggiunta, un calciatore può anche ritirarsi, perciò avrà una **data di ritiro (DataRitiro)** che, in caso il calciatore sia ancora in attività esso sarà **NULL di default**, dato che un calciatore che inizia la sua carriera si presume che non si ritiri immediatamente.

Il calciatore avrà per forza una carriera (altrimenti non è considerabile tale). In carriera militerà **in una o più squadre**, o anche in nessuna, per un periodo di tempo. In quest'ultimo caso l'attributo "squadra" avrà valore **svincolato**, ovvero non avrà squadra. Perciò l'associazione **militanza** avrà come attributo:

- **DataInizio**
- **DataFine**
- **PartiteGiocate**
- **GolSegnati**
- **GolSubiti**

Con gli attributi **DataInizio e DataFine** che indicheranno il periodo in cui il giocatore milita o ha militato in una determinata squadra oppure il periodo in cui è considerato svincolato **(DataFine sarà NULL** quando il calciatore si trova ancora in una determinata squadra), mentre **GolSubiti** è un attributo attivo solo se il **il giocatore risulterà avere il ruolo "portiere"**

Nella **squadra** invece militeranno sempre dei calciatori dato che il non averne significherebbe che la squadra non esiste. Quest'entità avrà come attributi:

- **Nome**
- **Nazionalità**
- **AnnoFondazione**
- **Categoria** (ci indica se è una squadra femminile o maschile)

La squadra può **partecipare** in uno o più competizioni in diverse annate, dove in caso di retrocessione si ritroverà in un campionato di serie inferiore rispetto a quello dove si trovava, viceversa in caso di promozione. Quindi l'entità **competizione** avrà come attributi:

- **Nome**
- **Nazionalità**
- **Descrizione**
- **Categoria** (ci indica se è una competizione femminile o maschile)

In questo caso l'attributo **Nazionalità** avrà valore "Europa" quando si tratterà di competizioni che coinvolgono squadre di più campionati europei, "Asia" quando si tratterà di competizioni che coinvolgono squadre di più campionati asiatici, "Africa" quando si tratterà di competizioni che coinvolgono squadre di più campionati africani, "Sudamerica" quando si tratterà di competizioni che coinvolgono squadre di più campionati sudamericani, "Nordamerica" quando si tratterà di competizioni che coinvolgono squadre di più campionati nordamericani e "Oceania" quando si tratterà di competizioni che coinvolgono squadre di più campionati oceanici.

Invece la relazione **partecipa** avrà come attributo:

- **Stagione** (Che avrà il formato yyyy/yyyy)

Un calciatore e una squadra potrebbero vincere dei **trofei**, come no. Perciò avremmo altre due entità ovvero **TrofeoCalciatore** per i calciatori e **TrofeoCompetizione** per la squadra ed entrambe le entità avranno gli attributi:

- **Nome**
- **Descrizione**
- **Categoria** (ci indica se è un trofeo femminile o maschile)

Verranno presi in considerazione solo quei trofei riconosciuti dalle federazioni ufficiali calcistiche mondiali (UEFA, FIFA, CONMEBOL, etc)
VinceCalciatore invece è la relazione che associa al calciatore il trofeo calciatore mentre **VinceSquadra** è la relazione che associa alla squadra il trofeo vinto e avranno entrambe le entità come attributo:

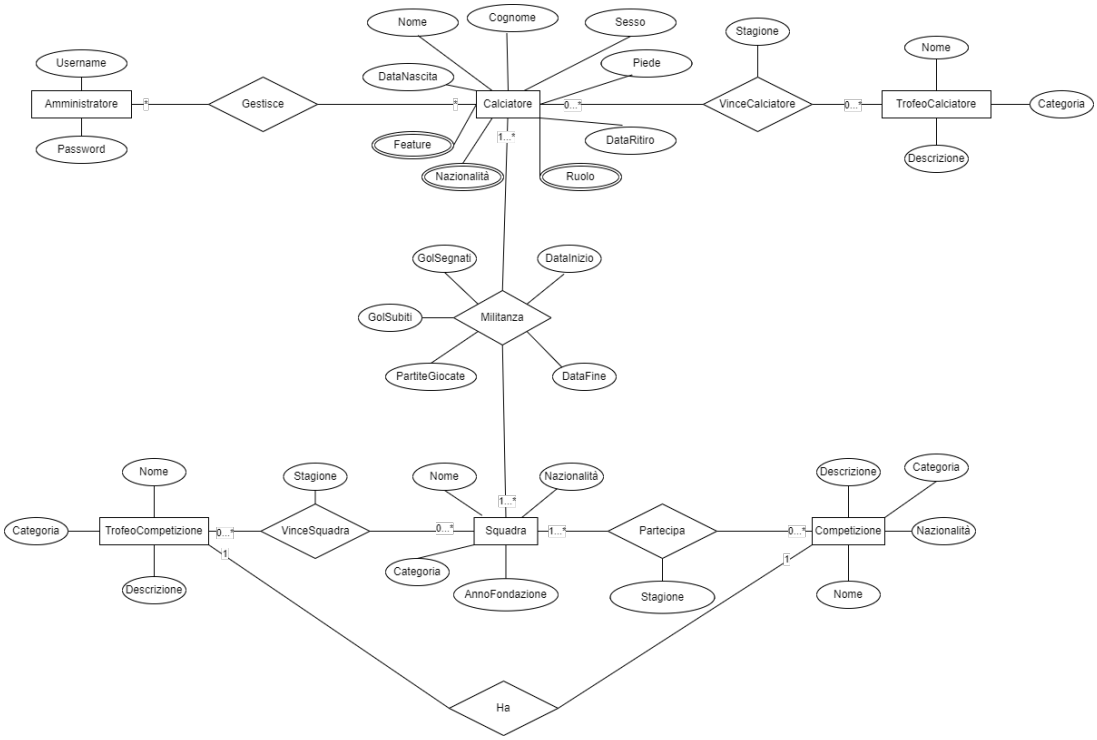
- **Stagione** (Che avrà il formato yyyy/yyyy)

Che può assumere anche una data del tipo 2007/2007 che si riferisce quindi all'intero anno solare

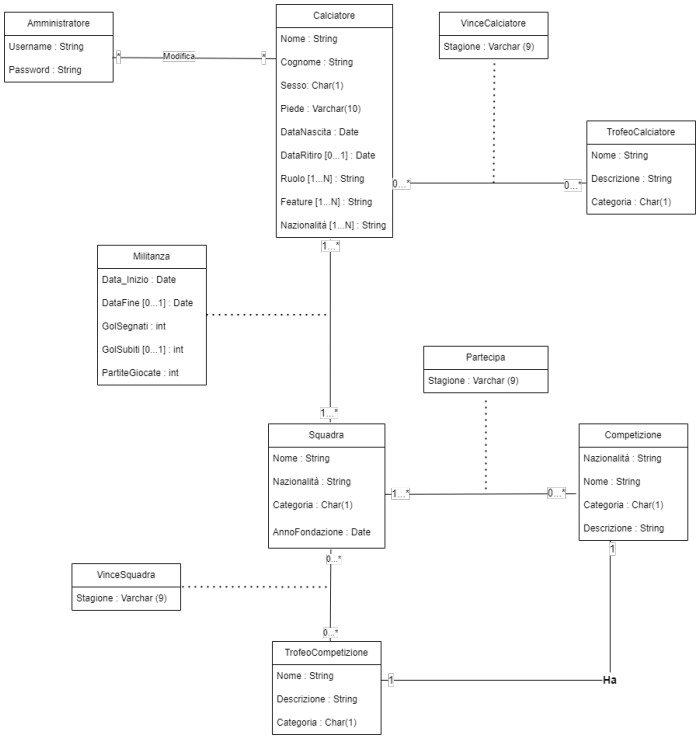
La relazione **ha** assocerà ad ogni competizione un trofeo di competizione.

