UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Documentazione per Database "Esame_Traccia_3"

Giuseppe Testa

N86004843

Andrea Pignieri

N86004636

1 Progettazione concettuale

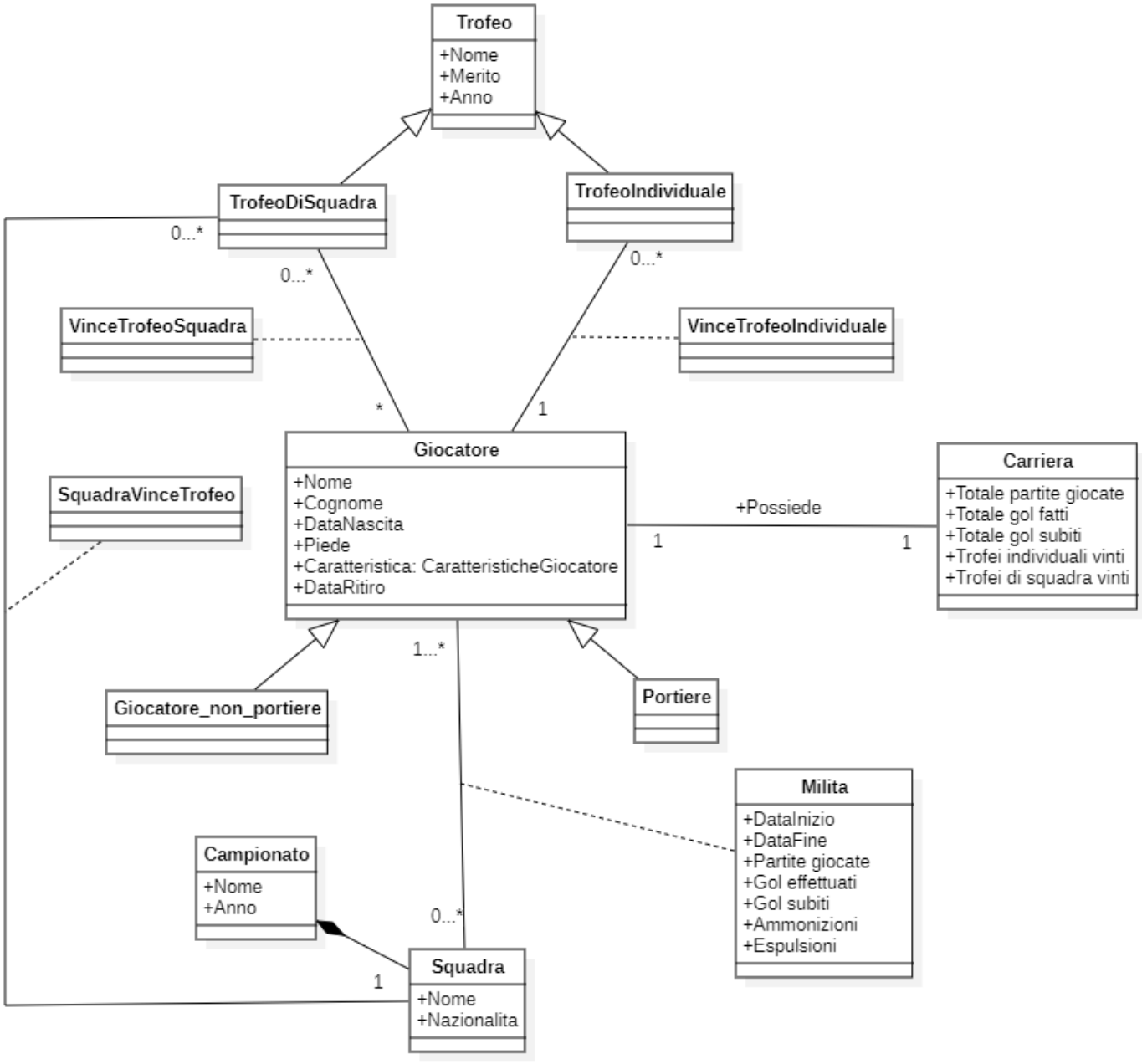
1.1 Analisi e specifica dei requisiti

Nella prima fase di analisi dei requisiti verranno identificate le informazioni fondamentali che porteranno allo sviluppo della struttura e delle funzionalità del database del cinema multisala. Durante il processo di analisi verranno individuate le diverse entità, le relazioni che abbiamo tra di esse ed eventuali vincoli e comportamenti del database.

Si sviluppi un sistema informativo per la gestione dei calciatori di tutti 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 inizio e 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 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.

1.2 Diagramma concettuale

Schema concettuale del database ottenuto dalle informazioni ricavate durante l'analisi dei requisiti.



Si consideri di voler partire dall'entità Giocatore, esso si specializza (totale, disgiunta) in Giocatore_non_portiere e Portiere, ora iniziamo a descrivere la relazione "Milita" che ha con l'entità Squadra; un giocatore può militare in nessuna (qui la relazione è parziale in quanto un giocatore può anche ritirarsi) o più squadre (in periodi di tempo differenti), e viceversa una squadra deve ospitare uno o più giocatori. L'attributo GolSubiti verrà valorizzato solo se il giocatore è un portiere (verrà risolto durante la fase successiva)

Successivamente un giocatore possiede una e una sola carriera, e viceversa una carriera è posseduta da uno e un solo giocatore, si noti che l'entità Carriera serve per avere un riassunto di tutte le statistiche del giocatore.

