DIPARTIMENTO DI INFORMATICA

Corso di Laurea — Basi di Dati I

Progettazione e Sviluppo di un Database 

Docente	**Autori**	**Matricola**
Silvio Barra	Vittorio Emanuele Testa	N86/4823
	Antonio Vincenti	N86/4545
	Stefano Testa	N86/4871

Anno Accademico 2023/2024

Contents

1 Progettazione Concettuale

1.1 Analisi dei requisiti

La classe **Giocatore** serve a tenere traccia di tutti i giocatori. Pertanto, conterrà le informazioni anagrafiche (*Nome, Cognome, SSN, Nazionalità, Data di Nascita, Altezza, Peso e Sesso*) e le informazioni relative al ruolo, l'abilità e il piede principale del giocatore.

Nazionalità dovrà essere riferita ad una Nazione esistente.

Peso viene espresso in kg.

Altezza viene espressa in cm.

Ruolo deve riferirsi ad un ruolo esistente.

Abilità deve essere relativa ad una abilità valida nel mondo del calcio.

Dobbiamo tenere traccia delle militanze del giocatore in ogni squadra e della sua carriera. Per far ciò, abbiamo deciso di utilizzare la classe **Militanza** che si occuperà di gestire ogni Militanza del giocatore e, quando si vorrà ottenere informazioni sulla carriera in modo generale, ciò sarà ricavabile dalla stessa classe.

La data_Inizio deve essere antecedente alla data_Fine.

L'attributo *Goal_Subiti* può essere assegnato solo ai Portieri.

L'attributo *Sesso* può essere solo "M" o "F" rispettivamente.

Ogni giocatore può far parte di una **Squadra** per un determinato periodo di tempo, quindi occorre salvare le squadre.

Presenta gli attributi *NomeSquadra, Nazionalità, Data_Fondazione*.

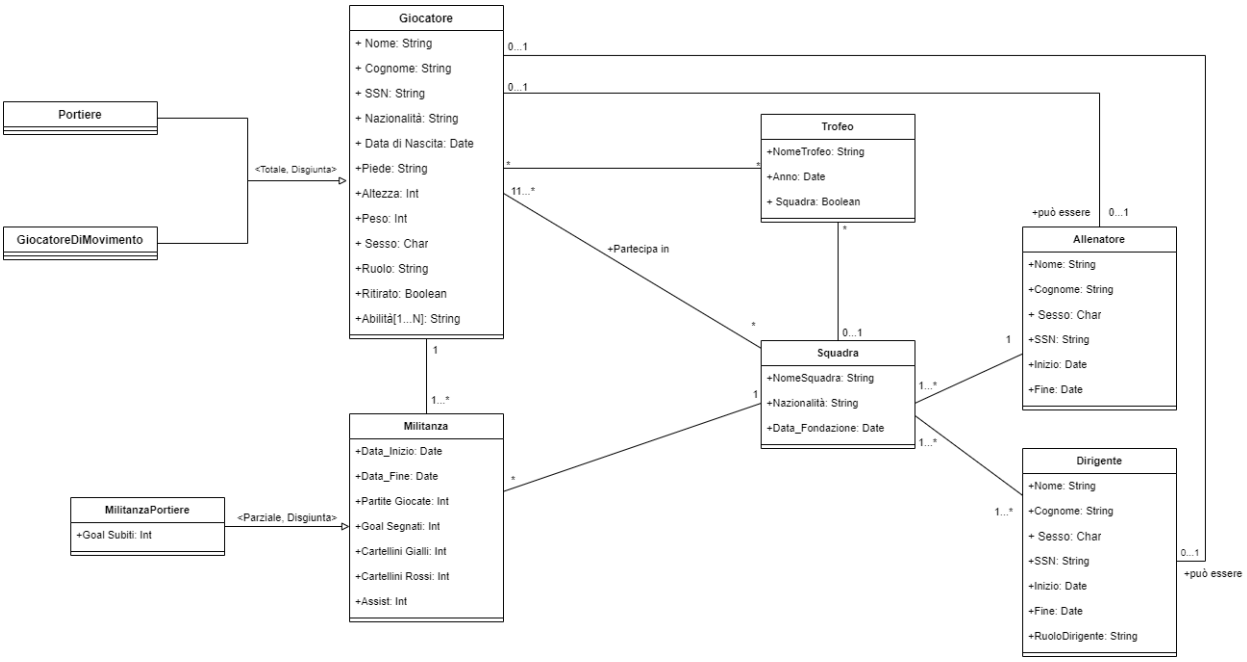
Nazionalità avrà lo stesso dominio della *Nazionalità* del **Giocatore**.

Può succedere che uno o più Giocatori vincano trofei (in base al tipo di trofeo, individuale o di squadra). Per questo motivo è necessario avere informazioni sui trofei nel Database. La classe **Trofeo**(*NomeTrofeo, Anno*) ci tiene quindi traccia di tutti i trofei presenti nel nostro mini-mondo.

Un Giocatore può ritirarsi. Una volta ritirato, egli può decidere se diventare un Allenatore, un Dirigente oppure andare in pensione. Nel caso in cui diventi un **Allenatore**, si terrà traccia di alcuni dati anagrafici già inseriti in precedenza del Giocatore (*Nome, Cognome, SSN*) e della data in cui questo passaggio è avvenuto.

Similmente accadrà se il Giocatore decide di diventare un **Dirigente**.Tuttavia, un Dirigente può avere diversi ruoli all'interno di una società, pertanto viene specificato con l'attributo *RuoloDirigente*.

1.2 Schema Concettuale



1.3 Dizionario delle entità e delle associazioni

Entità	Descrizione	Attributi
Giocatore	Informazioni generali univoche ad ogni giocatore.	**Nome**(String): Nome del Giocatore. **Cognome**(String): Cognome del Giocatore **SSN**(String): Social Serial Number identificativo e univoco ad ogni giocatore **Nazionalità**(String): Nazione di nascita del giocatore **Data di Nascita**(Date): Anno, Mese e Giorno di nascita del giocatore **Piede**(String): Piede dominante del Giocatore **Altezza**(Int): Altezza del giocatore espresso in cm **Peso**(Int): Peso del giocatore espresso in kg **Sesso**(Char) : Sesso del giocatore **Ruolo**(String): Ruolo del giocatore **Ritirato**(Boolean): Stato della carriera del giocatore. TRUE indica che è ritirato, FALSE indica che è ancora in Carriera. **Abilità**(String): Una o più tecniche speciali per le quali il Giocatore è conosciuto.
Portiere	Specializzazione di Giocatore. Rappresenta tutti i giocatori di ruolo portiere	
Giocatore Di Movimento	Specializzazione di Giocatore. Rappresenta tutti i giocatori che non sono portieri	

Entità	**Descrizione**	**Attributi**
Militanza	Dati della carriera di un giocatore	**Data_Inizio**(Date): Data di inizio di una militanza in una squadra **Data_Fine**(Date): Data di fine di una militanza in una squadra **Partite Giocate**(Int): Numero di partite giocate **Goal Segnati**(Int): Numero di goal fatti **Cartellini Gialli**(Int): Numero di cartellini gialli presi **Cartellini Rossi**(Int): Numero di cartellini rossi presi **Assist**(Int): Numero di assist fatti
Militanza Portiere	Specializzazione di Militanza. Rappresenta tutte le militanze dei portieri	**Goal Subiti**(Int): Numero di goal subiti da un portiere
Trofeo	Premi che i giocatori e le squadre possono ottenere	**NomeTrofeo**(String): Nome del trofeo **Anno**(Date): Anno in cui è stato creato il trofeo **Squadra**(Boolean): Sarà *TRUE* se il trofeo è di Squadra, *FALSE* altrimenti.
Squadra	Dati riguardanti una squadra	**NomeSquadra**(String): Nome della squadra **Nazionalità**(String): Nazionalità della squadra **Data_Fondazione**(Date): Data della fondazione della squadra

Entità	Descrizione	Attributi
Allenatore	Dati anagrafici e della carriera di un Allenatore	**Nome**(String): Nome dell'allenatore **Cognome**(String): Cognome dell'allenatore **Sesso**(Char): Sesso del Giocatore **SSN**(String): Social Serial Number dell'allenatore **Inizio**(Date): Indica quando un allenatore inizia la sua carriera **Fine**(Date): Indica quando un allenatore termina la sua carriera
Dirigente	Dati anagrafici e della carriera di un Dirigente	**Nome**(String): Nome del dirigente **Cognome**(String): Cognome del dirigente **Sesso**(Char): Sesso dell'Allenatore **SSN**(String): Social Serial Number del dirigente **Inizio**(Date): Data dell'inizio della carriera di un dirigente **Fine**(Date): Data che indica quando un dirigente si ritira **RuoloDirigente**(String): Ruolo che ha un dirigente in una squadra

Associazioni	**Descrizione**
Avere	Associazione **uno-a-molti** tra Giocatore e Militanza. Un Giocatore ha almeno una o più Militanze, ma la singola Militanza si riferirà soltanto ad un singolo Giocatore.
Essere	Associazione **uno-a-uno** tra Giocatore e Allenatore. Un Giocatore può decidere se essere un Allenatore, ed un Allenatore può esser stato Giocatore in passato.
Diventare	Associazione **uno-a-uno** tra Giocatore e Allenatore. Un Giocatore può decidere di diventare un Dirigente, ed un Dirigente può esser diventato tale, anche se in precedenza era un Giocatore.
Riferire	Associazione **uno-a-molti** tra Squadra e Militanza. Una Militanza si riferisce ad una sola Squadra ma ad una Squadra possono riferirsi 0, 1, o più Militanze.
Allenare	Associazione **uno-a-uno** tra Allenatore e Squadra. Una Squadra viene allenata da un solo Allenatore. Un Allenatore allena una sola Squadra.
Dirigere	Associazione **uno-a-molti** tra Squadra e Dirigente. Una Squadra viene diretta da almeno un Dirigente "Capo". Tuttavia, esistono altri Dirigenti che si occupano di gestire tutto ciò che riguarda il lato amministrativo. Un Dirigente dirige solo una Squadra.
Vincere	Associazione **molti-a-molti** tra Giocatore e Trofeo. Un Giocatore può vincere uno o più trofei. Un Trofeo può essere vinto da uno o più Giocatori (in caso esso sia di Squadra).
Partecipare	Associazione **molti-a-molti** tra Giocatore e Squadra. Un Giocatore può partecipare in una o più Squadre. In una Squadra devono esserci almeno 11 o più Giocatori.
Ottenuto Da	Associazione **uno-a-molti** tra Trofeo e Squadra. Una Squadra può ottenere uno o più Trofei. Un Trofeo può essere ottenuto da una sola Squadra in quel determinato anno.