2 Modello concettuale

2.1 Modello ER



2.2 Modello UML



2.3 Dizionario entità

Entità	Descrizione	Attributi
Calciatore	Rappresenta i calciatori del mondo con tutte le loro caratteristiche	**Nome**(String): Nome del calciatore **Cognome**(String): Cognome del calciatore **Piede**(Varchar(10)): Piede forte del calciatore **Sesso**(Char(1)): Genere del calciatore **DataNascita**(Date): Data di nascita del calciatore **DataRitiro[0...1]**(Date): Data in cui si è ritirato il calciatore (In caso giochi ancora sarà a NULL) **Ruolo[1...N]**(String): Posizione del campo dove il calciatore gioca **Feature[1...N]**(String): Caratteristiche forti del calciatore **Nazionalità[1..N]**(String): Nazionalità del calciatore
Squadra	Rappresenta le squadre di calcio con le loro informazioni	**Nome**(String): Nome della squadra **Nazionalità**(String): Nazionalità della squadra **AnnoFondazione**(Char(4)): Anno in cui è stata fondata la squadra **Categoria**(Char(1)): Categoria della squadra
Competizione	Rappresenta tutte le competizioni esistenti	**Nazionalità**(String): Dove si svolge la competizione **Nome**(String): Nome della competizione **Descrizione**(String): Descrizione della competizione **Categoria**(Char(1)): Categoria della competizione
TrofeoCalciatore	Rappresenta i trofei individuali con le loro informazioni	**Nome**(String): Nome del trofeo **Descrizione**(String): Descrizione del trofeo **Categoria**(Char(1)): Categoria del trofeo
TrofeoCompetizione	Rappresenta i trofei delle competizioni con le loro informazioni	**Nome**(String): Nome del trofeo **Descrizione**(String): Descrizione del trofeo **Categoria**(Char(1)): Categoria del trofeo

Amministratore	Rappresenta i dati di accesso dell'amministratore	**Username**(String): Rappresenza l'username usato dall'amministratore per accedere **Password**(String): Rappresenta la password usata dall'amministratore per accedere

2.4 Dizionario associazioni

Associazione	Descrizione
Militanza	Associazione di tipo **molti-a-molti** tra **Calciatore** e **Squadra**. Questa associazione si riferisce al fatto che un calciatore milita in una o più squadre e in una squadra militano più giocatori. Per questo motivo la relazione avrà come attributi **DataInizio**(Date), **DataFine**[0...1](Date) che compongono il periodo in cui il giocatore ha militato in una determinata squadra, **PartiteGiocate**(Int), **GolSegnati**(Int) e **GolSubiti**[0...1](Int) con quest'ultimo che è riferito unicamente a quei calciatori che tra i ruoli avranno anche quello del "portiere".
Partecipa	Associazione di tipo **molti-a-molti** tra **Squadra** e **Competizione**. Questa associazione fa riferimento al fatto che una o più squadre possono partecipare ad una competizione e ad una competizione possono partecipare zero o più squadre. Per questo motivo avrà come attributo **Stagione**(varchar(9)) dal formato yyyy/yyyy e indicherà le stagioni di ogni competizione.
VinceCalciatore	Associazione di tipo **molti-a-molti** tra **Calciatore** e **TrofeoCalciatore**. Questa associazione fa riferimento al fatto che un calciatore può vincere zero o più trofei e un trofeo può essere vinto da zero (in caso viene appena creato) o più calciatori. Per questo motivo avrà come attributi **Stagione**(Char(9)) dal formato yyyy/yyyy che fa riferimento alla stagione o all'anno solare della vincita del determinato trofeo
VinceSquadra	Associazione di tipo **molti-a-molti** tra **Squadra** e **TrofeoCompetizione**. Questa associazione fa riferimento al fatto che una squadra può vincere zero o più trofei e un trofeo può essere vinto da zero (in caso viene appena reato) o più squadre. Per questo motivo avrà come attributi **Stagione**(Char(9)) dal formato yyyy/yyyy che fa riferimento alla stagione o all'anno solare della vincita del determinato trofeo
Ha	Associazione di tipo **uno-a-uno** tra **TrofeoCompetizione** e **Competizione**. Questa associazione si riferisce al fatto che un trofeo è associato ad una sola competizione e che una competizione ha solo un trofeo
Gestisce	Associazione di tipo **molti-a-molti** tra **Amministratore** e **Calciatore**. Questa associazione fa riferimento al fatto che uno o più amministratori possono modificare un calciatore e che uno o più calciatori possono essere modificati da un amministratore

3 Ristrutturazione modello concettuale

La ristrutturazione di uno schema concettuale si divide in 6 fasi:

1. Analisi delle ridondanze
2. Eliminazione delle generalizzazioni
3. Eliminazione attributi multivalore
4. Eliminazione attributi strutturati
5. Partizionamento/accorpamento di entità e associazioni
6. Scelta degli identificatori primari

1) L'analisi delle ridondanze consiste nell'analizzare quest'ultime in uno schema concettuale. Una ridondanza corrisponde alla presenza di un dato che può essere derivato da altre informazioni ed essa presenta dei vantaggi, ovvero la semplificazione delle query, ma anche dei svantaggi cioè verrà occupato maggiore spazio e gli aggiornamenti risulteranno più pesanti.

2) Possiamo eliminare una generalizzazione in 3 modi:

1. **Accorpamento delle figlie nel padre**. In questo primo caso le classi figlie verranno eliminate e i loro attributi verranno inseriti nella classe padre con l'aggiunta dell'attributo tipo che specifica in quale dei due casi ci troviamo.
2. **Accorpamento del padre nelle figlie.** In questo secondo caso elimineremo l'entità padre e i suoi attributi li inseriremo nelle classi figlie.
3. **Sostituzione della generalizzazione con associazioni.** In questo terzo caso la generalizzazione viene trasformata in associazioni tra padre e figlie.

3) Il modello relazionale ristrutturato non consente la rappresentazione degli attributi multivalore pertanto ci sono tre modi per eliminarli:

1. Per quell'attributo si crea un entità esterna associata.
2. L'attributo viene trattato in modo non più multiplo ma singolo.
3. Replicare l'attributo nella classe tante volte quanto si crede che servirà.

4) Il modello relazionale ristrutturato non consente l'esistenza di attributi strutturati pertanto ci sono tre modi per eliminarli:

1. Si introduce una classe per l'attributo strutturato.
2. Gli attributi che compongono l'attributo strutturato diventano attributi della classe

3. Gli attributi che compongono l'attributo strutturato vengono eliminati.

5) L'accorpamento/partizionamento di entità è un'operazione generalmente effettuata su associazioni di tipo 1:1 ed è conveniente quando le operazioni accedono a dati esistenti su entrambe le classi. Nel caso dell'accorpamento consiste nell'inglobare una classe nell'altra, mentre nel caso del partizionamento si decompongono le associazioni o le entità.

6) Nell'identificare una chiave primaria si seguono vari criteri tra cui:

1. Escludere gli attributi con possibilità di ritrovarsi valori NULL
2. Scegliere l'attributo o gli attributi che vengono più coinvolti nelle associazioni
3. Scegliere il numero minimo di attributi

Se nessun attributo rispetta questi criteri allora si inserisce un attributo contenente valori speciali (codici) che si generano in modo sistematico e identificano le occorrenze delle entità in modo univoco.

3.1 Analisi delle ridondanze

Nel nostro caso non sono presenti ridondanze

3.2 Eliminazione delle generalizzazioni

Nel nostro modello non sono presenti gerarchie.

3.3 Eliminazione attributi multivalore

Gli attributi multivalore **Ruolo, Nazionalità e Feature** li abbiamo rimossi con la stessa procedura, ovvero rendendole delle entità assestanti.

3.4 Eliminazione attributi strutturati

Nel nostro modello non sono presenti attributi composti.

3.5 Partizionamento/Accorpamento di entità e associazioni

Nel nostro modello abbiamo partizionato l'attributo nazionalità all'interno delle entità **Squadra** e **Competizione** associandole all'entità **Nazionalità** che abbiamo ottenuto durante l'eliminazione degli attributi multivalore.

3.6 Scelta degli identificatori primari

Nell'entità **Calciatore** scegliamo come chiave primaria un **idCalciatore**.

nell'entità **Amministratore** scegliamo come chiave primaria **username**

Nell'entità **Squadra** dato che esistono squadre con lo stesso nome e della stessa nazionalità come ad esempio "F.C. Barcelona" che ha sia una squadra maschile che femminile senza specificarlo nel nome, perciò abbiamo deciso di aggiungere un **idSquadra** come chiave primaria per evitare ridondanze.

Nell'entità **Competizione** individuiamo come chiave primaria il **Nome** dato che è sempre diverso.