Un trofeo si specializza (totale, disgiunta) in Trofeo Individuale e Trofeo Di **Squadra.**

Un giocatore può vincere nessuno o più trofei individuali, e viceversa un trofeo individuale può essere vinto da uno e un solo giocatore.

Un giocatore può vincere nessuno o più trofei di squadra, e viceversa un trofeo di squadra può essere vinto da più giocatori.

Analizzando l'entità Squadra passiamo ora a descrivere le sue relazioni, innanzitutto le squadre andranno a comporre un Campionato, caratterizzato da un nome.

Infine una squadra può vincere uno o più trofei di squadra e viceversa un trofeo di squadra può essere vinto da una e una sola squadra.

2 Ristrutturazione

Durante questa fase andremo a modificare alcuni aspetti del diagramma concettuale al fine di renderlo più adatto per una traduzione al modello logico

2.1 Analisi delle ridondanze

Abbiamo diverse ridondanze all'interno del diagramma concettuale, andiamo innanzitutto ad intervenire sull'entità Carriera; essa ha attributi che sono delle "estensioni" degli attributi delle relazioni "Milita", quindi tutti i suoi dati sono già reperibili, di conseguenza andremo ad eliminare l'entità Carriera.

Un'altra ridondanza che si può notare è la relazione "Vince_trofeo_squadra" in quanto costituisce un'informazione reperibile già dalla relazione "SquadraVinceTrofeo" tra l'entità Squadra e l'entità TrofeoDiSquadra; di conseguenza andremo ad eliminare tale relazione.

2.2 Analisi degli identificativi

In questa fase andremo a scegliere (o a creare) un attributo per identificare univocamente le varie entità presenti nello schema precedente, in particolare:

1. L'entità Giocatore ha presente un insieme piuttosto ampio di chiavi candidate, per questo motivo è stato scelto di aggiungere l'attributo *CodFiscale.*
2. L'entità Squadra ha già presente l'attributo Nome che può fungere da chiave primaria.
3. L'entità Campionato ha diverse chiavi candidate Nome, Anno, quindi per semplificare gli accessi si è deciso di creare un attributo IdCampionato come chiave primaria.
4. L'entità Trofeo ha due chiavi candidate (Nome, Anno), si è quindi deciso di trasformarle entrambe in chiave primaria.

2.3 Rimozione attributi multivalore

In questa fase andremo ad eliminare gli attributi che contengono delle liste di valori.

L'unico attributo multivalore presente nel diagramma concettuale è l'attributo *Caratteristica che indica una caratteristica particolare per quel giocatore, che per* l'appunto può essere anche più di una.

Andiamo ad aggiungere un'entità Caratteristica, con attributo Tipo_caratteristica che è anche chiave primaria.

Un giocatore può avere nessuna o più caratteristiche, viceversa una caratteristica può essere posseduta da uno o più giocatori.

2.4 Rimozione attributi composti

Non sono presenti attributi composti.

2.5 Partizione/Accorpamento delle associazioni

Non sono presenti associazioni 1:1.

2.6 Rimozione delle gerarchie

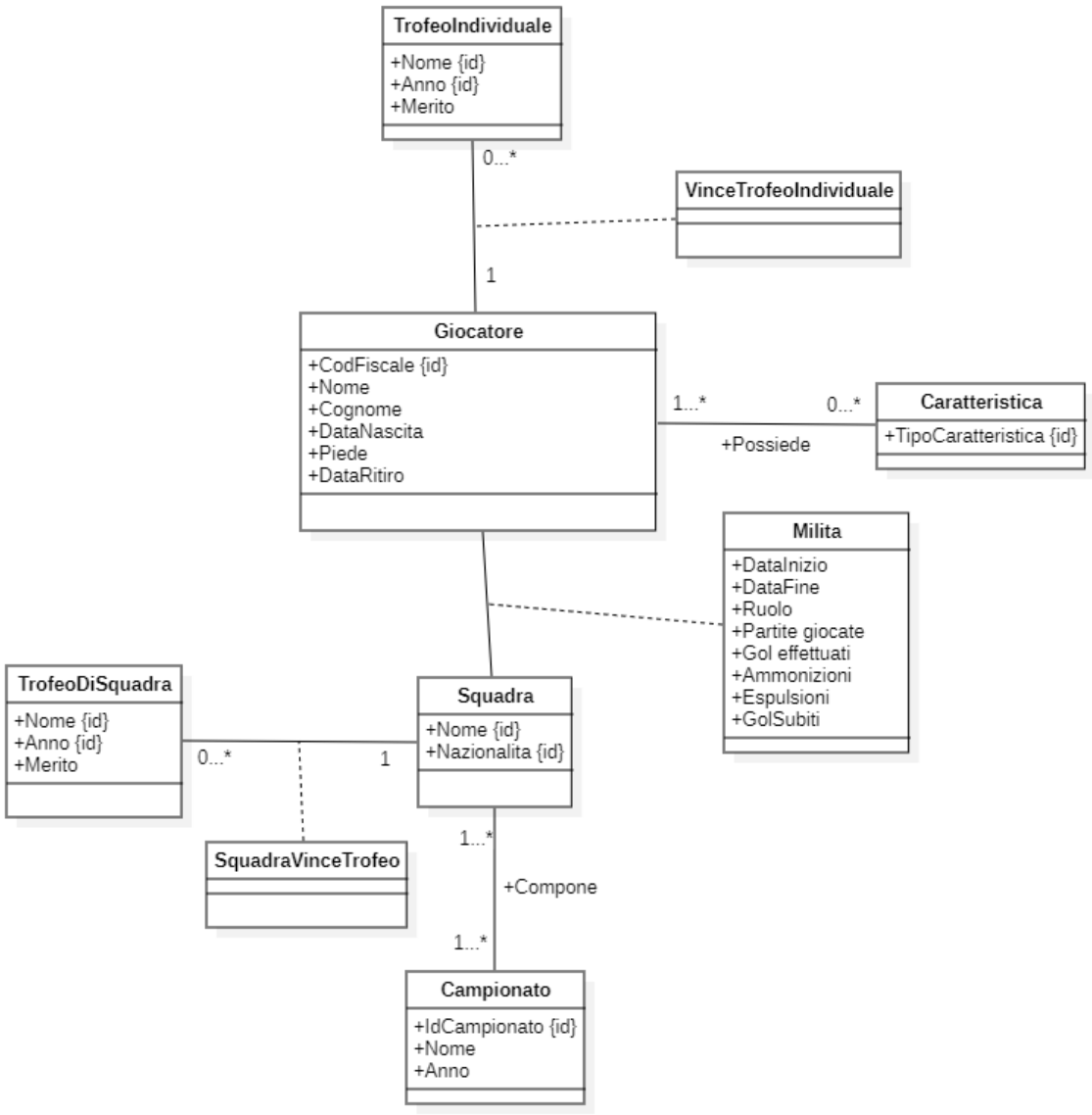
In questo diagramma sono presenti 1 generalizzazione ed 1 composizione. In particolare:

1. Per quanto riguarda la generalizzazione Trofeo, siccome è una totale disgiunta, si è deciso di accorpare il padre nelle figlie, ottenendo come risultato:
   - Un'entità TrofeoIndividuale; e sarà in relazione con l'entità **Giocatore tramite la relazione "VinceTrofeoIndividuale" con** molteplicità 1:N
   - Un'entità TrofeoDiSquadra; E sarà in relazione con l'entità **Squadra tramite la relazione "SquadraVinceTrofeo" con molteplicità** 1:N.
2. Per quanto riguarda la generalizzazione Giocatore, siccome portiere non ha attributi aggiuntivi e la generalizzazione è una totale disgiunta, si è deciso di accorpare l'entità figlie nelle entità padre, ma se andassimo ad

inserire all'interno dell'entità giocatore l'attributo Ruolo (per l'identificazione del ruolo del giocatore), questo risulterà ridondante, in quanto per rendere più dinamico il cambio di ruolo, da squadra in squadra, si è deciso di inserire l'attributo Ruolo solo nell'associazione milita.

3. Infine per quanto riguarda la composizione di Campionato si è deciso di creare una relazione tra essa e l'entità Squadra di molteplicità N:N, cioè:
    - Una squadra compone uno o più campionati, viceversa un campionato è composto da una o più squadre.

2.7 Diagramma UML ristrutturato



3 Dizionario delle classi, delle associazioni e dei vincoli

3.1 Dizionario delle classi

Entità	Descrizione	Attributi
Giocatore	Entità principale della base di dati, calciatore	**CodFiscale (String): identificativo** del giocatore. **Nome (String): nome anagrafico** del giocatore. **Cognome (String): cognome** anagrafico del giocatore. **DataNascita (Date): data di** nascita del giocatore. **Piede (String): stringa per** indicare il piede forte del giocatore **DataRitiro (Date): data di ritiro** del giocatore
Squadra	Associazione di appartenenza di un giocatore	**Nome (String): nome** identificativo della squadra **Nazionalita (String): paese di** provenienza della squadra
Campionato	Competizione formata da diverse squadre che giocano tra di loro, ed una classifica	**IdCampionato (Serial): seriale per** identificare il campionato **Nome (String): nome** identificativo del campionato **Anno (String): annata del** campionato
Trofeo individuale	Premio assegnato ad un singolo giocatore	**NomeTrofeo (String): nome del** trofeo **Anno (String): anno di** assegnazione del trofeo **Merito (String): il motivo per il** quale è stato assegnato il trofeo

Entità	Descrizione	Attributi
Caratteristiche	Tratti distintivi o iconici del giocatore	**TipoCaratteristica (String): gesto** tecnico caratteristico del giocatore
Trofeo di squadra	Premio assegnato ad una squadra	**NomeTrofeo (String): nome del** trofeo **Anno (String): anno di** assegnazione del trofeo **Merito (String): il motivo per il** quale è stato assegnato il trofeo

3.2 Dizionario associazioni

Associazione	Descrizione	Attributi
Possiede	Associazione molti a molti tra Giocatore e **Caratteristiche. Un giocatore** possiede nessuna o più caratteristiche, viceversa una caratteristica è contenuta da uno o più giocatori.	
Milita	Associazione molti a molti tra Giocatore e Squadra. Un giocatore milita in una o più squadre, una squadra ospita uno o più giocatori	**DataInizio (Date): data inizio** del contratto **DataFine (Date): data fine** contratto **Ruolo (String): ruolo del** giocatore in una determinata squadra **PartiteGiocate (int): numero** di partite giocate **GolEffettuati (int): numero** di gol segnati **GolSubiti (int): nel caso in** cui il giocatore sia un portiere è il numero di gol subiti **Ammonizioni (int): numero** ammonizioni **Espulsioni (int): numero** espulsioni