2 Ristrutturazione del modello Concettuale

2.1 Analisi delle Ridondanze

Quando un **Giocatore** si ritira e decide di diventare Allenatore o Dirigente, abbiamo deciso di non eliminare la tupla che si riferisce a quel Giocatore. Ciò crea una ridondanza, ma ci dà un vantaggio. Per salvare le informazioni richieste dalla traccia di un ex Giocatore, accediamo in modo diretto al suo record, evitando l'uso di query potenzialmente complesse.

2.2 Eliminazione degli attributi multivalore

L'attributo multivalore che abbiamo utilizzato nello schema concettuale è *Abilità*. Per gestire questo attributo, li convertiamo in una singola stringa dove ci possono essere 1 o più valori dello stesso tipo.

2.3 Analisi delle Generalizzazioni

Generalizzazioni:

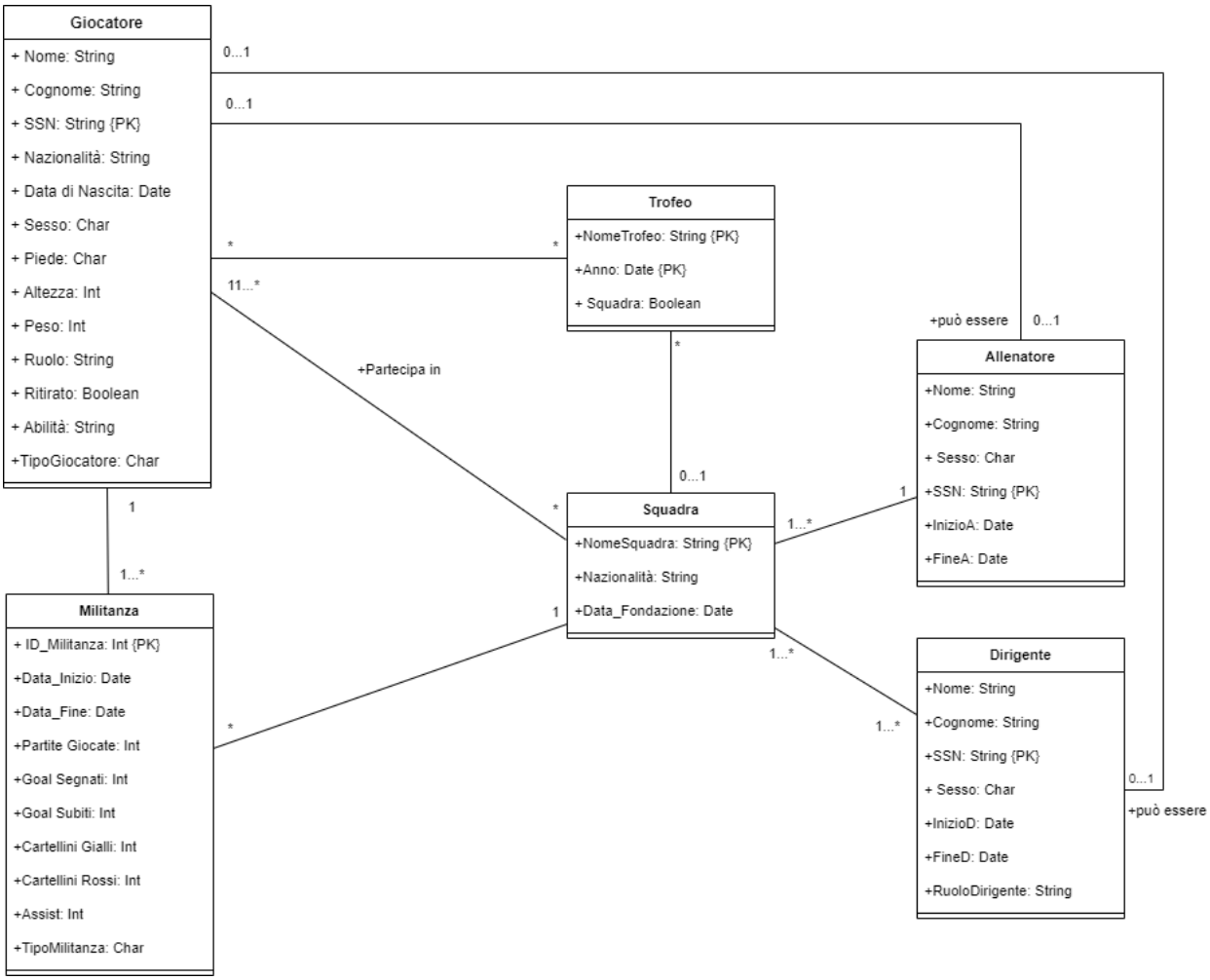
Giocatore — *Portiere*, *GiocatoreDiMovimento*

Militanza — *MilitanzaPortiere*

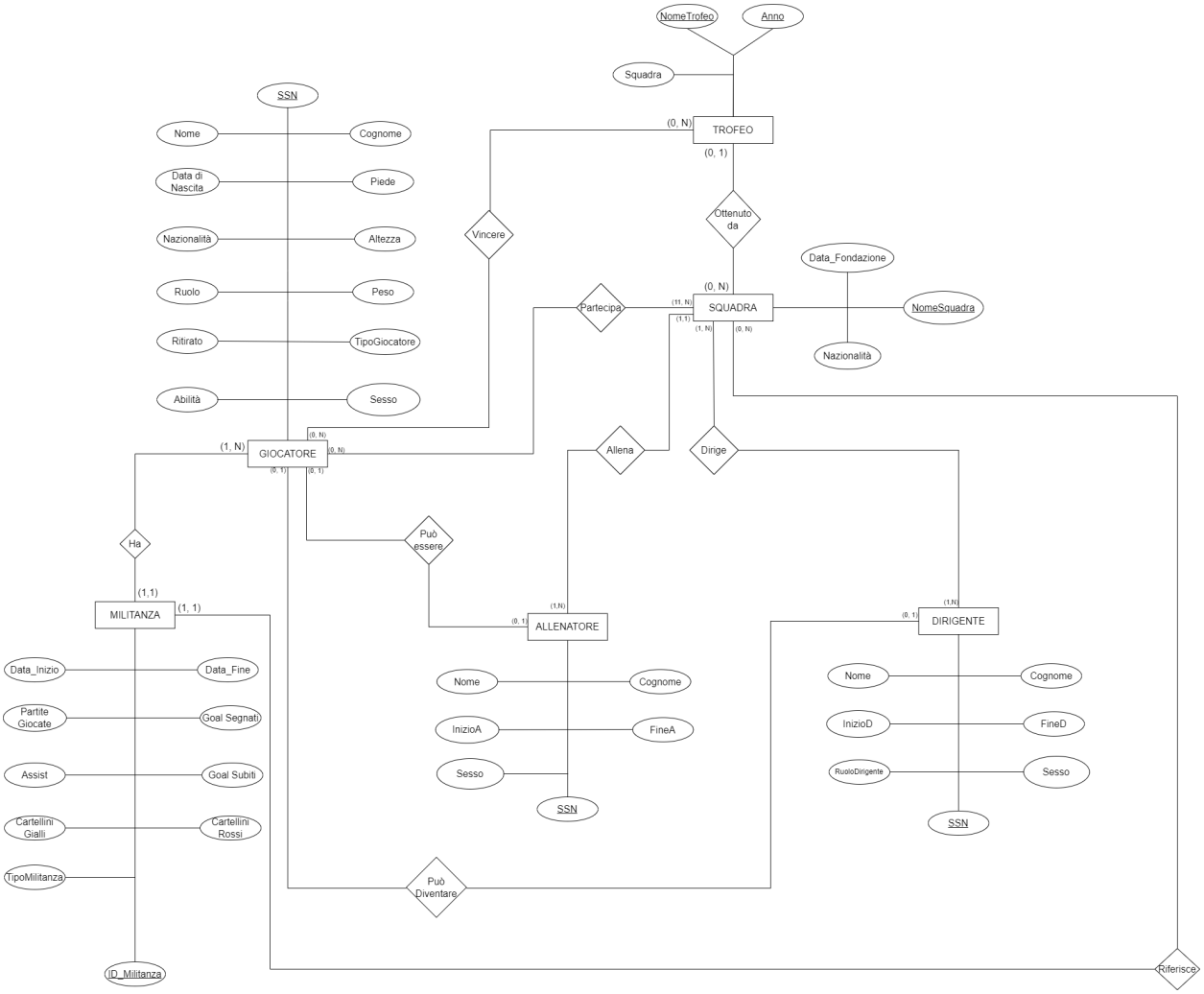
2.4 Identificazione Chiavi Primarie

2.5 Class Diagram Ristrutturato

2.5.1 UML Diagram



2.5.2 ER Diagram



2.6 Dizionario delle classi

Classe	Attributi	Descrizione
Giocatore	Nome Cognome SSN Nazionalità Data di Nascita Piede Altezza Peso Sesso Ruoli Ritirato Abilità TipoGiocatore	Classe che conserva le informazioni relative ai Giocatori
Militanza	ID_Militanza Data_Inizio Data_Fine Partite Giocate Goal Segnati Goal Subiti Cartellini Gialli Cartellini Rossi Assist TipoMilitanza	Classe che conserva le informazioni relative alla militanza del giocatore.
Squadra	NomeSquadra Nazionalità Data_Fondazione	Classe che conserva le informazioni della Squadra.
Trofeo	NomeTrofeo Anno Squadra	Classe che distingue i vari trofei basandosi sull'anno ai quali si riferiscono.
Allenatore	Nome Cognome Sesso SSN InizioA FineA	Classe che specifica i dati anagrafici dell'allenatore e della sua carriera.
Dirigente	Nome Cognome Sesso SSN InizioD FineD	Classe che specifica i dati anagrafici del Dirigente e della sua carriera.