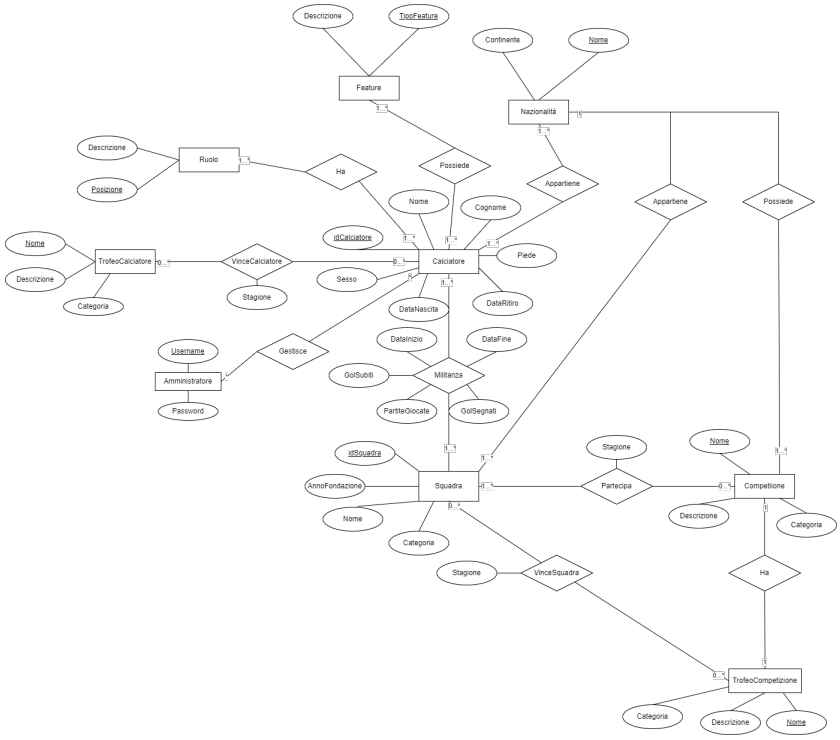
Nell'entità **TrofeoCalciatore** e **TrofeoCompetizione** individuiamo come chiave primaria l'attributo **Nome** dato che ogni trofeo ha un nome univoco.

Nell'entità **Ruolo** individuiamo come chiave primaria l'attributo **Posizione** dato che è univoco

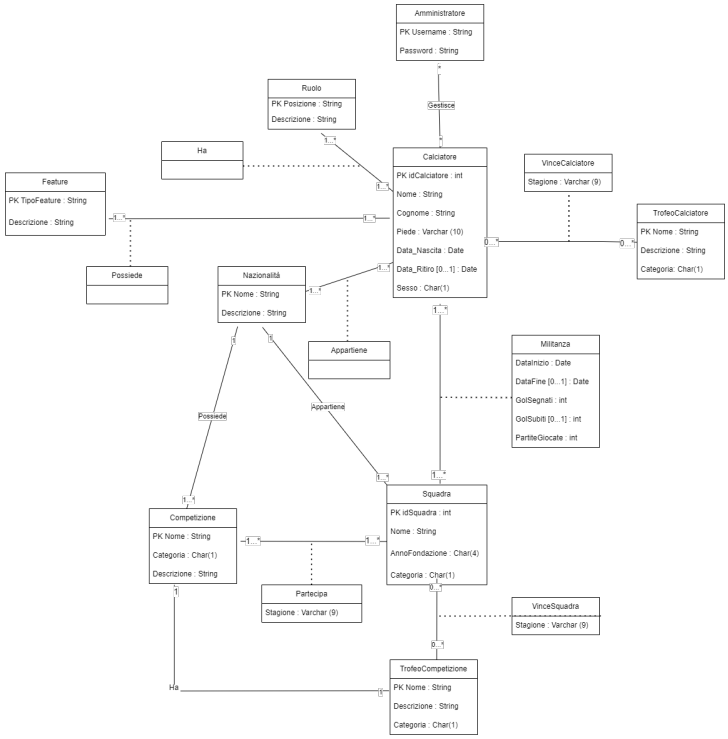
Nell'entità **Feature** individuiamo come chiave primaria l'attributo **TipoFeature** dato che ci garantisce l'unicità.

Nell'entità **Nazionalità** individuiamo come chiave primaria l'attributo **Nome** che ci permette l'unicità della chiave.

3.7 Modello ER ristrutturato



3.8 Modello UML ristrutturato



3.9 Dizionario entità ristrutturato

Classe	Attributi	Descrizione
Ruolo	(PK)Posizione Descrizione	Classe che contiene tutti i ruoli del calcio
Feature	(PK)TipoFeature Descrizione	Classe che contiene tutte le feature che un calciatore può avere
Nazionalità	(PK)Nome Continente	Classe che racchiude tutte le nazionalità esistenti e i continenti a cui appartengono
Calciatore	(PK)idCalciatore Nome Cognome DataNascita Piede Sesso DataRitiro	Classe che conserva tutte le informazioni dei calciatori
TrofeoCalciatore	(PK)Nome Descrizione Categoria	Classe che contiene tutti i trofei individuali
Squadra	(PK)idSquadra Nome AnnoFondazione Categoria	Classe che conserva le informazioni delle squadre
TrofeoCompetizione	(PK)Nome Descrizione Categoria	Classe che contiene tutti i trofei di tutte le competizioni
Competizione	(PK)Nome Descrizione Categoria	Classe che contiene tutte le competizioni
Amministratore	(PK)Username Password	Classe che contiene i dati di accesso degli amministratori.

3.10 Dizionario associazioni ristrutturato

Associazioni	Classi coinvolte	Descrizione
Ha	Ruolo e Calciatore TrofeoCompetizione e Competizione	Uno o più ruoli appartengono ad un calciatore e uno o più calciatori hanno un ruolo Un trofeo di squadra appartiene ad una competizione e una competizione ha un trofeo di squadra
Appartiene	Nazionalità e Calciatore Squadra e Nazionalità	Uno o più nazionalità appartengono ad un calciatore e uno o più calciatori hanno una nazionalità Una o più squadre appartengono ad una nazionalità e una nazionalità appartiene ad una squadra
Possiede	Feature e Calciatore Competizione e Nazionalità	Uno o più feature appartengono ad un calciatore e uno o più calciatori hanno una feature Una o più competizioni possiedono una nazionalità e una nazionalità possiede una competizione
VinceCalciatore	TrofeoCalciatore e Calciatore	Zero, uno o più trofei individuali vengono vinti da un calciatore e zero, uno o più calciatori vincono un trofei individuale
VinceSquadra	TrofeoCompetizione e Squadra	Zero, uno o più trofei della competizione vengono vinti da una squadra e zero, uno o più squadre vincono un trofeo della competizione
Militanza	Calciatore e Squadra	Uno o più calciatori militano in una squadra e in una o più squadra militano dei calciatori
Partecipa	Squadra e Competizione	Una o più squadre partecipano ad una competizione e zero, una o più competizione hanno una squadra
Gestisce	Amministratore e Calciatore	Uno o più amministratori modificano un calciatore, uno o più calciatori vengono modificati da un amministratore

4 Modello logico

Gli attributi sottolineati sono chiavi primarie.
Gli attributi con affianco una freccia↑sono chiavi esterne.
Gli attributi con l'asterisco* possono essere NULL.
Ruolo (Posizione, Descrizione)
Feature (TipoFeature, Descrizione)
Nazionalità (Nome, Continente)
Calciatore (idCalciatore, Nome, Cognome, Piede, Sesso, DataRitiro*, DataNascita)
TrofeoCalciatore (Nome, Descrizione, Categoria)
Squadra (idSquadra, Nazionalità↑, Nome, AnnoFondazione, Categoria)
TrofeoCompetizione (Nome, Descrizione, Categoria)
Competizione (Nome, Categoria, Nazionalità↑, TrofeoCompetizione↑)
Amministratore (Username, Password)
Ha (Ruolo↑, Calciatore↑)
Possiede (Feature↑, Calciatore↑)
Appartiene (Nazionalità↑, Calciatore↑)
VinceCalciatore (Stagione, TrofeoCalciatore↑, Calciatore↑)
Militanza (Calciatore↑, Squadra↑, DataInizio, DataFine*, GolSegnati, PartiteGiocate, GolSubiti*)
VinceSquadra (Stagione, TrofeoSquadra↑, Squadra↑)
Partecipa (Stagione, Squadra↑, Competizione↑)