Associazione	Descrizione	Attributi
SquadraVinceTrofeo	Associazione uno a molti tra **Squadra e TrofeoDiSquadra.** Una squadra può vincere nessuno o più trofei di squadra, viceversa un trofeo di squadra può essere vinto da nessuno o più squadre	
VinceTrofeoIndividuale	Associazione uno a molti tra **Giocatore e TrofeoIndividuale. Un** giocatore può vincere nessuno o più trofei individuali, viceversa un trofeo individuale può essere vinto da nessuno o un giocatore	
Compone	Associazione molti a molti tra Squadra e Campionato. Una squadra compone uno o più campionati, viceversa un campionato è composto da una o più squadre	

4 Traduzione al modello logico

Nelle seguenti traduzione al modello logico verrà utilizzata la sottolineatura per indicare la chiave primaria, e una freccia '↑' per indicare una chiave esterna.

4.1 Mapping associazioni

4.1.1 Associazioni 1:N

Associazione SquadraVinceTrofeo tra l'entità Squadra e l'entità TrofeoDiSquadra, accorpamento della relazione nell'entità TrofeoDiSquadra:

- TrofeoDiSquadra (Nome, Anno, Merito, NomeSquadra↑)

Con l'attributo NomeSquadra della relazione TrofeoDiSquadra come chiave esterna sull'attributo Nome della relazione Squadra

Associazione VinceTrofeoIndividuale tra l'entità Giocatore e l'entità **TrofeoIndividuale, accorpamento della relazione nell'entità TrofeoIndividuale:**

- TrofeoIndividuale (Nome, Anno, Merito, CodFisc↑)

Con l'attributo CodFisc della relazione TrofeoIndividuale come chiave esterna sull'attributo CodFisc della relazione Giocatore

Associazione Compone tra l'entità Squadra e l'entità Campionato, accorpamento della relazione nell'entità Squadra:

- Squadra (Nome, Nazionalita, IdCampionato)

Con l'attributo IdCampionato della relazione Squadra come chiave esterna sull'attributo IdCampionato dell'entità Campionato

4.1.2 Associazioni N:N

Per questo tipo di associazioni, durante la traduzione al modello logico, verrà creata una relazione contenente le chiavi esterne dell'entità che ne prendono parte.

- Associazione Milita tra l'entità Giocatore e l'entità Squadra:
  - Milita (CodFisc↑, NomeSquadra↑, NazionalitaSquadra↑, DataInizio, DataFine, RuoloSquadra, PartiteGiocate, GolEffettuati, GolSubiti, Ammonizioni, Espulsioni)

Con l'attributo CodFisc della relazione Milita come chiave esterna sull'attributo *CodFisc della relazione Giocatore*

Con l'attributo NomeSquadra e NazionalitaSquadra della relazione Milita come chiave esterna sulla relazione Squadra

- Associazione Possiede tre l'entità Giocatore e l'entità Caratteristiche:
    - Possiede (CodFisc↑, Caratteristica↑)

Con l'attributo CodFisc della relazione Possiede come chiave esterna sulla relazione **Giocatore**

Con l'attributo Caratteristica della relazione Possiede come chiave esterna sulla relazione Caratteristica

4.2 Mapping entità

- Giocatore (CodFisc, Nome, Cognome, DataNascita, Piede, DataRitiro)
- Squadra (relazione già mappata in precedenza)
- Campionato (IdCampionato, Nome, Anno)
- TrofeoIndividuale (relazione già mappata in precedenza)
- TrofeoDiSquadra (relazione già mappata in precedenza)
- Caratteristica (TipoCaratteristica)

5 Progettazione fisica

In questa fase andremo ad implementare fisicamente il modello logico prodotto nelle fasi precedenti all'interno di un DBMS.