2.7 Dizionario delle associazioni

Associazione	Classi coinvolte	Descrizione
Ha ... appartiene ad un	Giocatore e Militanza	Un giocatore può avere 0, 1 o più militanze Ogni Militanza appartiene ad un singolo Giocatore.. [1 , *]
Può essere... poteva essere	Giocatore e Allenatore	Un giocatore potrebbe diventare un allenatore. Un allenatore poteva essere un giocatore in passato. [0...1, 0...1]
Può diventare ... poteva essere	Giocatore e Dirigente	Un giocatore potrebbe diventare un dirigente. Un dirigente poteva essere un giocatore in passato. [0...1, 0...1]
Vincere ... è vinto da	Giocatore e Trofeo	Un giocatore può vincere molti trofei. Un trofeo se è di squadra può essere vinto da molti giocatori. [0...*, 0...*]
Partecipa ... è formata da	Giocatore e Squadra	Un giocatore può partecipare in più squadre durante la sua carriera. In una squadra partecipano minimo 11 giocatori. [0...*, 11...*]
Ottenere ... è ottenuto da	Trofeo e Squadra	Un trofeo può essere ottenuto da una squadra. Una squadra può ottenere più trofei. [0...1, 0...*]
Allena ... è allenata da	Squadra e Allenatore	Una squadra può essere allenata solo da un allenatore. Un allenatore può allenare più squadre in momenti diversi. [1, 1...*]
Dirige ... è diretta da	Squadra e Dirigente	Una squadra può essere diretta da più dirigenti. Un dirigente può dirigere più squadre in momenti diversi. [1...*, 1...*]
Riferisce ... riferita ad	Militanza e Squadra	Una squadra può avere più militanze di diversi giocatori. Una militanza data un giocatore si riferisce ad una sola squadra. [0...*, 1]

3 Traduzione al modello logico

3.1 Mapping Associazioni

3.1.1 Associazioni 1 - N

- *Trofeo - Ottenuto Da - Squadra* : *Trofeo* ha una partecipazione parziale, si procede come N-N
- *Allenatore - Allena - Squadra* : Inserimento chiave di *Allenatore* in *Squadra* come chiave esterna.
- *Giocatore - Ha - Militanza* : Inserimento chiave di *Giocatore* in *Militanza* come chiave esterna.
- *Squadra - Riferisce - Militanza* : Inserimento chiave di *Squadra* in *Militanza* come chiave esterna.

3.1.2 Associazioni N - N

Per queste relazioni, le chiavi delle due relazioni si inseriscono in una nuova relazione come chiavi esterne.

Associazione	**Relazione**	**Associazione**
Giocatore	*Vincere*	Trofeo
Giocatore	*Partecipa in*	Squadra
Squadra	*Ottenere*	Trofeo
Dirigente	*Dirige*	Squadra

3.2 Modello Logico

Gli attributi sottolineati sono chiavi primarie
Gli attributi che terminano con un * alla fine sono chiavi esterne

Giocatore (Nome, Cognome, SSN, Nazionalità*, Data_di_Nascita,
Piede, Altezza, Peso, Sesso, Ruoli, Ritirato,
Abilita, TipoGiocatore)
Nazionalità $\mapsto$ Nazione.Nome

Militanza (ID_Militanza, Data_Inizio, Data_Fine, Partite_Giocate, Goal_Segnati, Goal_Subiti, Cartellini_Gialli, Cartellini_Rossi, Assist, TipoMilitanza, NomeSquadra*, SSN*)
NomeSquadra $\mapsto$ Squadra.Nome_Squadra
SSN $\mapsto$ Giocatore.SSN

Trofeo (NomeTrofeo, Anno, Squadra)

Squadra (Nome_Squadra, Nazionalità*, Data_Fondazione, SSN_Allenatore*)
SSN_Allenatore $\mapsto$ Allenatore.SSN
Nazionalità $\mapsto$ Nazione.Nome

Vincere (SSN*, NomeTrofeo*, Anno*)
SSN $\mapsto$ Giocatore.SSN
NomeTrofeo $\mapsto$ Trofeo.NomeTrofeo
Anno $\mapsto$ Trofeo.Anno

Partecipa in (SSN, NomeSquadra*)
NomeSquadra $\mapsto$ Squadra.Nome_Squadra

Ottenuto Da (NomeSquadra*, NomeTrofeo*, Anno*)
NomeSquadra $\mapsto$ Squadra.Nome_Squadra
NomeTrofeo $\mapsto$ Trofeo.NomeTrofeo
Anno $\mapsto$ Trofeo.Anno

Allenatore (SSN, Nome, Cognome, Sesso, InizioA, FineA)

Dirigente (Nome, Cognome, Sesso, SSN, InizioD, FineD, RuoloDirigente)

Dirige (SSN_Dirigente*, NomeSquadra*)
NomeSquadra $\mapsto$ Squadra.Nome_Squadra
SSN_Dirigente $\mapsto$ Dirigente.SSN

Nazione (Nome)

Credenziali (Username, Password, Ruolo, SSN)

4 Progettazione Fisica

4.1 Creazione Database

L'amministratore esegue il comando

```
1 CREATE DATABASE CalciatoriDB;
```

Per dar vita al Database sul quale giocatori, allenatori e dirigenti si connetteranno in seguito (Per tal motivo, non è stato imposto un preciso limite alle connessioni possibili).

4.2 Gestione Ruoli e Permessi

Nel CalciatoriDB, ci sono vari ruoli che vanno specificati:

Ruolo	**Permessi**
Utente	Vedere le caratteristiche dei giocatori e richiedere filtri per attributo
Giocatore	Modificare i suoi dati anagrafici
Amministratore	Ricerca, Modifica, Creazione, Cancellazione Giocatori

```
2 CREATE ROLE Utente LOGIN PASSWORD ' ';
3 CREATE ROLE Giocatore LOGIN PASSWORD 'giocatore';
4 CREATE ROLE Amministratore LOGIN PASSWORD 'Amministratore';
5
6 GRANT SELECT ON TABLE Giocatore, Militanza TO Utente;
7 GRANT SELECT, UPDATE ON TABLE Giocatore TO Giocatore;
8 GRANT SELECT ON TABLE Credenziali TO Giocatore,
    Amministratore;
9 GRANT SELECT, DELETE, INSERT, UPDATE ON TABLE Giocatore TO
    Amministratore;
```

4.3 Creazione Domini

Prima di creare le tabelle, vengono creati alcuni domini in grado di regolare cosa può essere inserito nelle tabelle del Database:

```
10 CREATE DOMAIN eta_valida AS DATE CHECK (VALUE <= CURRENT_DATE
     - INTERVAL '16 years');
```

I giocatori non possono possedere età inferiore a 16 anni, questo dominio si occupa di controllare che l'età venga rispettata.

Ecco il ragionamento:

CURRENT_DATE rappresenta la data corrente. Ad essa viene sottratto il valore "16 anni" (In questo caso, il valore risultante sarebbe '2008-x-x');

Se la data che si sta inserendo è $>$ di questa data risultante, allora l'età non è corretta. Se invece essa è minore oppure uguale a questa data, allora la data che si sta inserendo è valida.

E.g.

'2006-x-x' $<$ '2008-x-x' $\mapsto$ Data corretta;

'2009-x-x' $>$ '2008-x-x' $\mapsto$ Data non corretta;

```
11 CREATE DOMAIN ssn_valido AS VARCHAR(11) CHECK (VALUE SIMILAR
     TO '[0-9]{3}-[0-9]{3}-[0-9]{3}');
```

L'ssn deve rispettare un certo formato:

$'XXX - XXX - XXX'$ dove $X \in \{0 \dots 9\}$

Quindi:

'012-231-219' $\mapsto$ accettato;

'91a-131-928' $\mapsto$ rifiutato;

```
12 CREATE DOMAIN RuoloType AS VARCHAR(30) CHECK (VALUE IN ('
     Portiere', 'Difensore centrale', 'Terzino sinistro', '
     Terzino destro', 'Centrocampista difensivo', '
     Centrocampista centrale', 'Esterno sinistro', 'Esterno
     destro', 'Centrocampista offensivo', 'Ala sinistra', 'Ala
     destra', 'Trequartista', 'Punta'));
```

Il Giocatore può avere uno di questi ruoli.

```
13 CREATE DOMAIN intero_non_negativo AS INTEGER CHECK (VALUE >=
     0);
```

Tutti gli interi che si possono inserire nel Database non assumono mai valore negativo, di conseguenza abbiamo aggiunto questo dominio per evitare di dover creare un elevato numero di constraint che controllano tutti la stessa cosa.

```
14 CREATE DOMAIN sesso_corretto AS CHAR CHECK (VALUE = 'F' OR
     VALUE = 'M');

15 CREATE DOMAIN ruoloDirigente AS VARCHAR(30)
16   CHECK(VALUE IN ('Presidente', 'Amministratore delegato',
     'Dirigente sportivo', 'Dirigente tecnico', 'Preparatore
     atletico', 'Preparatore dei portieri', 'Medico sportivo',
     'Mental coach', 'Fisioterapista', 'Osservatore', '
     Magazziniere') OR VALUE IS NULL);
```

Il Dirigente può assumere uno dei ruoli specificati in questo dominio, oppure nessuno.

```
17 CREATE DOMAIN ruoli_ammessi AS VARCHAR(15) CHECK (VALUE IN ('
     Giocatore','Amministratore'));
```

Solo i Giocatori oppure gli Amministratori hanno credenziali.

4.4 Creazione Tabelle

```
18 CREATE TABLE Nazione(
19 Nome VARCHAR(50),
20 PRIMARY KEY (Nome));
```

Più che essere una vera e propria relazione, questa tabella serve puramente per rendere il Dominio delle nazionalità meno tedioso da rappresentare.

Qualsiasi nazione coinvolta nel Database, ha bisogno di essere inserita qui prima. Così facendo, in base alle necessità, le nazioni possono essere inserite o anche rimosse se non sono più necessarie.