5 Modello fisico

5.1 Creazione Database

```
CREATE DATABASE Traccia3
```

5.2 Creazione Schema

```
CREATE SCHEMA progettobd
```

5.3 Domini

```
CREATE DOMAIN String AS varchar(100) NOT NULL;

CREATE DOMAIN TipoStagione AS char(9) NOT NULL
CHECK(VALUE LIKE ('____/____'));

CREATE DOMAIN TipoPiede AS varchar(10) NOT NULL
CHECK(VALUE IN ('Destro','Sinistro','Ambidestro'));

CREATE DOMAIN Genere AS char(1) NOT NULL
CHECK(VALUE IN ('M','F'));

CREATE DOMAIN TipoRuolo AS varchar(27) NOT NULL
CHECK(VALUE IN ( 'Ala destra', 'Attaccante centrale', 'Ala sinistra', 'Seconda
Punta', 'Trequartista', 'Esterno sinistro', 'Esterno destro', 'Centrocampista centrale',
'Mezzala sinistra', 'Mezzala destra', 'Centrocampista difensivo', 'Mediano', 'Terzi-
no sinistro', 'Terzino destro', 'Difensore centrale sinistro', 'Difensore centrale destro',
'Portiere', 'Libero', 'Esterno basso destro', 'Esterno basso sinistro' ));

CREATE DOMAIN TipoContinente AS varchar(11) NOT NULL
CHECK(VALUE IN ( 'NordAmerica', 'SudAmerica', 'Africa', 'Oceania', 'Euro-
pa', 'Asia' ));

CREATE DOMAIN TipoAnnoFondazione AS INT NOT NULL
CHECK(VALUE >= 1857);

CREATE DOMAIN TipoNazione AS String NOT NULL
CHECK(VALUE IN ( 'Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
'Anguilla', 'Antartide', 'Antigua e Barbuda', 'Arabia Saudita', 'Argentina', 'Arme-
nia', 'Aruba', 'Australia', 'Austria', 'Azerbaigian', 'Bahamas', 'Bahrein', 'Bangla-
desh', 'Barbados', 'Belgio', 'Belize', 'Benin', 'Bermuda', 'Bhutan', 'Bielorussia',
'Birmania', 'Bolivia', 'Bosnia ed Erzegovina', 'Botswana', 'Brasile', 'Brunei', 'Bul-
garia', 'Burkina Faso', 'Burundi', 'Cambogia', 'Camerun', 'Canada', 'Capo Verde',
'Ciad', 'Cile', 'Cina', 'Cipro', 'Citta del Vaticano', 'Colombia', 'Comore', 'Corea
del Nord', 'Corea del Sud', 'Costa d"Avorio', 'Costa Rica', 'Croazia', 'Cuba', 'Cu-
```

racao', 'Danimarca', 'Dominica', 'Ecuador', 'Egitto', 'El Salvador', 'Emirati Arabi Uniti', 'Eritrea', 'Estonia', 'Etiopia', 'Figi', 'Filippine', 'Finlandia', 'Francia', 'Gabon', 'Gambia', 'Georgia', 'Georgia del Sud e isole Sandwich meridionali', 'Germania', 'Ghana', 'Giamaica', 'Giappone', 'Gibilterra', 'Gibuti', 'Giordania', 'Grecia', 'Grenada', 'Groenlandia', 'Guadalupa', 'Guam', 'Guatemala', 'Guernsey', 'Guinea', 'Guinea-Bissau', 'Guinea Equatoriale', 'Guyana', 'Guyana francese', 'Haiti', 'Honduras', 'Hong Kong', 'India', 'Indonesia', 'Inghilterra', 'Iran', 'Iraq', 'Irlanda', 'Islanda', 'Isola Bouvet', 'Isola di Man', 'Isola di Natale', 'Isola Norfolk', 'Isole Aland', 'Isole BES', 'Isole Cayman', 'Isole Cocos (Keeling)', 'Isole Cook', 'Isole Faroe', 'Isole Falkland', 'Isole Heard e McDonald', 'Isole Marianne Settentrionali', 'Isole Marshall', 'Isole minori esterne degli Stati Uniti', 'Isole Pitcairn', 'Isole Salomone', 'Isole Vergini britanniche', 'Isole Vergini americane', 'Israele', 'Italia', 'Jersey', 'Kazakistan', 'Kenya', 'Kirghizistan', 'Kiribati', 'Kuwait', 'Laos', 'Lesotho', 'Lettonia', 'Libano', 'Liberia', 'Libia', 'Liechtenstein', 'Lituania', 'Lussemburgo', 'Macao', 'Macedonia', 'Madagascar', 'Malawi', 'Malesia', 'Maldive', 'Mali', 'Malta', 'Marocco', 'Martinica', 'Mauritania', 'Mauritius', 'Mayotte', 'Messico', 'Micronesia', 'Moldavia', 'Mongolia', 'Montenegro', 'Montserrat', 'Mozambico', 'Namibia', 'Nauru', 'Nepal', 'Nicaragua', 'Niger', 'Nigeria', 'Niue', 'Norvegia', 'Nuova Caledonia', 'Nuova Zelanda', 'Oman', 'Paesi Bassi', 'Pakistan', 'Palau', 'Palestina', 'Panama', 'Papua Nuova Guinea', 'Paraguay', 'Perų', 'Polinesia Francese', 'Polonia', 'Porto Rico', 'Portogallo', 'Monaco', 'Qatar', 'Regno Unito', 'Repubblica Democratica del Congo', 'Repubblica Ceca', 'Repubblica Centrafricana', 'Repubblica del Congo', 'Repubblica Dominicana', 'Riunione', 'Romania', 'Ruanda', 'Russia', 'Sahara Occidentale', 'Saint Kitts e Nevis', 'Santa Lucia', 'Saint Vincent e Grenadine', 'Saint-Barthœlemy', 'Saint-Martin', 'Saint-Pierre e Miquelon', 'Samoa', 'Samoa Americane', 'San Marino', 'Sao Tomœ e Prˆncipe', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore', 'Sint Maarten', 'Siria', 'Slovacchia', 'Slovenia', 'Somalia', 'Spagna', 'Sri Lanka', 'Stati Uniti', 'Sudafrica', 'Sudan', 'Sudan del Sud', 'Suriname', 'Svalbard e Jan Mayen', 'Svezia', 'Svizzera', 'Swaziland', 'Taiwan', 'Tagikistan', 'Tanzania', 'Terre australi e antartiche francesi', 'Territorio britannico dell"oceano Indiano', 'Thailandia', 'Timor Est', 'Togo', 'Tokelau', 'Tonga', 'Trinidad e Tobago', 'Tunisia', 'Turchia', 'Turkmenistan', 'Turks e Caicos', 'Tuvalu', 'Ucraina', 'Uganda', 'Ungheria', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam', 'Wallis e Futuna', 'Yemen', 'Zambia', 'Zimbabwe', 'Europa', 'Asia', 'Oceania', 'Africa', 'NordAmerica', 'SudAmerica'))

5.4 Creazione tabelle

```
CREATE TABLE Ruolo (
        Posizione String,
        Descrizione String
);

CREATE TABLE Feauture (
        TipoFeature String,
        Descrizione String
);
```

```
CREATE TABLE Nazionalità (
        Nome TipoNazione,
        Continente TipoContinente
);

CREATE TABLE TrofeoCalciatore (
        Nome String,
        Descrizione String,
        Categoria Genere
);

CREATE TABLE Calciatore (
        idCalciatore SERIAL,
        Nome String,
        Cognome String,
        Piede TipoPiede,
        Sesso Genere,
        DataNascita DATE NOT NULL,
        DataRitiro DATE DEFAULT NULL
);

CREATE TABLE Amministratore (
        Username String,
        "Password" String
);

CREATE TABLE Squadra (
        idSquadra SERIAL,
        Nome String,
        Categoria Genere,
        AnnoFondazione TipoAnnoFondazione,
        Nazionalità TipoNazione,

);

CREATE TABLE TrofeoCompetizione (
        Nome String,
        Descrizione String
        Categoria Genere
);

CREATE TABLE Competizione (
        Nome String,
        Descrizione String,
        Categoria Genere,
```

```
        Nazionalità TipoNazione,
        TrofeoCompetizione String,

);

CREATE TABLE VinceCalciatore(
        Stagione TipoStagione,
        Calciatore INT,
        TrofeoCalciatore String,
);

CREATE TABLE Militanza(
        DataInizio DATE,
        DataFine DATE DEFAULT NULL,
        PartiteGiocate INT DEFAULT 0 NOT NULL,
        GolSegnati INT DEFAULT 0 NOT NULL,
        GolSubiti INT DEFAULT NULL,
        Calciatore INT,
        Squadra INT,
);

CREATE TABLE Partecipa(
        Stagione TipoStagione,
        Squadra INT,
        Competizione String,
);

CREATE TABLE VinceSquadra(
        Stagione TipoStagione,
        Squadra INT,
        TrofeoCompetizione String,
);

CREATE TABLE Ha(
        Ruolo String,
        Calciatore INT
);

CREATE TABLE Possiede(
        Feature String,
        Calciatore INT
);
```

```
    CREATE TABLE Appartiene(
        Nazionalità String,
        Calciatore INT
)
```

5.5 Vincoli

Il vincolo dei valori **NULL e Predefiniti** si verifica grazie ad SQL che consente di avere un attributo con valore **NULL** quindi se si vuole impedire ciò utilizziamo il vincolo **NOT NULL**. Inoltre SQL consente che un attributo abbia un valore di default tramite la clausola **DEFAULT value**.

Il vincolo **CHECK** serve a controllare che un attributo rispetti precise condizioni.