5.1 Creazione dello schema

CREATE SCHEMA progetto

5.2 Inserimento delle relazioni

```
–Creazione tabella Giocatore
CREATE TABLE progetto.Giocatore(
CodFisc char(16) PRIMARY KEY,
Nome varchar NOT NULL,
Cognome varchar NOT NULL,
DataNascita Date NOT NULL,
Piede char(2) NOT NULL,
DataRitiro Date NULL
);

–Creazione tabella Campionato
CREATE TABLE progetto.Campionato(
IdCampionato SERIAL,
Nome varchar,
Anno varchar,
PRIMARY KEY (IdCampionato)
);

–Creazione tabella Caratteristica
CREATE TABLE progetto.Caratteristica(
TipoCaratteristica varchar PRIMARY KEY
);

–Creazione tabella Squadra
CREATE TABLE progetto.Squadra(
Nome varchar NOT NULL,
Nazionalita varchar NOT NULL,
IdCampionato integer NOT NULL,
```

```
PRIMARY KEY (Nome, Nazionalita),
UNIQUE(Nome, IdCampionato),
FOREIGN KEY (IdCampionato) REFERENCES
progetto.Campionato(IdCampionato)
);

–Creazione tabella TrofeoDiSquadra
CREATE TABLE progetto.TrofeoDiSquadra(
Nome varchar NOT NULL,
Anno varchar NOT NULL,
Merito varchar NOT NULL,
NomeSquadra varchar NOT NULL,
NazionalitaSquadra varchar NOT NULL,
PRIMARY KEY(Nome, Anno),
FOREIGN KEY (NomeSquadra, NazionalitaSquadra) REFERENCES
progetto.Squadra(Nome, Nazionalita)
ON UPDATE CASCADE
ON DELETE CASCADE,
UNIQUE(Nome, Anno)
);

–Creazione tabella TrofeoIndividuale
CREATE TABLE progetto.TrofeoIndividuale(
Nome varchar NOT NULL,
Anno varchar NOT NULL,
Merito varchar NOT NULL,
CodF varchar NOT NULL,
PRIMARY KEY(Nome, Anno),
FOREIGN KEY (CodF) REFERENCES progetto.Giocatore(CodFisc)
ON UPDATE CASCADE
ON DELETE CASCADE,
```

```
UNIQUE(Nome, Anno)
);

–Creazione tabella Milita
CREATE TABLE progetto.Milita(
CodFisc char(16) NOT NULL,
NomeSquadra varchar NOT NULL,
NazionalitaSquadra varchar NOT NULL,
DataInizio Date NOT NULL,
DataFine Date NOT NULL,
Ruolo varchar NOT NULL,
PartiteGiocate integer NOT NULL,
GolEffettuati integer NOT NULL,
GolSubiti integer NULL,
Ammonizioni integer NOT NULL,
Espulsioni integer NOT NULL,

PRIMARY KEY(CodFisc, NomeSquadra, NazionalitaSquadra, DataInizio),
FOREIGN KEY (CodFisc) REFERENCES progetto.Giocatore(CodFisc)
ON UPDATE CASCADE
ON DELETE CASCADE,
FOREIGN KEY (NomeSquadra, NazionalitaSquadra) REFERENCES
progetto.Squadra(Nome, Nazionalita)
ON UPDATE CASCADE
ON DELETE CASCADE
);

–Creazione tabella Possiede
CREATE TABLE progetto.Possiede(
CodFisc char(16) NOT NULL,
```

Caratteristica varchar NOT NULL,

PRIMARY KEY(CodFisc, Caratteristica),

FOREIGN KEY (CodFisc) REFERENCES progetto.Giocatore(CodFisc),

FOREIGN KEY (Caratteristica) REFERENCES
progetto.Caratteristica(TipoCaratteristica)

);

5.3 Inserimento vincoli

ALTER TABLE progetto.Milita

ADD CONSTRAINT DataMilitanza

CHECK(DataInizio < DataFine);

–**Descrizione: la data inizio della militanza di un giocatore in una squadra non può** essere maggiore della data di fine militanza.

ALTER TABLE progetto.Giocatore

ADD CONSTRAINT checkPiede

CHECK(Piede='Dx' OR Piede='Sx' OR Piede='Am');

–**Descrizione: un giocatore può essere solo destro(Dx), sinistro(Sx), oppure** ambidestro(Am).

ALTER TABLE progetto.Milita

ADD CONSTRAINT checkDati

CHECK(PartiteGiocate >= 0 AND GolEffettuati >= 0 AND Ammonizioni >= 0
AND Espulsioni >= 0);

–**Descrizione: le partite giocate, i gol effettuati, le ammonizioni e le espulsioni non** possono avere valore minore di 0.

ALTER TABLE progetto.Milita

ADD CONSTRAINT checkRuoli

CHECK(ruolo = 'Portiere' OR ruolo = 'Difensore' OR ruolo = 'Centrocampista' OR ruolo = 'Attaccante');

–**Descrizione: i ruoli all'interno della militanza possono essere solo quelli elencanti**

ALTER TABLE progetto.Campionato

ADD CONSTRAINT checkanno1

CHECK(anno LIKE '____/__');

Descrizione: gli anni dei campionati potranno apparire solo nella forma "yyyy/yy"

ALTER TABLE progetto.trofeoindividuale

ADD CONSTRAINT checkanno2

CHECK(anno LIKE '____/__')

Descrizione: gli anni dei trofei individuali potranno apparire solo nella forma "yyyy/yy"

ALTER TABLE progetto.trofeodisquadra

ADD CONSTRAINT checkanno3

CHECK(anno LIKE '____/__')

Descrizione: gli anni dei trofei di squadra potranno apparire solo nella forma "yyyy/yy"

5.4 Creazione procedure e funzioni

CREATE PROCEDURE progetto.InserisciGiocatore(IN CodF char(16), IN Nome varchar, IN Cognome varchar, IN Datanascita date, IN Piede char(2))

AS $$

BEGIN

INSERT INTO progetto.Giocatore (CodFisc, Nome, Cognome, Datanascita, Piede)

VALUES (CodF, Nome, Cognome, Datanascita, Piede);

END;

$$ LANGUAGE plpgsql;

–**Descrizione: procedura per l'inserimento di un giocatore.**

CREATE OR REPLACE FUNCTION progetto.getGiocatore(CodF IN char(16))

RETURNS SETOF progetto.Giocatore

LANGUAGE plpgsql

AS

$$

BEGIN

    RETURN QUERY(SELECT * FROM progetto.Giocatore t where t.CodFisc = CodF);

END;

$$;

–**Descrizione: funzione che dato in input un codice fiscale restituisce la riga del** giocatore corrispondente.

–Le seguenti funzioni riguardo ai ruoli sono state implementate per questioni legate all'applicativo, per organizzare una buona visualizzazione dei dati.