Ciò rappresenta oltre che un vantaggio a livello di convenienza rappresentativa, anche un vantaggio implementativo. Anziché controllare ad ogni inserimento che la nazione sia corretta, le uniche nazioni che si possono inserire sono direttamente quelle che rispettano il vincolo di chiave esterna su questa tabella.

```
21 CREATE TABLE Giocatore (
22 Nome VARCHAR(50) NOT NULL,
23 Cognome VARCHAR(50) NOT NULL,
24 SSN ssn_valido NOT NULL,
25 Nazionalit  VARCHAR(50) REFERENCES Nazione(Nome) NOT NULL,
26 DataDiNascita eta_valida NOT NULL,
27 Sesso sesso_corretto NOT NULL,
28 Piede CHARACTER(1) NOT NULL,
29 Altezza NUMERIC(3,0) NOT NULL,
30 Peso NUMERIC(4,0) NOT NULL,
31 Ruoli RuoloType NOT NULL,
32 Ritirato BOOLEAN NOT NULL DEFAULT FALSE,
33 Abilit  VARCHAR(50) NOT NULL,
34 TipoGiocatore CHARACTER(1) NOT NULL,
35
36 PRIMARY KEY (SSN),
37
38 CONSTRAINT AltezzaCorretta CHECK(Altezza > 1.50),
39 CONSTRAINT PesoCorretto CHECK(Peso > 40.00),
40 CONSTRAINT PiedeCorretto CHECK(Piede = 'D' or Piede = 'S'),
```

```
41 CONSTRAINT TipoGiocatoreCorretto CHECK(TipoGiocatore= 'M' or
     TipoGiocatore= 'P')
42 CONSTRAINT tipo_giocatore_corretto CHECK ((Ruolo = 'Portiere'
      AND TipoGiocatore = 'P') OR (Ruolo <> 'Portiere' AND
     TipoGiocatore = 'M'))
43 );

44 CREATE TABLE Militanza(
45 ID_Militanza SERIAL NOT NULL,
46 Data_Inizio DATE NOT NULL,
47 Data_Fine DATE,
48 PartiteGiocate intero_non_negativo NOT NULL DEFAULT 0,
49 GoalSegnati intero_non_negativo NOT NULL DEFAULT 0,
50 GoalSubiti intero_non_negativo NOT NULL DEFAULT 0,
51 CartelliniGialli intero_non_negativo NOT NULL DEFAULT 0,
52 CartelliniRossi intero_non_negativo NOT NULL DEFAULT 0,
53 Assist intero_non_negativo NOT NULL DEFAULT 0,
54 TipoMilitanza CHAR(1) NOT NULL,
55
56 PRIMARY KEY (ID_Militanza),
57
58 CONSTRAINT TipoMilitanzacorretto CHECK(TipoMilitanza ='M' or
     TipoMilitanza ='P')
59 CONSTRAINT contr_data CHECK (Data_Inizio < Data_Fine)
60 CONSTRAINT correttoGoal_Subiti CHECK ((Goal_Subiti = 0 AND
     TipoMilitanza = 'M') OR (Goal_Subiti >= 0 AND
     TipoMilitanza = 'P'));

61 CREATE TABLE Squadra(
62 Nome_Squadra VARCHAR(50) UNIQUE NOT NULL,
63 Nazionalita VARCHAR(50) REFERENCES Nazione(Nome) NOT NULL,
64 Data_Fondazione DATE NOT NULL,
65
66 PRIMARY KEY (Nome_Squadra));

67 CREATE TABLE Trofeo(
68 NomeTrofeo VARCHAR(30) NOT NULL,
69 Anno DATE NOT NULL,
70 Squadra BOOLEAN NOT NULL DEFAULT FALSE,
71
72 PRIMARY KEY (NomeTrofeo, Anno));
```

```
73 CREATE TABLE Allenatore(
74 Nome VARCHAR(50) NOT NULL,
75 Cognome VARCHAR(50) NOT NULL,
76 Sesso sesso_corretto NOT NULL,
77 SSN ssn_valido NOT NULL,
78 InizioA DATE NOT NULL,
79 FineA DATE,
80
81 PRIMARY KEY (SSN),
82
83 CONSTRAINT contr_data CHECK (InizioA < FineA)),
84
85 CONSTRAINT SuperSSN FOREIGN KEY (SSN) REFERENCES Giocatore(
       SSN)
86 ON UPDATE RESTRICT
87 ON DELETE RESTRICT);

88 CREATE TABLE Dirigente(
89 Nome VARCHAR(50) NOT NULL,
90 Cognome VARCHAR(50) NOT NULL,
91 Sesso sesso_corretto NOT NULL,
92 SSN ssn_valido NOT NULL,
93 RuoloDirigente ruoloDirigente NOT NULL,
94 InizioD DATE NOT NULL,
95 FineD DATE,
96
97 PRIMARY KEY (SSN),
98 CONSTRAINT contr_data CHECK (InizioD < FineD));

99 CREATE TABLE Vincere(
100 SSN ssn_valido REFERENCES Giocatore(SSN) NOT NULL,
101 NomeTrofeo VARCHAR(30) NOT NULL,
102 Anno DATE NOT NULL,
103
104 CONSTRAINT TrofeoFK FOREIGN KEY (NomeTrofeo, Anno) REFERENCES
       Trofeo(NomeTrofeo, Anno));

105 CREATE TABLE Partecipa_In(
106 SSN ssn_valido REFERENCES Giocatore(SSN) NOT NULL,
107 NomeSquadra VARCHAR(50) REFERENCES Squadra(Nome_Squadra) NOT
       NULL);
```

```
108 CREATE TABLE Ottenuto_Da(
109 NomeSquadra VARCHAR(50) REFERENCES Squadra(Nome_Squadra) NOT
      NULL,
110 NomeTrofeo VARCHAR(30) NOT NULL,
111 Anno DATE NOT NULL,
112 CONSTRAINT TrofeoFK FOREIGN KEY (NomeTrofeo, Anno) REFERENCES
       Trofeo(NomeTrofeo, Anno));

113 CREATE TABLE Dirige(
114 SSN_Dirigente ssn_valido REFERENCES Dirigente(SSN) NOT NULL,
115 NomeSquadra VARCHAR(50) REFERENCES Squadra(Nome_Squadra) NOT
      NULL);

116 CREATE TABLE Credenziali(
117     Username VARCHAR(20) UNIQUE NOT NULL
118         CHECK (LENGTH(Username) >= 6)
119         CONSTRAINT lunghezza_username CHECK (LENGTH(Username)
      >= 6),
120
121     Password VARCHAR(15) NOT NULL
122         CHECK (
123             LENGTH(Password) >= 5 AND
124             Password ~ '[A-Z]' AND
125             Password ~ '[0-9]'
126         )
127         CONSTRAINT complessita_password CHECK (
128             LENGTH(Password) >= 5 AND
129             Password ~ '[A-Z]' AND
130             Password ~ '[0-9]'
131         ),
132
133     Ruolo ruoli_ammessi NOT NULL,
134     SSN ssn_valido NOT NULL,
135     PRIMARY KEY(SSN)
136 );
```

4.5 Trigger e Trigger Function

Quando si inserisce una tupla in Squadra e aggiungiamo anche l'Allenatore (NEW.SSN_Allenatore IS NOT NULL), controlliamo che l'allenatore esista e Poi che non si sia ritirato dalla sua

carriera di allenatore:

```
137 CREATE OR REPLACE FUNCTION CheckInsertOnA()
138 RETURNS TRIGGER AS $$
139 DECLARE
140 CheckR Allenatore%ROWTYPE;
141 BEGIN
142   SELECT * INTO CheckR
143   FROM Allenatore
144   WHERE SSN = NEW.SSN_Allenatore;
145
146   IF FOUND AND CheckR.FineA IS NOT NULL THEN
147     RAISE EXCEPTION 'Impossibile inserire! SSN_Allenatore ha
      terminato la sua carriera';
148   END IF;
149
150   IF NOT FOUND THEN
151     RAISE EXCEPTION 'Si sta provando ad inserire un
      SSN_Allenatore inesistente';
152   END IF;
153
154   RETURN NEW;
155 END
156 $$ Language plpgsql;
157
158 CREATE TRIGGER CheckInsertOnA
159 BEFORE INSERT ON Squadra
160 FOR EACH ROW
161 WHEN (NEW.SSN_Allenatore IS NOT NULL)
162 EXECUTE FUNCTION CheckInsertOnA();
```

Quando si aggiorna una tupla di Squadra provando a cambiare l'allenatore della squadra (NEW.SSN_Allenatore <> OLD.SSN_Allenatore), si controlla che il nuovo allenatore inserito sia esistente e non abbia finito la sua carriera da allenatore:

```
163 CREATE OR REPLACE FUNCTION CheckANotRetiredBUpdate()
164 RETURNS TRIGGER AS $$
165 DECLARE
166 Controllo Allenatore%ROWTYPE;
167 BEGIN
168   SELECT * INTO Controllo
169   FROM Allenatore
170   WHERE SSN = NEW.SSN_Allenatore;
171
172   IF FOUND AND Controllo.FineA IS NOT NULL THEN
173     RAISE EXCEPTION 'Allenatore inserito ha terminato la sua
      carriera, non pu  allenare la squadra';
174   END IF;
175
176   IF NOT FOUND THEN
177     RAISE EXCEPTION 'Si sta provando ad inserire
      SSN_Allenatore inesistente!';
178   END IF;
179
180   RETURN NEW;
181 END;
182 $$ Language plpgsql
183
184 CREATE TRIGGER CheckANotRetiredBUpdate
185 BEFORE UPDATE ON Squadra
186 FOR EACH ROW
187 WHEN (NEW.SSN_Allenatore <> OLD.SSN_Allenatore)
188 EXECUTE FUNCTION CheckANotRetiredBUpdate();
```

Questo trigger si attiva prima di inserire una tupla in Vincere,controlliamo prima di tutto se il giocatore partecipa ancora in quella squadra, controlla se il trofeo è di squadra, in caso lo fosse devo controllare se la sua squadra ha vinto quel trofeo (in caso la squadra non ha vinto quel trofeo ma il trofeo è di squadra allora c'è un errore), in caso il trofeo non è di squadra allora lo si aggiunge normalmente:

```
189 CREATE OR REPLACE FUNCTION CheckInsV()
190 RETURNS TRIGGER AS $$
191 DECLARE
192     isSquadra BOOLEAN;
193     SquadraApp RECORD;
194     ControlloDate RECORD;
195 BEGIN
196     SELECT * INTO SquadraApp
197     FROM Partecipa_in
198     WHERE SSN = NEW.SSN;
199
200
201     IF SquadraApp IS NULL THEN
202
203       RAISE EXCEPTION 'Giocatore non partecipa a nessuna
    squadra, impossibile inserire';
204
205     END IF;
206
207     SELECT * INTO ControlloDate
208     FROM Militanza
209     WHERE SSN = NEW.SSN AND NomeSquadra = SquadraApp.
    NomeSquadra AND Data_Fine IS NULL;
210
211   IF ControlloDate.Data_Inizio <= NEW.Anno THEN
212
213       SELECT Squadra INTO isSquadra
214       FROM Trofeo
215       WHERE NomeTrofeo = NEW.NomeTrofeo AND Anno = NEW.Anno;
216
217       -- Se    vero che il trofeo    di squadra, controllo se
      esiste la situazione dove il trofeo    in ottenuto_da. Se
      ci   non fosse vero
218       -- allora sto tentando di inserire il trofeo ad un
```

```
    singolo giocatore, e non perch  il trofeo di squadra
    viene dato a tutti i giocatori.
219
220       IF isSquadra THEN
221
222      IF NOT EXISTS (
223          SELECT 1
224          FROM Ottenuto_da
225          WHERE NomeTrofeo = NEW.NomeTrofeo AND Anno = NEW.Anno
     AND NomeSquadra = SquadraApp.NomeSquadra
226      ) THEN
227          RAISE EXCEPTION 'Il Trofeo    di squadra. Impossibile
     assegnarlo al singolo giocatore';
228      END IF;
229
230       END IF;
231
232       RETURN NEW;
233
234   END IF;
235
236      RETURN OLD;
237 END;
238 $$ LANGUAGE plpgsql;
239
240 -- Creazione del trigger separatamente dalla definizione
    della funzione
241 CREATE TRIGGER CheckInsV
242 BEFORE INSERT ON Vincere
243 FOR EACH ROW
244 EXECUTE FUNCTION CheckInsV();
```