Il vincolo di **chiave** viene implementato all'interno dei create table dove per specificare la presenza di una chiave primaria utilizziamo la clausola attributo **PRIMARY KEY**.

Il vincolo di **unicità** lo utilizziamo per specificare che all'interno della tabella quei determinati valori sono unici, il vincolo **UNIQUE** ci specifica una chiave secondaria candidata ad essere chiave primaria.

Il vincolo di **integrità referenziale** lo specifichiamo tramite la clausola **FOREIGN KEY** seguita dall'attributo che fa da chiave esterna e dalla parola **REFERENCES** seguita dall'attributo esterno a cui si riferisce.

5.6 Creazione vincoli

ALTER TABLE Ruolo
ADD COSTRAINT chiave_ruolo **PRIMARY KEY** (Posizione);

ALTER TABLE Feature
ADD COSTRAINT chiave_feature **PRIMARY KEY** (TipoFeature);

ALTER TABLE Nazionalità
ADD COSTRAINT chiave_nazionalità **PRIMARY KEY** (Nome);

ALTER TABLE TrofeoCalciatore
ADD COSTRAINT chiave_trofeoCalciatore **PRIMARY KEY** (Nome);

ALTER TABLE Calciatore
ADD COSTRAINT chiave_calciatore **PRIMARY KEY** (idCalciatore);

ALTER TABLE Calciatore
ADD COSTRAINT controllo_dataC **CHECK** (dataRitiro > dataNascita);

ALTER TABLE Amministratore
ADD COSTRAINT chiave_amministratore **PRIMARY KEY** (Username);

ALTER TABLE Squadra
ADD COSTRAINT chiave_squadra **PRIMARY KEY** (idSquadra),
ADD COSTRAINT chiaveEsternaS_nazionalità **FOREIGN KEY** (Nazionalità) **REFERENCES** Nazionalità(Nome);

ALTER TABLE TrofeoCompetizione
ADD COSTRAINT chiave_trofeoCompetizione **PRIMARY KEY** (Nome);

ALTER TABLE Competizione
ADD COSTRAINT chiave_competizione **PRIMARY KEY** (Nome),
ADD COSTRAINT chiaveEsternaC_nazionalità **FOREIGN KEY** (Nazionalità) **REFERENCES** Nazionalità(Nome);
ADD COSTRAINT chiaveEsternaC_trofeoCompetizione **FOREIGN KEY** (TrofeoCompetizione) **REFERENCES** TrofeoCompetizione(Nome);

ALTER TABLE VinceCalciatore
ADD COSTRAINT chiave_vinceCalciatore **PRIMARY KEY** (Stagione, Calciatore, TrofeoCalciatore),
ADD COSTRAINT chiaveEsternaV_calciatore **FOREIGN KEY** (Calciatore) **REFERENCES** Calciatore(idCalciatore),
ADD COSTRAINT chiaveEsternaV_trofeoCalciatore **FOREIGN KEY** (TrofeoCalciatore) **REFERENCES** TrofeoCalciatore(Nome);

ALTER TABLE Militanza
ADD COSTRAINT chiave_militanza **PRIMARY KEY** (DataInizio, Calciatore, Squadra),
ADD COSTRAINT chiaveEsternaM_calciatore **FOREIGN KEY** (Calciatore) **REFERENCES** Calciatore(idCalciatore),
ADD COSTRAINT chiaveEsternaM_squadra **FOREIGN KEY** (Squadra) **REFERENCES** Squadra(idSquadra);

ALTER TABLE Militanza
ADD COSTRAINT controllo_data **CHECK** (dataFine > dataInizio);

ALTER TABLE Partecipa
ADD COSTRAINT chiave_partecipa **PRIMARY KEY** (Stagione, Squadra, Competizione),
ADD COSTRAINT chiaveEsternaP_squadra **FOREIGN KEY** (Squadra) **REFERENCES** Squadra(idSquadra),
ADD COSTRAINT chiaveEsternaC_competizione **FOREIGN KEY** (Competizione) **REFERENCES** Competizione(Nome);

ALTER TABLE VinceSquadra
ADD COSTRAINT chiave_vinceSquadra **PRIMARY KEY** (Stagione, Squadra, TrofeoCompetizione),
ADD COSTRAINT chiaveEsternaVS_squadra **FOREIGN KEY** (Squadra) **REFERENCES** Squadra(idSquadra),
ADD COSTRAINT chiaveEsternaVS_trofeoCompetizione **FOREIGN KEY** (TrofeoCompetizione) **REFERENCES** TrofeoCompetizione(Nome);

ALTER TABLE Ha
ADD COSTRAINT chiave_ha **PRIMARY KEY** (Ruolo, Calciatore),
ADD COSTRAINT chiaveEsternaH_ruolo **FOREIGN KEY** (Ruolo) **REFERENCES**Ruolo(Posizione),
ADD COSTRAINT chiaveEsternaH_calciatore **FOREIGN KEY** (Calciatore) **REFERENCES** Calciatore(idCalciatore);

ALTER TABLE Possiede
ADD COSTRAINT chiave_possiede **PRIMARY KEY** (Feature, Calciatore),
ADD COSTRAINT chiaveEsternaP_feature **FOREIGN KEY** (Feature) **REFERENCES** Feature(TipoFeature),
ADD COSTRAINT chiaveEsternaP_calciatore **FOREIGN KEY** (Calciatore) **REFERENCES** Calciatore(idCalciatore);

ALTER TABLE Appartiene
ADD COSTRAINT chiave_appartiene **PRIMARY KEY** (Nazionalità, Calciatore),
ADD COSTRAINT chiaveEsternaA_nazionalità **FOREIGN KEY** (Nazionalità) **REFERENCES** Nazionalità(Nome),

ADD COSTRAINT chiaveEsternaA_calciatore **FOREIGN KEY** (Calciatore)
REFERENCES Calciatore(idCalciatore)

5.6.1 Dizionario vincoli

Vincolo	Applicazione
Valori NULL e predefiniti	I vincoli di valori NULL sono impostati su tutti gli attributi delle tabelle tranne per quei attributi dove è presente la clausola NOT NULL, ovvero: - **DataNascita** della tabella **Calciatore** - **PartiteGiocate** della tabella **Militanza** - **GolSegnati** della tabella **Militanza** I vincoli predefiniti sono invece impostati sugli attributi: - **DataRitiro** della tabella **Calciatore** ed è impostato a **NULL** - **DataFine** della tabella **Militanza** ed è impostato a **NULL** - **PartiteGiocate** della tabella **Militanza** ed è impostato a **0** - **GolSegnati** della tabella **Militanza** ed è impostato a **0** - **GolSubiti** della tabella **Militanza** ed è impostato a **NULL**

Tuple (Check)	I vincoli di check sono stati inseriti nei domini e nelle tabelle **Calciatore** e **Militanza** e sono i seguenti: - **CHECK**(**VALUE LIKE** ('________')) che controlla se i dati inseriti siano nel formato richiesto, cioè quello di yyyy/yyyy - **CHECK**(**VALUE IN** ('destro','sinistro','ambidestro')) che controlla se i dati inseriti facciano parte del gruppo di valori scelto, ovvero il piede preferito di un calciatore - **CHECK**(**VALUE IN** ('M','F')) che controlla se i dati inseriti facciano parte del gruppo di valori scelto (quindi se la categoria è M (Maschio/Maschile) o F (Femmina/Femminile)) - **CHECK**(**VALUE IN** ('Ruoli di calcio')) controlla se i dati inseriti facciano parte del gruppo di valori scelto, ovvero tutti i ruoli esistenti nel calcio - **CHECK**(**VALUE IN** ('NordAmerica', 'SudAmerica', 'Africa', 'Oceania', 'Europa', 'Asia')) che controlla se i dati inseriti facciano parte del gruppo di valori scelto cioè dei continenti in cui il mondo è suddiviso - **CHECK**(**VALUE** >= 1857) che controlla se l'anno di fondazione di una società sia superiore all'anno 1857 (anno della prima squadra fondata nella storia del calcio) - **CHECK**(**VALUE IN** ('Nazioni del mondo')) che controlla se i dati inseriti facciano parte del gruppo di valori scelto, ovvero tutte le nazioni del mondo - **CHECK** (dataRitiro > dataNascita) che controlla se la data di nascita del calciatore non sia maggiore della data di ritiro nella tabella calciatore - **CHECK** (dataFine > dataInizio) che controlla se la data di fine di contratto sia maggiore della data inizio contratto nella tabella militanza