CREATE OR REPLACE FUNCTION progetto.getRuoli(CodF varchar)

RETURNS TABLE

(Ruolo varchar,

Ricorrenze bigint)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT t.Ruolo, Count(*)

FROM progetto.Milita t

WHERE CodFisc = CodF

GROUP BY t.Ruolo;

END; $$;

–**Descrizione: funzione che dato in input un codice fiscale, restituisce una lista** contenente tutti i ruoli in cui il giocatore corrispondente al codice fiscale ha giocato.

```
CREATE OR REPLACE FUNCTION progetto.getMFRuolo(CodF varchar)
RETURNS TABLE(
Ruolo varchar)
LANGUAGE plpgsql
AS $$
BEGIN
RETURN QUERY
SELECT t.Ruolo
FROM progetto.getRuoli(CodF) t
WHERE ricorrenze = (SELECT MAX(Ricorrenze)
FROM progetto.getRuoli(CodF));
END; $$;
```

–**Descrizione: funzione che dato in input un codice fiscale restituisce i ruoli con il** maggior numero di ricorrenze all'interno della carriera del giocatore.

```
CREATE OR REPLACE FUNCTION progetto.getSingleMFRuolo(CodF
char(16))
RETURNS varchar
LANGUAGE plpgsql
AS $$
DECLARE
ruolo_giocatore varchar;
BEGIN
SELECT t.Ruolo INTO ruolo_giocatore
FROM progetto.getmfruolo(CodF) t
LIMIT 1;

RETURN ruolo_giocatore;
```

END; $$;

–**Descrizione: funzione che dato un codice fiscale restituisce uno dei ruoli con il** maggior numero di ricorrenze del giocatore

CREATE OR REPLACE FUNCTION progetto.getMilitanza(CodF char(16))

RETURNS SETOF progetto.Milita

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT *

FROM progetto.Milita t

WHERE t.CodFisc = CodF AND t.Nomesquadra = Nomesquadra;

END; $$;

–**Descrizione: funzione che restituisce tutte le militanze di un giocatore**

CREATE OR REPLACE FUNCTION progetto.getSquadra(Nome varchar(16), Nazionalita varchar)

RETURNS SETOF progetto.Squadra

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT *

FROM progetto.Squadra t

WHERE t.Nome = Nome;

END; $$;

–**Descrizione: funzione che restituisce i dati di una determinata squadra**

CREATE OR REPLACE FUNCTION progetto.getTrofeiSquadra(Nomef varchar, Nazionalitaf varchar)

```
RETURNS SETOF progetto.TrofeodiSquadra
LANGUAGE plpgsql
AS $$
BEGIN
RETURN QUERY
SELECT *
FROM progetto.TrofeodiSquadra t
WHERE t.Nomesquadra = Nomef AND t.Nazionalitasquadra = Nazionalitaf;
END; $$;
```

–**Descrizione: funzione che restituisce i dati dei trofei di una determinata squadra**

```
CREATE OR REPLACE FUNCTION progetto.getCaratteristiche (CodF char(16))
RETURNS TABLE(Caratteristica varchar)
LANGUAGE plpgsql
AS $$
BEGIN
RETURN QUERY
SELECT p.caratteristica
FROM progetto.possiede p
WHERE p.CodFisc = CodF;
END; $$;
```

–**Descrizione: funzione che restituisce tutte le caratteristiche tipiche di un** giocatore

```
CREATE OR REPLACE FUNCTION progetto.getRuoloUtente()
RETURNS varchar AS $$
DECLARE
str_out varchar;
BEGIN
SELECT * INTO str_out from current_user;
```

IF(str_out='admin_db') THEN

RETURN 'amministratore';

ELSE

RETURN 'utente';

END; $$ LANGUAGE plpgsql;

–**Descrizione: funzione che restituisce il ruolo dell'utente che ha effettuato l'accesso**

CREATE OR REPLACE FUNCTION progetto.carrieraGiocatore(codicefiscale char(16))

RETURNS TABLE(

CodFisc char(16),

Nome varchar,

Cognome varchar,

Datanascita date,

Eta integer,

Piede char(2),

SquadraAttuale varchar,

RuoloPrincipale varchar,

Caratteristiche varchar,

Partitegiocate integer,

Goleffettuati integer,

Golsubiti integer,

Ammonizioni integer,

Espulsioni integer)

AS $$

DECLARE

nome varchar; cognome varchar; squadra varchar; ruolo varchar; caratteristiche varchar; carat varchar;

datan date; datar date;

piede char(2);

partite integer; goleff integer; golsub integer; amm integer; esp integer; eta integer;

cursor_caratteristiche REFCURSOR;

BEGIN

EXECUTE 'CREATE TABLE progetto.Tmp(

CodFisc char(16),

Nome varchar,

Cognome varchar,

Datanascita date,

Eta integer,

Piede char(2),

SquadraAttuale varchar,

RuoloPrincipale varchar,

Caratteristiche varchar,

Partitegiocate integer,

Goleffettuati integer,

Golsubiti integer,

Ammonizioni integer,

Espulsioni integer)';

IF NOT EXISTS(SELECT g.CodFisc FROM progetto.Giocatore g WHERE g.CodFisc = codicefiscale) THEN

RAISE NOTICE 'Codice fiscale non collegato a nessun giocatore';

END IF;

caratteristiche = '';

OPEN cursor_caratteristiche FOR SELECT caratteristica FROM progetto.getcaratteristiche(codicefiscale);

LOOP

FETCH cursor_caratteristiche INTO carat;