Quando facciamo l'update di un dirigente e gli mettiamo la data di fine carriera <> NULL se esiste una sua tupla in Dirige allora deve essere eliminata dato che il dirigente ha finito la sua carriera da dirigente (forse si può migliorare il codice mettendo l'attivazione del trigger WHEN New.FineD IS NOT NULL):

```
245 CREATE OR REPLACE FUNCTION DelDirAfterUpOnD()
246 RETURNS TRIGGER AS $$
247 BEGIN
248     -- Verifica se FineD non    nullo
249     IF NEW.FineD IS NOT NULL THEN
250         -- Esegue la cancellazione dalla tabella Dirige
251         DELETE FROM Dirige WHERE SSN_Dirigente = NEW.SSN;
252     END IF;
253     RETURN NEW;
254 END;
255 $$ LANGUAGE plpgsql;
256
257 CREATE TRIGGER del_dir_after_update_on_d
258 AFTER UPDATE ON Dirigente
259 FOR EACH ROW
260 EXECUTE FUNCTION DelDirAfterUpOnD();
```

Quando si aggiorna la tupla di un allenatore mettendo una data di fine carriera <> NULL allora si controlla se esiste una squadra che ha come allenatore il nostro allenatore e si cambia l'attributo in NULL:

```
261 CREATE OR REPLACE FUNCTION DelOnSFromUpOnAFunction()
262 RETURNS TRIGGER AS $$
263 DECLARE
264     Check CURSOR FOR SELECT Nome_Squadra FROM Squadra;
265     SquadraRecord RECORD;
266 BEGIN
267     OPEN Check;
268
269     LOOP
270         FETCH Check INTO SquadraRecord;
271         EXIT WHEN NOT FOUND;
272
273         UPDATE Squadra
274         SET SSN_Allenatore = NULL
275         WHERE Nome_Squadra = SquadraRecord.Nome_Squadra AND
      SSN_Allenatore = NEW.SSN;
```

```
276     END LOOP;
277
278     CLOSE Check;
279
280     RETURN NEW;
281 END;
282 $$ LANGUAGE plpgsql;
283
284 CREATE TRIGGER DelOnSFromUpOnA
285 AFTER UPDATE ON Allenatore
286 FOR EACH ROW
287 WHEN (NEW.FineA IS NOT NULL)
288 EXECUTE FUNCTION DelOnSFromUpOnAFunction();
```

Quando inseriamo una tupla in Ottenuto_Da vediamo prima di tutto se il trofeo aggiunto è un trofeo di squadra (in caso non lo è non può essere aggiunto), una volta accertati che sia un trofeo di squadra inseriamo nella tabella Vincere tutti i giocatori che stanno giocando in quel momento nella squadra (controllando con Partecipa_In) con il nome e l'anno del trofeo:

```
289 CREATE OR REPLACE FUNCTION InsVFromOdA()
290 RETURNS TRIGGER AS $$
291 DECLARE
292     Controllo Trofeo%ROWTYPE;
293     Giocatore CURSOR FOR
294         SELECT SSN
295         FROM Partecipa_In
296         WHERE NomeSquadra = NEW.NomeSquadra;
297     Scorr RECORD;
298 BEGIN
299     SELECT * INTO Controllo
300     FROM TROFEO
301     WHERE NomeTrofeo = NEW.NomeTrofeo AND Anno = NEW.Anno;
302
303     IF Controllo.Squadra IS NULL THEN
304         RAISE EXCEPTION 'Il trofeo non    di squadra,
    impossibile inserire';
305     END IF;
306
307     OPEN Giocatore;
308
309     LOOP
```

```
310         FETCH Giocatore INTO Scorr;
311         EXIT WHEN NOT FOUND;
312
313         INSERT INTO Vincere VALUES(Scorr.SSN, NEW.NomeTrofeo,
      NEW.Anno);
314     END LOOP;
315
316     CLOSE Giocatore;
317
318     RETURN NEW;
319 END;
320 $$ LANGUAGE plpgsql;
321
322 -- Creazione del trigger separatamente dalla definizione
      della funzione
323 CREATE TRIGGER InsVFromOdA
324 AFTER INSERT ON Ottenuto_Da
325 FOR EACH ROW
326 EXECUTE FUNCTION InsVFromOdA();
```

Prima di inserire una tupla in dirige dobbiamo controllare che il dirigente non si sia ritirato:

```
327 CREATE OR REPLACE FUNCTION CheckDIsNotRetired()
328 RETURNS TRIGGER AS $$
329 BEGIN
330   IF((SELECT FineD
331       FROM Dirigente
332       WHERE SSN = New.SSN_Dirigente) IS NOT NULL) THEN
333     RAISE EXCEPTION 'Il Dirigente non pu  essere inserito
      perch  si    ritirato';
334   END IF;
335   RETURN NEW;
336 END;
337 $$ Language plpgsql;
338
339 CREATE TRIGGER check_d_is_not_retired
340 BEFORE INSERT ON Dirige
341 FOR EACH ROW
342 EXECUTE FUNCTION CheckDIsNotRetired();
```

Quando inseriamo un allenatore controlliamo se in passato è stato un giocatore, se era un giocatore controlliamo che nella tabella Giocatore l'attributo ritirato sia TRUE, in caso contrario viene aggiornato l'attributo ed impostato a TRUE:

```
343 -- allenatore devo controllare che i suoi valori vengano
      rispettati. Stessa cosa da fare se faccio update sul
      giocatore. Se elimino
344 -- un giocatore allora deve essere CASCADE
345
346 CREATE OR REPLACE FUNCTION ExGiocatoreA()
347 RETURNS TRIGGER AS $$
348 DECLARE
349     Controllo RECORD;
350 BEGIN
351     SELECT * INTO Controllo
352     FROM Giocatore
353     WHERE SSN = NEW.SSN;
354
355     IF Controllo IS NOT NULL THEN
356         IF Controllo.Nome <> NEW.Nome OR Controllo.Cognome <>
       NEW.Cognome OR Controllo.Sesso <> NEW.Sesso THEN
357             RAISE EXCEPTION 'I dati non coincidono! Si prega
      di inserire i valori corretti.';
358         END IF;
359
360         UPDATE Giocatore
361         SET Ritirato = TRUE
362         WHERE SSN = NEW.SSN AND Ritirato = FALSE;
363     END IF;
364
365     RETURN NEW;
366 END;
367 $$ LANGUAGE plpgsql;
368
369 CREATE TRIGGER ex_giocatoreA
370 AFTER INSERT ON Allenatore
371 FOR EACH ROW
372 EXECUTE FUNCTION ExGiocatoreA();
```

Quando inseriamo un dirigente controlliamo se in passato è stato un giocatore, se era un giocatore controlliamo che nella tabella Giocatore l'attributo ritirato sia TRUE, in caso contrario viene aggiornato l'attributo ed impostato a TRUE:

```
373 CREATE OR REPLACE FUNCTION ExGiocatoreD()
374 RETURNS TRIGGER AS $$
375 DECLARE
376     Controllo RECORD;
377 BEGIN
378     SELECT * INTO Controllo
379     FROM Giocatore
380     WHERE SSN = NEW.SSN;
381 
382     IF Controllo IS NOT NULL THEN
383         IF Controllo.Nome <> NEW.Nome OR Controllo.Cognome <>
      NEW.Cognome OR Controllo.Sesso <> NEW.Sesso THEN
384             RAISE EXCEPTION 'I dati non coincidono! Si prega
    di inserire i valori corretti.';
385         END IF;
386 
387         UPDATE Giocatore
388         SET Ritirato = TRUE
389         WHERE SSN = NEW.SSN AND Ritirato = FALSE;
390     END IF;
391 
392     RETURN NEW;
393 END;
394 $$ LANGUAGE plpgsql;
395 
396 CREATE TRIGGER ex_giocatoreD
397 AFTER INSERT ON Dirigente
398 FOR EACH ROW
399 EXECUTE FUNCTION ExGiocatoreD();
```

Quando inserisco una tupla in Partecipa_in controllo che il sesso del giocatore inserito sia lo stesso degli altri giocatori della squadra:

```
400 CREATE OR REPLACE FUNCTION CheckSexPInG()
401 RETURNS TRIGGER AS $$
402 DECLARE
403     sex_in CHAR(1);
404     sex_squadra CHAR(1);
405 BEGIN
406     SELECT Sesso INTO sex_in
407     FROM Giocatore
408     WHERE SSN = NEW.SSN;
409 
410     SELECT Sesso INTO sex_squadra
411     FROM Giocatore
412     WHERE SSN IN (
413         SELECT SSN
414         FROM Partecipa_in
415         WHERE SSN <> NEW.SSN AND NomeSquadra = NEW.
    NomeSquadra
416         LIMIT 1
417     );
418 
419     IF sex_in <> sex_squadra THEN
420         RAISE EXCEPTION 'Il giocatore inserito non    dello
    stesso sesso dei giocatori della squadra';
421     END IF;
422 
423     RETURN NEW;
424 END;
425 $$ LANGUAGE plpgsql;
426 
427 CREATE TRIGGER check_sex_pi_g
428 AFTER INSERT ON Partecipa_in
429 FOR EACH ROW
430 EXECUTE FUNCTION CheckSexPInG();
```

```
431 CREATE TABLE Dirige(
432 SSN_Dirigente ssn_valido REFERENCES Dirigente(SSN) NOT NULL,
433 NomeSquadra VARCHAR(50) REFERENCES Squadra(Nome_Squadra) NOT
    NULL);
```