Chiave	I vincoli di chiave sono stati inseriti in ogni tabella tramite gli **ADD COSTRAINT** così da avere dei nomi per questi vincoli e sono i seguenti: - **Posizione** della tabella **Ruolo** - **TipoFeature** della tabella **Feature** - **Nome** della tabella **Nazionalità** - **Nome** della tabella **TrofeoCalciatore** - **idCalciatore** della tabella **Calciatore** - **Username** della tabella **Amministratore** - **idSquadra** della tabella **Squadra** - **Nome** della tabella **TrofeoCompetizione** - **Nome** della tabella **Competizione** - **Stagione, Calciatore, TrofeoCalciatore** della tabella **VinceCalciatore** - **DataInizio, Calciatore, Squadra** della tabella **Militanza** - **Stagione, Squadra, Competizione** della tabella **Partecipa** - **Stagione, Squadra, TrofeoCompetizione** della tabella **VinceSquadra** - **Ruolo, Calciatore** della tabella **Ha** - **Feature, Calciatore** della tabella **Possiede** - **Nazionalità, Calciatore** della tabella **Appartiene**
Unicità	Non ci sono vincoli di unicità

Integrità referenziale	I vincoli di integrità referenziale sono stati inseriti in ogni tabella tramite gli **ADD COSTRAINT** così da avere dei nomi per questi vincoli e sono i seguenti: - **Nazionalità** della tabella **Squadra** che fa riferimento al **Nome** della tabella **Nazionalità** - **Nazionalità** della tabella **Competizione** che fa riferimento al **Nome** della tabella **Nazionalità** - **TrofeoCompetizione** della tabella **Competizione** che fa riferimento al **Nome** della tabella **TrofeoCompetizione** - **Calciatore** della tabella **VinceCalciatore** che fa riferimento al **idCalciatore** della tabella **Calciatore** - **TrofeoCalciatore** della tabella **VinceCalciatore** che fa riferimento al **Nome** della tabella **TrofeoCalciatore** - **Calciatore** della tabella **Militanza** che fa riferimento al **idCalciatore** della tabella **Calciatore** - **Squadra** della tabella **Militanza** che fa riferimento all'**idSquadra** della tabella **Squadra** - **Squadra** della tabella **Partecipa** che fa riferimento all'**idSquadra** della tabella **Squadra** - **Competizione** della tabella **Partecipa** che fa riferimento al **Nome** della tabella **Competizionwe** - **Squadra** della tabella **VinceSquadra** che fa riferimento all'**idSquadra** della tabella **Squadra** - **TrofeoCompetizione** della tabella **VinceSquadra** che fa riferimento al**Nome** della tabella **TrofeoCompetizione** - **Ruolo** della tabella **Ha** che fa riferimento alla **Posizione** della tabella **Ruolo** - **Calciatore** della tabella **Ha** che fa riferimento all'**idCalciatore** della tabella **Calciatore**

	- **Feature** della tabella **Possiede** che fa riferimento al **TipoFeature** della tabella **Feature** - **Calciatore** della tabella **Possiede** che fa riferimento al **idCalciatore** della tabella **Calciatore** - **Nazionalità** della tabella **Appartiene** che fa riferimento al **Nome** della tabella **Nazionalità** - **Calciatore** della tabella **Appartiene** che fa riferimento all'**idCalciatore** della tabella **Calciatore**

5.7 Creazione trigger

5.7.1 calciatore_id

Col seguente trigger ci assicuriamo che gli id della tabella Calciatore siano sempre consecutivi senza salti dovuti da inserimenti in altre tabelle che hanno anche essi degli id. Difatti prima di un inserimento nella tabella Calciatore controlliamo qual è l'ultimo id inserito e se il nuovo id non corrisponde all'ultimo inserito aumentato di 1 esso verrà modificato in ciò, altrimenti non ci saranno modifiche

```
CREATE OR REPLACE FUNCTION calciatore_id()
RETURNS TRIGGER AS $$

DECLARE
        max_id INTEGER;
BEGIN
        SELECT MAX(idCalciatore) INTO max_id FROM Calciatore;

        IF max_id IS NULL THEN
                NEW.idCalciatore := 1;
        ELSIF NEW.idCalciatore IS NULL OR
        NEW.idCalciatore != max_id + 1 THEN
                NEW.idCalciatore := max_id + 1;
        END IF;

        RETURN NEW;
END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER controllo_idCalciatore
BEFORE INSERT ON Calciatore
FOR EACH ROW
EXECUTE FUNCTION calciatore_id()
```

5.7.2 squadra_id

Col seguente trigger ci assicuriamo che gli id della tabella Squadra siano sempre consecutivi senza salti dovuti da inserimenti in altre tabelle che hanno anche essi degli id. Difatti prima di un inserimento nella tabella Squadra controlliamo qual è l'ultimo id inserito e se il nuovo id non corrisponde all'ultimo inserito aumentato di 1 esso verrà modificato in ciò, altrimenti non ci saranno modifiche

```
CREATE OR REPLACE FUNCTION squadra_id() RETURNS
TRIGGER AS $$

DECLARE
        max_id INTEGER;
BEGIN
        SELECT MAX(idSquadra) INTO max_id FROM Squadra;

        IF max_id IS NULL THEN
                NEW.idSquadra := 1;
        ELSIF NEW.idSquadra IS NULL OR
        NEW.idSquadra != max_id + 1 THEN
                NEW.idSquadra := max_id + 1;
        END IF;

        RETURN NEW;
END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER controllo_idSquadra
BEFORE INSERT ON Squadra
FOR EACH ROW
EXECUTE FUNCTION squadra_id()
```

5.7.3 naz_calciatoreN

Dato che nella tabella nazionalità ci troviamo anche i valori Europa, Africa, Asia, Oceania, NordAMerica e SudAmerica, con questo trigger evitiamo che l'amministratore di sistema assegni uno di questi continenti come nazionalità al calciatore. Ciò che fa è controllare se il valore della colonna nazionalità che stiamo andando ad inserire non sia tra quelli elencati precedentemente. In caso questa condizione fosse vera, l'inserimento andrà a buon fine, altrimenti l'amministratore verrà allertato dell'errore che sta commettendo

```
CREATE OR REPLACE FUNCTION naz_calciatoreN()
RETURNS TRIGGER AS $$

DECLARE

BEGIN
    IF NEW.Nazionalita NOT IN ('Europa','Africa','Asia','Oceania','NordAmerica','SudAmerica')
    THEN
        RETURN NEW;
    ELSE
        RAISE NOTICE 'Il calciatore non può avere questa nazionalità';
        RETURN NULL;
    END IF;
END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER naz_calciatore
    BEFORE INSERT ON Appartiene
    FOR EACH ROW
    EXECUTE FUNCTION naz_calciatoreN();
```

5.7.4 naz_squadraN

Dato che nella tabella nazionalità ci troviamo anche i valori Europa, Africa, Asia, Oceania, NordAMerica e SudAmerica, con questo trigger evitiamo che l'amministratore di sistema assegni uno di questi continenti come nazionalità alla squadra. Ciò che fa è controllare se il valore della colonna nazionalità che stiamo andando ad inserire non sia tra quelli elencati precedentemente. In caso questa condizione fosse vera, l'inserimento andrà a buon fine, altrimenti l'amministratore verrà allertato dell'errore che sta commettendo

```
CREATE OR REPLACE FUNCTION naz_squadraN()
RETURNS TRIGGER AS $$

DECLARE

BEGIN
    IF NEW.Nazionalita NOT IN ('Europa','Africa','Asia','Oceania','NordAmerica','SudAmerica')
    THEN
        RETURN NEW;
    ELSE
        RAISE NOTICE 'La squadra non può avere questa nazionalità';
        RETURN NULL;
    END IF;
END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER naz_squadra
    BEFORE INSERT ON Squadra
    FOR EACH ROW
    EXECUTE FUNCTION naz_squadraN();
```

5.7.5 controllo_dataNM

Questo trigger controlla se le date di inizio e fine contratto di un calciatore in una determinata squadra siano congrue con la data di nascita del calciatore, cioè quindi che quest'ultimo non venga dopo le date del contratto. Per fare ciò ci salviamo la data di nascita del calciatore in una variabile e subito dopo lo confrontiamo con dataInizio e dataFine della tabella militanza. Se queste ultime risulteranno essere minori della data di nascita verrà notificato l'errore, altrimenti l'inserimento andrà a buon fine

```
CREATE OR REPLACE FUNCTION controllo_dataNM()
RETUNRS TROGGER AS $$

DECLARE
data_nascita calciatore.dataNascita%TYPE;
BEGIN
        SELECT dataNascita
        INTO data_nascita
        FROM Calciatore
        WHERE idCalciatore = NEW.Calciatore;

        IF (NEW.dataInizio < data_nascita OR NEW.dataFine < data_nascita) THEN
                RAISE NOTICE 'Il calciatore non era ancora nato, controlla le
                                date del contratto';
                RETURN NULL;
        ELSE
                RETURN NEW;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER controllo_dataN
        BEFORE INSERT OR UPDATE ON militanza
        FOR EACH ROW
        EXECUTE FUNCTION controllo_dataNM();
```