EXIT WHEN NOT FOUND;

```
caratteristiche = caratteristiche||', '||carat;
END LOOP;
CLOSE cursor_caratteristiche;
caratteristiche = SUBSTRING(caratteristiche from 2 for
LENGTH(caratteristiche));
SELECT g.nome, g.cognome, g.piede, g.datanascita, g.dataritiro
INTO nome, cognome, piede, datan, datar
FROM progetto.Giocatore g
WHERE g.CodFisc = codicefiscale;

SELECT EXTRACT('YEAR' FROM AGE(CURRENT_DATE, datan))
INTO eta;

IF(datar IS NULL) THEN
SELECT m.Nomesquadra
INTO squadra
FROM progetto.Giocatore g, progetto.Milita m
WHERE g.CodFisc = codicefiscale AND g.CodFisc = m.CodFisc AND
m.Datainizio = (SELECT Max(m.Datainizio)
FROM progetto.Giocatore g, progetto.Milita m
WHERE g.CodFisc = codicefiscale AND g.CodFisc =
m.CodFisc);
ELSE
squadra = 'Ritirato';
END IF;

SELECT t.ruolo
INTO ruolo
FROM progetto.getmfruolo(codicefiscale) t;
```

IF EXISTS(SELECT * FROM progetto.getruoli(codicefiscale) t WHERE t.ruolo = 'Portiere') THEN

SELECT SUM(m.partitegiocate), SUM(m.goleffettuati), SUM(m.golsubiti), SUM(m.ammonizioni), SUM(m.espulsioni)

INTO partite, goleff, golsub, amm, esp

FROM progetto.Giocatore g, progetto.Milita m

WHERE g.CodFisc = codicefiscale AND g.CodFisc = m.CodFisc;

INSERT INTO progetto.Tmp VALUES(codicefiscale, nome, cognome, datan, eta, piede, squadra, ruolo, caratteristiche, partite, goleff, golsub, amm, esp);

RETURN QUERY SELECT * FROM progetto.Tmp; DROP TABLE progetto.Tmp; RETURN;

END IF;

SELECT SUM(m.partitegiocate), SUM(m.goleffettuati), SUM(m.ammonizioni), SUM(m.espulsioni)

INTO partite, goleff, amm, esp

FROM progetto.Giocatore g, progetto.Milita m

WHERE g.CodFisc = codicefiscale AND g.CodFisc = m.CodFisc;

INSERT INTO progetto.Tmp VALUES(codicefiscale, nome, cognome, datan, eta, piede, squadra, ruolo, caratteristiche, partite, goleff, golsub, amm, esp);

RETURN QUERY SELECT * FROM progetto.Tmp; DROP TABLE progetto.Tmp; RETURN;

END; $$ LANGUAGE plpgsql;

–**Descrizione: funzione che dato in input il codice fiscale restituisce una tabella** contenente un resoconto dei dati della carriera del giocatore

CREATE OR REPLACE FUNCTION progetto.carrieragiocatoriall()

RETURNS TABLE(CodiceFiscale char(16),

Nome varchar,

Cognome varchar,

Datanascita date,

Eta integer,

```
Piede char(2),
SquadraAttuale varchar,
RuoloPrincipale varchar,
Caratteristiche varchar,
Partitegiocate integer,
Goleffettuati integer,
Golsubiti integer,
Ammonizioni integer,
Espulsioni integer)
AS $$
DECLARE
cursor_codf REFCURSOR;
codf char(16); nomef varchar; cognomef varchar; datanf date; piedef char(2);
squadraf varchar; caratteristiche varchar;
ruolof varchar; partitef integer; goleff integer; golsubf integer; ammf integer;
espf integer; eta integer;
BEGIN
EXECUTE 'CREATE TABLE progetto.Tmpfunc(CodFisc char(16),
Nome varchar,
Cognome varchar,
Datanascita date,
Eta integer,
Piede char(2),
SquadraAttuale varchar,
RuoloPrincipale varchar,
Caratteristiche varchar,
Partitegiocate integer,
Goleffettuati integer,
Golsubiti integer,
Ammonizioni integer,
Espulsioni integer)';
```

OPEN cursor_codf FOR SELECT CodFisc FROM progetto.Giocatore;

LOOP

FETCH cursor_codf INTO codf;

EXIT WHEN NOT FOUND;

SELECT * INTO codf, nomef, cognomef, datanf, eta, piedef, squadraf, ruolof, caratteristiche, partitef, goleff, golsubf, ammf, espf FROM progetto.carrieragiocatore(codf);

INSERT INTO progetto.Tmpfunc VALUES(codf, nomef, cognomef, datanf, eta, piedef, squadraf, ruolof, caratteristiche, partitef, goleff, golsubf, ammf, espf);

END LOOP;

CLOSE cursor_codf;

RETURN QUERY SELECT * FROM progetto.Tmpfunc; DROP TABLE progetto.Tmpfunc; RETURN;

END; $$ LANGUAGE plpgsql;

–**Descrizione: funzione che restituisce un resoconto di tutte le militanze di tutti i** giocatori

CREATE OR REPLACE FUNCTION progetto.ricerca(nomeg varchar, ruolo varchar, piedeg char(2), golsegnati integer,

segnogolsegnati varchar, ordinegolsegnati varchar, golsubitig integer,

segnogolsubiti
varchar, ordinegolsubiti varchar, etag integer, segnoeta varchar, ordineeta varchar,

squadra
varchar)