Quando si inserisce una tupla in Partecipa_in si controlla prima se il giocatore sta giocando per quella squadra controllando la tabella Militanza, dopo si controlla se il giocatore non abbia già finito la sua militanza in quella squadra, se l'ha già finita allora non verrà aggiunto:

```
434 CREATE OR REPLACE FUNCTION DeleteFromInsOnPIn()
435 RETURNS TRIGGER AS $$
436 DECLARE
437     Giocatore RECORD;
438 BEGIN
439 
440     -- se ho due data_inizio che cominciano allo stesso
    momento, avrei un errore, ma quel caso non avverr  mai
    per il singolo giocatore.
441     SELECT * INTO Giocatore
442     FROM Militanza
443     WHERE NomeSquadra = NEW.NomeSquadra AND SSN = NEW.SSN
444     ORDER BY Data_Inizio DESC
445     LIMIT 1;
446 
447     -- Se la tupla non esiste in Militanza, allora si sta
    inserendo una tupla in PartecipaIn non corretta.
448     IF NOT FOUND THEN
449         RAISE EXCEPTION 'Giocatore non trovato per
    NomeSquadra = % e SSN = %', NEW.NomeSquadra, NEW.SSN;
450     END IF;
451 
452     IF Giocatore.Data_Fine IS NOT NULL THEN
453         RAISE EXCEPTION 'Il Giocatore ha terminato la sua
    militanza nella Squadra';
454     END IF;
455 
456     RETURN NEW;
457 END;
458 $$ LANGUAGE plpgsql;
459 
460 CREATE TRIGGER DeleteFromInsOnPIn
461 AFTER INSERT ON Partecipa_in
462 FOR EACH ROW
463 EXECUTE FUNCTION DeleteFromInsOnPIn();
```

Dopo aver eliminato la militanza di un giocatore in Militanza, se il giocatore la quale è stata eliminata la tupla in militanza aveva una tupla in Partecipa_in riguardante quella militanza allora deve essere eliminata:

```
464 CREATE OR REPLACE FUNCTION DeletePInFromDeleteOnM()
465 RETURNS TRIGGER AS $$
466 BEGIN
467     DECLARE
468         Controllo RECORD;
469     BEGIN
470         SELECT * INTO Controllo
471         FROM Partecipa_in AS P
472         WHERE P.SSN = OLD.SSN AND P.NomeSquadra = OLD.
    NomeSquadra;
473
474         IF Controllo.SSN IS NOT NULL THEN
475             DELETE FROM Partecipa_in
476             WHERE SSN = OLD.SSN AND NomeSquadra = OLD.
    NomeSquadra;
477         END IF;
478
479     END;
480     RETURN OLD;
481 END;
482 $$ LANGUAGE plpgsql;
483
484
485 CREATE TRIGGER DeletePInFromDeleteOnM
486 AFTER DELETE ON Militanza
487 FOR EACH ROW
488 EXECUTE FUNCTION DeletePInFromDeleteOnM();
```

Quando viene aggiornata la militanza di un giocatore inserendo una data fine alla sua militanza allora si deve eliminare la tupla di quella militanza in Partecipa_in:

```
489 CREATE OR REPLACE FUNCTION DeletePInFromUpdateOnM()
490 RETURNS TRIGGER AS $$
491 BEGIN
492     IF NEW.NomeSquadra <> OLD.NomeSquadra OR NEW.SSN <> OLD.
    SSN THEN
493         RAISE EXCEPTION 'Impossibile eseguire la richiesta.
    Inserire una nuova riga con i dati corretti per il
    giocatore';
494     END IF;
495
496     IF NEW.Data_Fine IS NULL AND OLD.Data_Fine IS NOT NULL
    THEN
497         RAISE EXCEPTION 'Non    possibile annullare la fine
    di una militanza una volta inserita';
498     END IF;
499
500     IF NEW.Data_Fine IS NOT NULL THEN
501         DELETE FROM Partecipa_in
502         WHERE SSN = NEW.SSN AND NomeSquadra = NEW.NomeSquadra
    ;
503     END IF;
504
505     RETURN NEW;
506 END;
507 $$ LANGUAGE plpgsql;
508
509
510 CREATE TRIGGER DeletePInFromUpdateOnM
511 AFTER UPDATE ON Militanza
512 FOR EACH ROW
513 EXECUTE FUNCTION DeletePInFromUpdateOnM();
```

Quando viene inserita una tupla in Militanza si controlla che il giocatore non si sia ritirato e che non abbia un'altra militanza ancora attiva (Data_Fine <> NULL), se queste condizioni si attivano allora si può inserire il giocatore nella tabella Partecipa_in:

```
514 -- la logica segue una serie di step: se il giocatore
    ritirato, la militanza non si pu  inserire.
    Successivamente, controllo che
515 -- non esista una militanza con una data_fine ancora a null.
    Se ci     vero, allora non posso inserire un'altra
    militanza.
516 -- controllato anche questo, inserisco inoltre la mia
    militanza in partecipa_in
517
518 CREATE OR REPLACE FUNCTION InsertPInFromInsOnM()
519 RETURNS TRIGGER AS $$
520 DECLARE
521     NoDouble CURSOR FOR
522         SELECT Data_Fine
523         FROM Militanza
524         WHERE SSN = NEW.SSN AND Data_Fine IS NULL AND
    ID_Militanza <> NEW.ID_Militanza
525         ORDER BY Data_Inizio DESC
526         LIMIT 1;
527     Controllo DATE;
528     Rit BOOLEAN;
529 BEGIN
530     SELECT Ritirato INTO Rit
531     FROM Giocatore
532     WHERE SSN = NEW.SSN;
533
534     IF Rit THEN
535         RAISE EXCEPTION 'Il Giocatore    ritirato,
    impossibile inserire la Militanza';
536     END IF;
537
538     IF NEW.Data_Fine IS NULL THEN
539         OPEN NoDouble;
540
541         FETCH NoDouble INTO Controllo;
542
```

```
543         IF NOT FOUND THEN
544             CLOSE NoDouble;
545             INSERT INTO Partecipa_in(SSN, NomeSquadra) VALUES
      (NEW.SSN, NEW.NomeSquadra);
546         ELSE
547             CLOSE NoDouble;
548             RAISE EXCEPTION 'Militanza gi  in corso per il
    giocatore che    stato inserito.';
549         END IF;
550     END IF;
551
552     RETURN NEW;
553 END;
554 $$ LANGUAGE plpgsql;
555
556 -- Creazione del trigger separatamente dalla definizione
      della funzione
557 CREATE TRIGGER InsertPInFromInsOnM
558 AFTER INSERT ON Militanza
559 FOR EACH ROW
560 EXECUTE FUNCTION InsertPInFromInsOnM();
```

Questo trigger blocca le modifiche delle tuple di Partecipa_in:

```
561 CREATE OR REPLACE FUNCTION NoChangesUpdatePIn()
562 RETURNS TRIGGER AS $$
563 BEGIN
564
565     RAISE EXCEPTION 'Non    possibile modificare i valori
    inseriti';
566
567     RETURN NEW;
568
569 END;
570 $$ LANGUAGE plpgsql;
571
572 CREATE TRIGGER NoChangesUpdatePIn
573 BEFORE UPDATE ON Partecipa_in
574 FOR EACH ROW
575 WHEN (OLD.SSN IS DISTINCT FROM NEW.SSN OR OLD.NomeSquadra IS
      DISTINCT FROM NEW.NomeSquadra)
```

```
576 EXECUTE FUNCTION NoChangesUpdatePIn();
```

Quando si elimina una tupla da Partecipa_in allora si aggiorna la Data_Fine di quella carriera in Militanza:

```
577 CREATE OR REPLACE FUNCTION UpdateMFromDelOnPIn()
578 RETURNS TRIGGER AS $$
579 DECLARE
580     tmp Militanza%ROWTYPE;
581 BEGIN
582     -- Utilizzare direttamente la variabile tmp senza aprire
        il cursore
583     SELECT *
584     INTO tmp
585     FROM Militanza
586     WHERE NomeSquadra = OLD.NomeSquadra AND SSN = OLD.SSN
587     ORDER BY Data_Inizio DESC
588     LIMIT 1;
589
590     IF tmp.Data_Fine IS NULL THEN
591         UPDATE Militanza
592         SET Data_Fine = CURRENT_DATE
593         WHERE ID_Militanza = tmp.ID_Militanza;
594     END IF;
595
596     RETURN OLD;
597 END;
598 $$ LANGUAGE plpgsql;
599
600
601 -- Creazione del trigger separatamente dalla definizione
        della funzione
602 CREATE TRIGGER UpdateMFromDelOnPIn
603 BEFORE DELETE ON Partecipa_in
604 EXECUTE FUNCTION UpdateMFromDelOnPIn();
```

Quando si aggiorna la tupla di un Giocatore inserendo il valore TRUE al giocatore, in caso il giocatore aveva una militanza attiva (Data_Fine IS NULL) allora si mette la CURRENT_DATE a quella militanza per indicarne la fine:

```
605 CREATE OR REPLACE FUNCTION UpdateMwhenGretired()
606 RETURNS TRIGGER AS $$
607 BEGIN
608       UPDATE Militanza
609       SET Data_Fine = CURRENT_DATE
610       WHERE SSN = NEW.SSN AND Data_Fine IS NULL;
611     RETURN NEW;
612 END;
613 $$ LANGUAGE plpgsql;
614
615
616 CREATE TRIGGER UpdateMwhenGretired
617 AFTER UPDATE ON Giocatore
618 FOR EACH ROW
619 WHEN (NEW.Ritirato)
620 EXECUTE FUNCTION UpdateMwhenGretired();
```

Prima di inserire una tupla in Militanza dobbiamo controllare che la data di inizio non si trovi nel mezzo di una data di un'altra militanza:

```
621 CREATE OR REPLACE FUNCTION ControlloDataInsMBtw()
622 RETURNS TRIGGER AS $$
623 BEGIN
624   IF EXISTS (
625     SELECT 1
626     FROM Militanza
627     WHERE New.SSN = SSN
628       AND New.Data_Inizio BETWEEN Data_Inizio AND Data_Fine
629   ) THEN
630     RAISE EXCEPTION 'La militanza non pu  essere inserita
      perch  il giocatore in quella data ha gi  un''altra
      militanza';
631   END IF;
632   RETURN NEW;
633 END;
634 $$ LANGUAGE plpgsql;
635
636 CREATE TRIGGER controllo_ins_Data_Militanza_btw
```

```
637 BEFORE INSERT ON Militanza
638 FOR EACH ROW
639 EXECUTE FUNCTION ControlloDataInsMBtw();
```