5.7.6 controllo_dataRM

Questo trigger controlla se le date di inizio e fine contratto di un calciatore in una determinata squadra siano congrue con la data di ritiro del calciatore, cioè quindi che quest'ultimo non venga prima le date del contratto. Per fare ciò ci salviamo la data di ritiro del calciatore in una variabile e, prima controlliamo che esista e che non sia NULL perché in questo caso possiamo saltare ogni tipo di controllo e poter consentire da subito l'inserimento o l'aggiormaento, altrimenti se non risulta essere NULL lo confrontiamo con dataInizio e dataFine della tabella militanza. Se queste ultime risulteranno essere maggiori della data di ritiro verrà notificato l'errore, altrimenti l'inserimento andrà a buon fine

```
CREATE OR REPLACE FUNCTION controllo_dataRM()
RETUNRS TROGGER AS $$

DECLARE
data_ritiro calciatore.dataRitiro%TYPE;
BEGIN
        SELECT dataRitiro
        INTO data_ritiro
        FROM Calciatore
        WHERE idCalciatore = NEW.Calciatore;

        IF data_ritiro IS NULL THEN
                RETURN NEW;
        ELSIF (NEW.dataInizio > data_ritiro OR NEW.dataFine > data_ritiro)
        THEN
                RAISE NOTICE 'Il calciatore è ritirato, controlla le date del
                                contratto';
                RETURN NULL;
        ELSE
                RETURN NEW;
        END IF;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER controllo_dataR
        BEFORE INSERT OR UPDATE ON militanza
        FOR EACH ROW
        EXECUTE FUNCTION controllo_dataRM();
```

5.7.7 portiere_golsubiti

Il seguente trigger verifica se stiamo cercando di inserire i gol subiti ad un calciatore che tra i ruoli non ha quello del portiere. Ciò che fa è salvarsi in una variabile il numero di colonne che escono della query che va a cercare il ruolo portiere del determinato calciatore. Se la variabile avrà 0 come risultato e golSubiti non è stato inserito, allora l'inserimento andrà a buon fine senza problemi. Se la variabile avrà 0 come valore, ma golSubiti avrà un valore diverso da NULL in quel caso esso verrà sostituito con quel valore e andrà poi con l'inserimento. Infine se tutti questi controlli sono stati superati con una risposta negativa vuol dire che il calciatore ha il ruolo portiere e dunque, se golSubiti è uguale a NULL, il valore che verrà inserito verrà sostituito con lo 0

```
CREATE OR REPLACE FUNCTION portiere_golsubiti()
RETURNS TRIGGER AS $$

DECLARE
count_righe INTEGER;
BEGIN
        SELECT COUNT(*)
        INTO count_righe
        FROM Ha
        WHERE Ha.Calciatore = NEW.Calciatore AND Ruolo = 'Portiere';

        IF (count_righe = 0 AND NEW.golSubiti IS NULL) THEN
                RETURN NEW;
        ELSIF (count_righe = 0 AND NEW.golSubiti IS NOT NULL) THEN
                NEW.golSubiti := NULL;
                RETURN NEW;
        END IF;
        END IF;

        IF NEW.golSubiti IS NULL THEN
                NEW.golSubiti := 0;
        END IF;
        RETURN NEW;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER portiere
        BEFORE INSERT ON Militanza
        FOR EACH ROW
        EXECUTE FUNCTION portiere_golsubiti();
```

5.7.8 squadra_genereMF

Il seguente trigger controlla se stiamo inserendo un calciatore maschile in una squadra che appartiene alla categoria maschile e così con i calciatori di sesso femminile con le squadre di categoria femminile. Ciò che facciamo è salvarci in due variabili il sesso del calciatore coinvolto e la categoria della squadra coinvolta. Dopo di che li confrontiamo e se sono coincidenti allora l'inserimento andrà a buon fine, altrimenti all'amministratore gli verrà notificato errore

```
CREATE OR REPLACE FUNCTION squadra_genereMF()
RETURNS TRIGGER AS $$

DECLARE
genereS CHAR(1)
genereC CHAR(1)
BEGIN
        SELECT Categoria
        INTO genereS
        FROM Squadra
        WHERE idSquadra = NEW.Squadra;

        SELECT Sesso
        INTO genereC
        FROM Calciatore
        WHERE idCalciatore = NEW.Calciatore;

        IF genereS = genereC THEN
                RETURN NEW;
        ELSE
                RAISE NOTICE 'Il calciatore non può giocare in una squadra di
                                genere differente dal suo sesso';
                RETURN NULL;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER squadra_genere
        BEFORE INSERT ON Militanza
        FOR EACH ROW
        EXECUTE FUNCTION squadra_genereMF();
```

5.7.9 competizione_genereMF

Il seguente trigger controlla se stiamo inserendo una squadra maschile in una competizione che appartiene alla categoria maschile e così con le squadre femminili con le competizioni di categoria femminile. Ciò che facciamo è salvarci in due variabili la categoria della squadra coinvolta e della competizione coinvolta. Dopo di che li confrontiamo e se sono coincidenti allora l'inserimento andrà a buon fine, altrimenti all'amministratore gli verrà notificato errore

```
CREATE OR REPLACE FUNCTION competizione_genereMF()
RETURNS TRIGGER AS $$

DECLARE
genereS CHAR(1)
genereC CHAR(1)
BEGIN
        SELECT Categoria
        INTO genereS
        FROM Squadra
        WHERE idSquadra = NEW.Squadra;

        SELECT Categoria
        INTO genereC
        FROM Competizione
        WHERE Nome = NEW.Competizione;

        IF genereS = genereC THEN
                RETURN NEW;
        ELSE
                RAISE NOTICE 'La squadra non può giocare in una competizione
                                di genere differente dal suo';
                RETURN NULL;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER competizione_genere
        BEFORE INSERT ON Partecipa
        FOR EACH ROW
        EXECUTE FUNCTION competizione_genereMF();
```

5.7.10 trofeoC_genereMF

Il seguente trigger controlla se stiamo assegnando un trofeo del genere maschile ad un calciatore maschile o viceversa col genere femminile. Ciò che facciamo è salvarci in due variabili il sesso del calciatore coinvolto e la categoria del trofeo coinvolto. Dopo di che li confrontiamo e se sono coincidenti allora l'inserimento andrà a buon fine, altrimenti all'amministratore gli verrà notificato errore

```
CREATE OR REPLACE FUNCTION trofeoC_genereMF()
RETURNS TRIGGER AS $$

DECLARE
genereT CHAR(1)
genereC CHAR(1)
BEGIN
        SELECT Categoria
        INTO genereT
        FROM trofeoCalciatore
        WHERE Nome = NEW.trofeoCalciatore;

        SELECT Sesso
        INTO genereC
        FROM Calciatore
        WHERE idCalciatore = NEW.Calciatore;

        IF genereT = genereC THEN
                RETURN NEW;
        ELSE
                RAISE NOTICE 'Il calcitore non può vincere un trofeo che
                                appartiene ad una categoria differente dal suo sesso';
                RETURN NULL;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER trofeoC_genere
        BEFORE INSERT ON vinceCalciatore
        FOR EACH ROW
        EXECUTE FUNCTION trofeoC_genereMF();
```

5.7.11 trofeoS_genereMF

Il seguente trigger controlla se stiamo assegnando un trofeo del genere maschile ad una squadra maschile o viceversa col genere femminile. Ciò che facciamo è salvarci in due variabili la categoria della squadra coinvolta e la categoria del trofeo coinvolto. Dopo di che li confrontiamo e se sono coincidenti allora l'inserimento andrà a buon fine, altrimenti all'amministratore gli verrà notificato errore

```
CREATE OR REPLACE FUNCTION trofeoS_genereMF()
RETURNS TRIGGER AS $$

DECLARE
genereT CHAR(1)
genereS CHAR(1)
BEGIN
        SELECT Categoria
        INTO genereT
        FROM trofeoCompetizione
        WHERE Nome = NEW.trofeoCompetizione;

        SELECT Categoria
        INTO genereS
        FROM Squadra
        WHERE idCalciatore = NEW.Squadra;

        IF genereT = genereS THEN
                RETURN NEW;
        ELSE
                RAISE NOTICE 'La non puo vincere un trofeo che appartiene
                                ad una categoria differente dalla sua';
                RETURN NULL;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER trofeoS_genere
        BEFORE INSERT ON vinceSquadra
        FOR EACH ROW
        EXECUTE FUNCTION trofeoS_genereMF();
```