Prima di modificare la data di inzio di una tupla in Militanza dobbiamo controllare che la data di inizio non si trovi nel mezzo di una data di un'altra militanza:

```
640 CREATE OR REPLACE FUNCTION ControlloDataUpdMBtw()
641 RETURNS TRIGGER AS $$
642 BEGIN
643   IF EXISTS (
644     SELECT 1
645     FROM Militanza
646     WHERE New.SSN = SSN
647       AND New.Data_Inizio BETWEEN Data_Inizio AND Data_Fine
648   ) THEN
649     RAISE EXCEPTION 'La militanza non pu  essere inserita
      perch  il giocatore in quella data ha gi  un''altra
      militanza';
650   END IF;
651   RETURN NEW;
652 END;
653 $$ LANGUAGE plpgsql;
654
655 CREATE TRIGGER controllo_upd_Data_Militanza_btw
656 BEFORE UPDATE ON Militanza
657 FOR EACH ROW
658 WHEN (New.Data_Inizio <> Old.Data_Inizio OR New.Data_Fine <>
      OLD.Data_Fine)
659 EXECUTE FUNCTION ControlloDataUpdMBtw();
```

Prima di inserire una tupla in Militanza dobbiamo controllare che la data di inizio non si trovi nel mezzo di una data di un'altra militanza:

```
660 CREATE OR REPLACE FUNCTION ControlloDataInsMBtw()
661 RETURNS TRIGGER AS $$
662 BEGIN
663   IF EXISTS (
664     SELECT 1
665     FROM Militanza
666     WHERE New.SSN = SSN
667       AND New.Data_Inizio BETWEEN Data_Inizio AND Data_Fine
668   ) THEN
669     RAISE EXCEPTION 'La militanza non pu  essere inserita
     perch  il giocatore in quella data ha gi  un''altra
     militanza';
670   END IF;
671   RETURN NEW;
672 END;
673 $$ LANGUAGE plpgsql;
674
675 CREATE TRIGGER controllo_ins_Data_Militanza_btw
676 BEFORE INSERT ON Militanza
677 FOR EACH ROW
678 EXECUTE FUNCTION ControlloDataInsMBtw();
```

Prima di modificare la data di inizio di una tupla in Militanza dobbiamo controllare che la data di inizio non si trovi nel mezzo di una data di un'altra militanza:

```
679 CREATE OR REPLACE FUNCTION ControlloDataUpdMBtw()
680 RETURNS TRIGGER AS $$
681 BEGIN
682   IF EXISTS (
683     SELECT 1
684     FROM Militanza
685     WHERE New.SSN = SSN
686       AND New.Data_Inizio BETWEEN Data_Inizio AND Data_Fine
687   ) THEN
688     RAISE EXCEPTION 'La militanza non pu  essere inserita
     perch  il giocatore in quella data ha gi  un''altra
     militanza';
689   END IF;
690   RETURN NEW;
```

```
691 END;
692 $$ LANGUAGE plpgsql;
693
694 CREATE TRIGGER controllo_upd_Data_Militanza_btw
695 BEFORE UPDATE ON Militanza
696 FOR EACH ROW
697 WHEN (New.Data_Inizio <> Old.Data_Inizio OR New.Data_Fine <>
      OLD.Data_Fine)
698 EXECUTE FUNCTION ControlloDataUpdMBtw();
```

Se si vuole modificare uno dei seguenti attributi di Militanza: Partite_Giocate, Goal_Seganti, Goal_Subiti, Cartellini_Gialli, Cartellini_Rossi, Assist, si deve controllare che la militanza non sia finita:

```
699 CREATE OR REPLACE FUNCTION militanza_finitaUpd()
700 RETURNS TRIGGER AS $$
701 BEGIN
702   IF ((SELECT Data_Fine
703       FROM Militanza
704       WHERE ID_Militanza = New.ID_Militanza) IS NOT NULL)
     THEN
705     RAISE EXCEPTION 'Non si pu  modificare una militanza che
         terminata';
706   END IF;
707   RETURN NEW;
708 END;
709 $$ Language plpgsql;
710
711
712 CREATE TRIGGER militanza_finita
713 BEFORE UPDATE OF Partite_Giocate, Goal_Seganti, Goal_Subiti,
     Cartellini_Gialli, Cartellini_Rossi, Assist ON Militanza
714 FOR EACH ROW
715 EXECUTE FUNCTION militanza_finitaUpd();
```

Quando inserisco una tupla in militanza devo controllare che se il giocatore è un portiere allora il tipo di Militanza deve essere uguale a P, altrimenti deve essere M:

```
716 CREATE OR REPLACE FUNCTION Controllo_tipo_militanza()
717 RETURNS TRIGGER AS $$
718 BEGIN
719   IF(SELECT TipoGiocatore
```

```
720       FROM Giocatore
721       WHERE SSN = New.SSN) = 'M' THEN --Si attiva se il
      giocatore    di movimento
722     IF New.TipoMilitanza <> 'M' THEN
723       RAISE EXCEPTION 'Il TipoMilitanza = P ma TipoGiocatore
      = M';
724     END IF;
725   ELSE --Si attiva se il giocatore    un portiere
726     IF New.TipoMilitanza <> 'P' THEN
727       RAISE EXCEPTION 'Il TipoMilitanza = M ma TipoGiocatore
      = P';
728     END IF;
729   END IF;
730   RETURN NEW;
731 END;
732 $$ Language plpgsql;
733
734 CREATE TRIGGER controllo_tipo_militanza
735 BEFORE INSERT OR UPDATE ON Militanza
736 FOR EACH ROW
737 EXECUTE FUNCTION Controllo_tipo_militanza();
```

Prima di inserire una tupla in Ottenuto_Da un trofeo dobbiamo controllare che la squadra che ha vinto il trofeo ha almeno 11 giocatori nella squadra:

```
738 CREATE OR REPLACE FUNCTION CheckSquadra11()
739 RETURNS TRIGGER AS $$
740 DECLARE
741 Count INTEGER;
742 BEGIN
743     SELECT COUNT(*) INTO Count
744     FROM Partecipa_In
745     WHERE NomeSquadra = NEW.NomeSquadra;
746
747     IF(Count < 11) THEN
748
749   RAISE EXCEPTION 'I giocatori devono essere almeno 11 nella
      squadra!';
750
751     END IF;
752
```

```
753     RETURN NEW;
754 END;
755 $$ LANGUAGE plpgsql;
756
757 -- Creazione del trigger separatamente dalla definizione
      della funzione
758 CREATE TRIGGER CheckSquadra11
759 BEFORE INSERT ON Ottenuto_da
760 FOR EACH ROW
761 EXECUTE FUNCTION CheckSquadra11();
```

Quando un giocatore si è ritirato (ritirato = true) il suo attributo non può più essere impostato a false:

```
762 CREATE OR REPLACE FUNCTION NoUpdateRetiredG()
763 RETURNS TRIGGER AS $$
764 BEGIN
765   IF (OLD.Ritirato) THEN
766
767     RAISE EXCEPTION 'Impossibile eseguire la modifica. Il
      Giocatore si    ritirato!';
768
769   END IF;
770     RETURN NEW;
771 END;
772 $$ LANGUAGE plpgsql;
773
774
775 CREATE TRIGGER NoUpdateRetiredG
776 BEFORE UPDATE ON Giocatore
777 FOR EACH ROW
778 WHEN (NEW.Ritirato = FALSE)
779 EXECUTE FUNCTION NoUpdateRetiredG();
```

Prima di inserire una tupla in Allenatore controlliamo che il suo ssn non è già presente in Dirigente:

```
780 CREATE OR REPLACE FUNCTION Check_ssn_All
781 RETURNS TRIGGER AS $$
782 BEGIN
783   IF EXISTS(SELECT 1
784       FROM Dirigente
785       WHERE SSN = New.SSN) THEN
786     RAISE EXCEPTION 'Non puoi aggiungere un allenatore che ha
      lo stesso SSN di un dirigente';
787   END IF;
788   RETURN NEW;
789 END;
790 $$ Language plpgsql;
791
792 CREATE TRIGGER check_ssn_All
793 BEFORE INSERT ON Allenatore
794 FOR EACH ROW
795 EXECUTE FUNCTION Check_ssn_All();
```

Prima di inserire una tupla in Dirigente controlliamo che il suo ssn non è già presente in Allenatore:

```
796 CREATE OR REPLACE FUNCTION Check_ssn_Dir
797 RETURNS TRIGGER AS $$
798 BEGIN
799   IF EXISTS(SELECT 1
800       FROM Allenatore
801       WHERE SSN = New.SSN) THEN
802     RAISE EXCEPTION 'Non puoi aggiungere un allenatore che ha
      lo stesso SSN di un dirigente';
803   END IF;
804   RETURN NEW;
805 END;
806 $$ Language plpgsql;
807
808 CREATE TRIGGER check_ssn_Dir
809 BEFORE INSERT ON Dirigente
810 FOR EACH ROW
811 EXECUTE FUNCTION Check_ssn_Dir();
```

Prima di inserire un trofeo, se già ne esiste un altro con lo stesso nome allora controlliamo che siano dello stesso tipo:

```
812 --un trofeo che compare pi  volte nella tabella trofeo deve
      sempre essere o di squadra o individuale
813 CREATE OR REPLACE FUNCTION Controllo_coerenza_trofeo()
814 RETURNS TRIGGER AS $$
815 DECLARE
816 isSquadra BOOLEAN;
817 BEGIN
818   --controllo prima l'esistenza del trofeo stesso nella
      tabella
819     SELECT Squadra INTO isSquadra
820     FROM Trofeo
821     WHERE NomeTrofeo = New.NomeTrofeo
822     LIMIT 1;
823
824     IF isSquadra is NOT NULL AND isSquadra <> New.Squadra
      THEN
825
826     RAISE EXCEPTION 'Trofeo % non del tipo giusto. Dovrebbe
      essere %', NEW.NomeTrofeo, isSquadra;
827
828     END IF;
829
830   RETURN NEW;
831 END;
832 $$ Language plpgsql;
833
834 CREATE TRIGGER controllo_coerenza_trofeo
835 BEFORE INSERT ON Trofeo
836 FOR EACH ROW
837 EXECUTE FUNCTION Controllo_coerenza_trofeo();
```

4.6 Procedure e Funzioni

```
839 CREATE OR REPLACE FUNCTION Carriera_Giocatore(SSN_Giocatore
      ssn_valido)
840 RETURNS TABLE(
841     Inizio_Carr Date,
842     Fine_Carr Date,
843     s_part_gioc intero_non_negativo,
844     s_goal_segn intero_non_negativo,
845     s_goal_sub intero_non_negativo,
846     s_cart_g intero_non_negativo,
847     s_cart_r intero_non_negativo,
848     s_assist intero_non_negativo
849 )
850 AS $$
851 DECLARE
852     Inizio_Carr Date;
853     Fine_Carr Date;
854     s_part_gioc intero_non_negativo;
855     s_goal_segn intero_non_negativo;
856     s_goal_sub intero_non_negativo;
857     s_cart_g intero_non_negativo;
858     s_cart_r intero_non_negativo;
859     s_assist intero_non_negativo;
860
861 BEGIN
862     SELECT Data_Inizio INTO Inizio_Carr
863     FROM Militanza
864     WHERE SSN = SSN_Giocatore
865     ORDER BY Data_Inizio ASC
866     LIMIT 1;
867
868     SELECT Sum(partitegiocate), Sum(goalsegnati),
869     Sum(goalsubiti), Sum(cartellinigialli), Sum(
      cartellinirossi), Sum(Assist)
870     INTO s_part_gioc, s_goal_segn, s_goal_sub, s_cart_g,
      s_cart_r, s_assist
871     FROM Militanza
872     WHERE SSN = SSN_Giocatore;
873
```

```
874     SELECT Data_Fine INTO Fine_Carr
875     FROM Militanza
876     WHERE SSN = SSN_Giocatore
877     ORDER BY Data_Fine DESC
878     LIMIT 1;
879
880     RETURN QUERY SELECT
881   Inizio_Carr,
882   Fine_Carr,
883   s_part_gioc,
884   s_goal_segn,
885   s_goal_sub,
886   s_cart_g,
887   s_cart_r,
888   s_assist;
889
890 END;
891 $$ LANGUAGE plpgsql;

892 CREATE OR REPLACE FUNCTION Curr_Dirigenti(SquadraInEsame TEXT
    )
893 RETURNS TABLE(SSN_Dirigente ssn_valido)
894 AS $$
895
896     BEGIN
897       IF( NOT EXISTS ( SELECT *
898            FROM Squadra
899            WHERE Nome_Squadra = SquadraInEsame)) THEN
900
901         RAISE EXCEPTION 'La squadra indicata non esiste';
902
903       END IF;
904
905       RETURN QUERY SELECT D.SSN_Dirigente FROM Dirige AS D
    WHERE D.NomeSquadra = SquadraInEsame;
906
907     END;
908 $$ Language plpgsql;
```

```
910 CREATE OR REPLACE FUNCTION Curr_Giocatori(SquadraInEsame TEXT
    )
911 RETURNS TABLE(SSN_Giocatore ssn_valido)
912 AS $$
913
914     BEGIN
915
916       IF( NOT EXISTS ( SELECT *
917            FROM Squadra
918            WHERE Nome_Squadra = SquadraInEsame)) THEN
919
920         RAISE EXCEPTION 'La squadra indicata non esiste';
921
922       END IF;
923
924       RETURN QUERY SELECT SSN FROM Partecipa_In WHERE Squadra
     = SquadraInEsame;
925
926     END;
927 $$ Language plpgsql;

928 CREATE OR REPLACE FUNCTION Storico_Giocatori(SquadraInEsame
    TEXT)
929 RETURNS TABLE(SSN_Giocatore ssn_valido)
930 AS $$
931
932     BEGIN
933
934       IF( NOT EXISTS ( SELECT *
935            FROM Squadra
936            WHERE Nome_Squadra = SquadraInEsame)) THEN
937
938         RAISE EXCEPTION 'La squadra indicata non esiste';
939
940       END IF;
941
942       RETURN QUERY SELECT DISTINCT SSN FROM Militanza WHERE
    NomeSquadra = SquadraInEsame;
943
944     END;
```

```
945 $$ Language plpgsql;

947 CREATE OR REPLACE FUNCTION Storico_Squadre(SSN_Input TEXT)
948 RETURNS TABLE(Squadra VARCHAR(50))
949 AS $$
950
951     BEGIN
952
953       IF( NOT EXISTS ( SELECT *
954            FROM Giocatore as G
955            WHERE G.SSN = SSN_Input)) THEN
956
957         RAISE EXCEPTION 'L''ssn indicato non esiste';
958
959       END IF;
960
961       RETURN QUERY SELECT DISTINCT M.NomeSquadra FROM
    Militanza AS M WHERE M.SSN = SSN_Input;
962     END;
963 $$ Language plpgsql;
```

4.7 Dizionario dei Vincoli

Tipologia	**Attributi Coinvolti**	**Descrizione**
Valori Nullable	Data_Fine FineA FineD	Questi attributi possono essere **NULL**. Tutti gli altri saranno **NOT NULL** oppure avranno valore predefinito.
Valori Default	Partite_Giocate Goal Segnati Goal Subiti Cartellini Gialli Cartellini Rossi Assist Ritirato Squadra	Questi attributi hanno come valore predefinito 0 tranne Ritirato e Squadra che hanno FALSE.

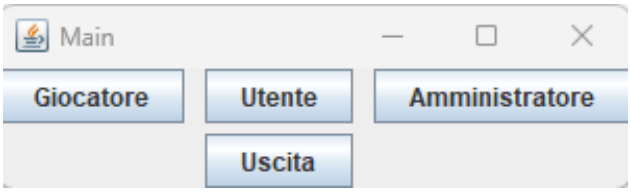
Tipologia	**Attributi Coinvolti**	**Descrizione**
Non Negativi	Altezza Peso Partite_Giocate Goal Segnati Goal Subiti Cartellini Gialli Cartellini Rossi Assist	Questi attributi non possono assumere valori negativi (Check)
Check_Età	Data di Nascita	Una check per controllare che l'età del giocatore sia corretta(Almeno 16 anni).
Check Integrità Date	Data_Inizio Date_Fine InizioA FineA InizioD FineD	Una check per evitare l'inserimento di una data fine che venga prima di una data inizio(semanticamente errato). In caso ciò accada, metterà Data_Fine a **NULL** in modo che chi sta inserendo la tupla si accorga dell'errore.
Domini Specifici	Nazionalità Ruoli RuoloDirigente Sesso	Questi attributi hanno un Dominio specifico che deve essere rispettato. Se viene inserito un valore che non appartiene al Dominio, la tupla non sarà inserita.

Nome	**Dominio**
Ruoli	Portiere Difensore Centrale Terzino Sinistro Terzino Destro Centrocampista Difensivo Centrocampista Centrale Esterno Sinistro Esterno Destro Centrocampista Offensivo Ala Sinistra Ala Destra Trequartista Punta

Nome	Dominio
RuoloDirigente	Presidente Amministratore delegato Dirigente sportivo Dirigente tecnico Preparatore atletico Preparatore dei portieri Medico sportivo Mental coach Fisioterapista Osservatore Magazziniere
Sesso	M F

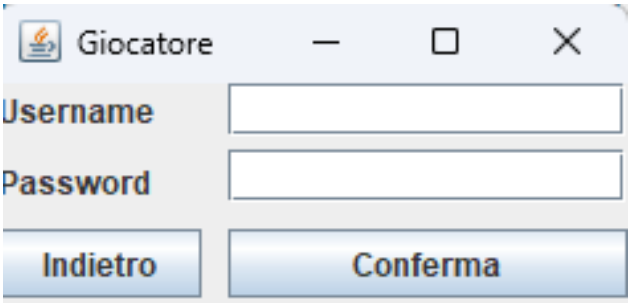
4.8 Esempio Applicativo

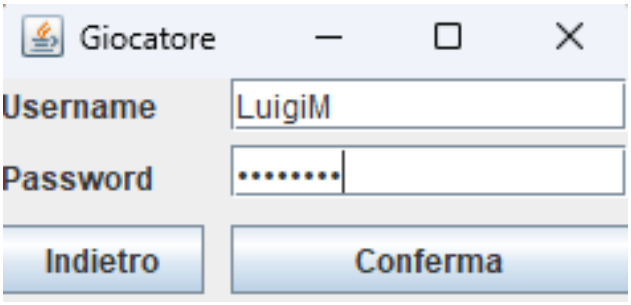
1. Apro l'applicativo java



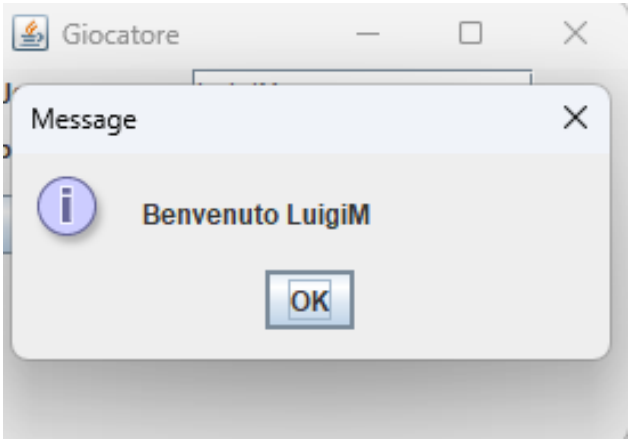
2. Voglio fare login come Giocatore:

3. Ho bisogno di inserire username e password presenti in credenziali(che abbiano ruolo giocatore, altrimenti errore)

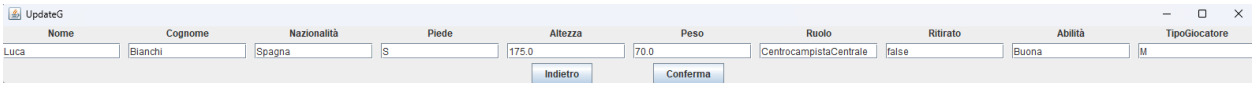




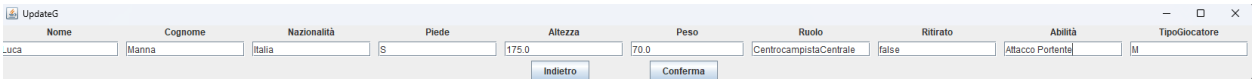
4. Se il login ha successo, comparirà un messaggio di conferma:



5. A questo punto sono entrato nel DB come quel Giocatore. Posso quindi modificare solo i suoi dati(Alcuni, ad esempio l'SSN, non sono modificabili)



6. Applico qualche leggera modifica



7. Messaggio di conferma UPDATE