5.7.12 vinc_trofeo_competS

Il seguente trigger controlla se stiamo assegnanto il trofeo di una competizione ad una squadra che effettivamente ha partecipato a quella competizione in quella determinata stagione. Per fare ciò ci salviamo in una variabile la stagione in cui la squadra ha partecipato alla competizione legata al trofeo e se risulterà essere NULL allora verrà reso noto che la squadra in quella specifica stagione non ha partecipato alla competizione e l'inserimento verrà bloccato, altrimenti se il controllo darà esito negativo, l'inserimento andrà a buon fine

```
CREATE OR REPLACE FUNCTION vinc_trofeo_competS()
RETURNS TRIGGER AS $$

DECLARE
stagioneV varchar(10);
BEGIN
        SELECT Stagione
        INTO stagioneV
        FROM Partecipa INNER JOIN Competizione ON Partecipa.competizione
                                                = Competizione.nome
        WHERE NEW.Stagione = stagione AND NEW.Squadra = squadra AND
                NEW.trofeoCompetiione = trofeoCompetizione;

        IF stagioneV IS NULL THEN
                RAISE NOTICE 'La squadra in quella stagione non ha partecipato
                                alla competizione';
                RETURN NULL;
        ELSE
                RETURN NEW;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER vinc_trofeo_compet
        BEFORE INSERT ON vinceSquadra
        FOR EACH ROW
        EXECUTE FUNCTION vinc_trofeo_competS();
```

5.7.13 naz_part_squadraMF

Il seguente trigger controlla se le squadre che inseriamo in una competizione siano della loro stessa nazione o che appartengano al loro stesso continente nel caso di competizioni internazionali. Per fare ciò ci salviamo la nazionalità della competizione in una variabile e controlliamo se essa non è una tra Europa, Africa, Asia, Oceania, Nordamerica e SudAmerica. Se l'esito del controllo è vero, allora ci prendiamo la nazionalità della squadra e la confrontiamo con quella della competizione. In caso non fossero coincidenti notificheremo l'errore all'amministratore, altrimenti l'inserimento andrà a buon fine. Invece se la nazionalità della competizione risulterà essere un continente, salveremo non la nazionalità della squadra, ma il continente di cui quella nazione fa parte. Dopo di che la confrontiamo con la nazionalità della competizione e anche qui se il confronto risulterà non coincidere l'amministratore verrà allertato, viceversa l'inserimento andrà a buon fine

```
CREATE OR REPLACE FUNCTION naz_part_squadraMF()
RETURNS TRIGGER AS $$

DECLARE
nazSquadra varchar(100);
nazComp varchar(100);
BEGIN
        SELECT Nazionalità
        INTO nazComp
        FROM Competizione
        WHERE Nome = NEW.Competizione;

        IF nazComp NOT IN ('Europa','Africa','Asia','Oceania','NordAmerica','SudAmerica')
        THEN
                SELECT Nazionalità
                INTO nazSquadra
                FROM Squadra
                WHERE idSquadra = NEW.Squadra;
                IF nazComp = nazSquadra THEN
                        RETURN NEW;
                ELSE
                        RAISE NOTICE 'La squadra non fa parte della nazione
                                      della competizione'
                        RETURN NULL;
                END IF;
        ELSE
                SELECT Continente
                INTO nazSquadra
                FROM Squadra INNER JOIN Nazionalità ON Squadra.Nazionalità
                                                     = Nazionalità.Nome
```

```
            WHERE idSquadra = NEW.Squadra;
            IF nazComp = nazSquadra THEN
                RETURN NEW;
            ELSE
                RAISE NOTICE 'La squadra non fa parte del continente
                                della competizione;
                RETURN NULL;
            END IF;
        END IF;

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER naz_part_squadra
        BEFORE INSERT ON Partecipa
        FOR EACH ROW
        EXECUTE FUNCTION naz_part_squadraMF();
```

5.7.14 controllo_nazG

Il seguente trigger controlla che le squadre di club non partecipino alle competizioni per nazionali e viceversa. Per fare questo controlliamo a monte se la competizione è per nazionali. Se il controllo da esito positivo allora ci salviamo in una variabile il nome della nazionale e lo confrontiamo con tutte le nazionali presenti nella tabella nazionalità. Se dovessimo trovarla allora l'inserimento andrà a buon fine perché vorrà dire che stiamo inserendo una nazione in una competizione per nazioni, altrimenti notifichiamo all'amministratore che sta tentando di mettere una squadra di club in una competizioni per nazioni. Dopo di che, se il controllo a monte dovesse dare esito negativo, controlleremo sempre se il nome della squadra è quella di una nazione e se dovesse darci esito positivo notificheremo all'amministratore l'errore di star volendo inserire una nazionale in una competizione per club. In caso contrario invece l'inserimento verrà fatto senza problemi.

```
CREATE OR REPLACE FUNCTION controllo_nazG()
RETURNS TRIGGER AS $$

DECLARE
NomeSquadra varchar(100);
BEGIN
        IF NEW.Competizione IN ('Mondiale','Europeo','Coppa D"Africa',
                                'Coppa D"Asia','Coppa America','Coppa D"Oceania')
       THEN
                SELECT Nome
                INTO NomeSquadra
                FROM Squadra
                WHERE idSquadra = NEW.Squadra;
                IF NomeSquadra IN (SELECT Nome FROM Nazionalità) THEN
                        RETURN NEW;
                ELSE
                        RAISE NOTICE 'Stai inserendo una squadra di club in una
                                      competizioni per nazioni';
                        RETURN NULL;
                END IF;
        ELSIF NomeSquadra IN (SELECT Nome FROM Nazionalità) THEN
                RAISE NOTICE 'Stai inserendo una nazione in una competizione
                              per club';
                RETURN NULL;
        ELSE
                RETURN NEW;
                END IF;
        END IF;
END;
```

```
$$ language plpgsql

CREATE OR REPLACE TRIGGER controllo_naz
        BEFORE INSERT ON Partecipa
        FOR EACH ROW
        EXECUTE FUNCTION controllo_nazG();
```

5.7.15 agg_ruoloP

Il seguente trigger fa sì che, quando inseriamo in Ha un calciatore con il ruolo di portiere, tutte le righe della tabella militanza che coinvolgono quel calciatore vedranno aggiornarsi la colonna golSubiti che da NULL avrà valore 0

```
CREATE OR REPLACE FUNCTION agg_ruoloP()
RETURNS TRIGGER AS $$

DECLARE

BEGIN
        UPDATE Militanza
        SET golSubiti = '0'
        WHERE NEW.Calciatore = Calciatore;
        RETURN NEW;
END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER agg_ruolo
        AFTER INSERT ON Ha
        FOR EACH ROW
        WHEN (NEW.Ruolo = 'Portiere')
        EXECUTE FUNCTION agg_ruoloP();
```

5.7.16 agg_data_ritiro

Il seguente trigger controlla se la data di ritiro che stiamo aggiungendo ad un calciatore non vada in conflitto con le date del suo contratto in Militanza. Per fare ciò ci creiamo un cursorse da cui salveremo in due variabili la data di inizio di tutti i contratti del giocatore in questione e le date di fine contratto. Una volta fatto ciò le confronteremo con la data di ritiro inserito e se ci saranno incongurenze l'operazione di aggiornamento verrà annullato e l'amministratore verrà informato del problema, altrimenti l'aggiornamento andrà a buon fine.

```
CREATE OR REPLACE FUNCTION agg_data_ritiro()
RETURNS TRIGGER AS $$

DECLARE
date_contratto CURSOR FOR
SELECT dataInizio, dataFine
FROM Militanza
WHERE Calciatore = NEW.idCalciatore;
data_inizio Militanza.dataInizio%TYPE;
data_fine Militanza.dataFine%TYPE;
BEGIN
        OPEN date_contratto;
        LOOP
                FETCH date_contratto INTO data_inizio, data_fine;
                IF (NEW.dataRitiro < data_inizio OR data_fine IS NOT NULL
                    AND NEWdataRitiro < data_fine) THEN
                        RAISE NOTICE 'Data in conflitto con le date del contratto';
                        RETURN NULL;
                ELSE
                        RETURN NEW;
                END IF;
                EXIT WHEN NOT FOUND;
         END LOOP;
         CLOSE date_contratto

END;

$$ language plpgsql

CREATE OR REPLACE TRIGGER aggiunta_data_ritiro
        BEFORE UPDATE ON Calciatore
        FOR EACH ROW
        WHEN (NEW.dataRitiro IS NOT NULL)
        EXECUTE FUNCTION agg_data_ritiro();
```