RETURNS TABLE (Codicefiscale char(16),

Nome varchar,

Cognome varchar,

Datanascita date,

Eta integer,

Piede char(2),

```
SquadraAttuale varchar,
RuoloPrincipale varchar,
Caratteristiche varchar,
Partitegiocate integer,
Goleffettuati integer,
Golsubiti integer,
Ammonizioni integer,
Espulsioni integer)
AS $$
DECLARE
condizione varchar;
andv varchar;
return_query varchar;
BEGIN
condizione = 'WHERE ';
andv = '';
return_query = 'SELECT * FROM progetto.carrieragiocatoriall() ';

IF(nomeg <> ' ') THEN
condizione = condizione || 'nome = '||quote_literal(nomeg);
andv = ' AND ';
END IF;

IF(ruolo <> ' ') THEN
condizione = condizione||andv||'RuoloPrincipale =
'||quote_literal(ruolo);
andv = ' AND ';
END IF;

IF(piedeg <> ' ') THEN
```

```
condizione = condizione||andv||'piede = '||quote_literal(piedeg);
andv = ' AND ';
END IF;

IF(golsegnati <> -1) THEN
IF(segnogolsegnati <> ' ') THEN
condizione = condizione||andv||'goleffettuati
'||segnogolsegnati||' '||golsegnati;
ELSE
condizione = condizione||andv||'goleffettuati '||'=
'||golsegnati;
END IF;
andv = ' AND ';
END IF;

IF(golsubitig <> -1) THEN
IF(segnogolsubiti <> ' ') THEN
condizione = condizione||andv||'golsubiti IS NOT NULL AND
golsubiti '||segnogolsubiti||' '||golsubitig;
ELSE
condizione = condizione||andv||'golsubiti IS NOT NULL AND
golsubiti '||'= '||golsubitig;
END IF;
andv = ' AND ';
END IF;

IF(etag <> -1) THEN
IF(segnoeta <> ' ') THEN
condizione = condizione||andv||'eta '||segnoeta||etag;
ELSE
condizione = condizione||andv||'eta '||'= '||etag;
END IF;
```

```
andv = ' AND ';
END IF;

IF(squadra <> ' ') THEN
condizione = condizione||andv||'squadraattuale =
'||quote_literal(squadra);
andv = ' AND ';
END IF;

IF(condizione = 'WHERE ') THEN
condizione = '';
END IF;
condizione = condizione||' ORDER BY ';

andv = '';
IF(ordinegolsubiti <> ' ') THEN
condizione = condizione||'Golsubiti '||ordinegolsubiti;
andv=', ';
END IF;

IF(ordinegolsegnati <> ' ') THEN
condizione = condizione||andv||'Goleffettuati '||ordinegolsegnati;
andv=', ';
END IF;

IF(ordineeta <> ' ') THEN
condizione = condizione||andv||'Eta '||ordineeta;
andv=', ';
END IF;
```

IF(ordinegolsubiti = ' ' AND ordinegolsegnati = ' ' AND ordineeta = ' ')
THEN

condizione = SUBSTRING(condizione, 1, position(' ORDER BY ' IN condizione));

END IF;

return_query = return_query || condizione;

RETURN QUERY EXECUTE return_query;

END; $$ LANGUAGE plpgsql;

–Descrizione: funzione che restituisce la tabella di resoconto del giocatore filtrata in base alla scelta dei parametri

CREATE OR REPLACE PROCEDURE progetto.createUser(nome VARCHAR, pass VARCHAR(20))
AS $$
BEGIN
IF NOT EXISTS (SELECT FROM pg_user WHERE usename = nome) THEN
EXECUTE 'CREATE USER ' || quote_ident(nome) || ' WITH PASSWORD '||quote_literal(pass);
EXECUTE 'GRANT utente TO ' || quote_ident(nome);
ELSE
RAISE NOTICE 'Utente % già esistente', nome;
END IF;
END;
$$ LANGUAGE plpgsql;

–**Descrizione: procedura che dato in input un nome, ed una password, permette di** creare un nuovo utente che come permesso avrà solo quello di interrogare la base di dati.

5.5 Creazione trigger

CREATE OR REPLACE FUNCTION progetto.updateGolSubiti()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.Ruolo = 'Portiere' AND NEW.Ruolo <> OLD.Ruolo THEN
NEW.Golsubiti := 0;
END IF;
RETURN NEW;
END;

$$ LANGUAGE plpgsql;

–**Descrizione: funzione trigger che si occupa di modificare l'attributo Golsubiti della** –tabella Militanza nel momento in cui il ruolo del giocatore viene modificato in – 'Portiere';

```
CREATE TRIGGER triggerUpdateGolSubiti
BEFORE UPDATE ON progetto.Milita
FOR EACH ROW
EXECUTE FUNCTION progetto.updateGolSubiti();
```

–**Descrizione: implementazione trigger con la funzione descritta sopra**

```
CREATE OR REPLACE FUNCTION progetto.inserisciDateOverleap()
RETURNS TRIGGER AS $$
BEGIN
IF EXISTS (
SELECT 1
FROM progetto.Milita t
WHERE NEW.CodFisc = t.CodFisc AND NEW.Nomesquadra =
t.Nomesquadra AND (NEW.datainizio BETWEEN t.datainizio AND t.datafine
OR NEW.datafine BETWEEN t.datainizio AND t.datafine)
)
THEN
– Le nuove date sono comprese tra date già presenti
RAISE EXCEPTION 'Errore: le date di inizio o fine sono comprese tra date
già presenti in tabella';
END IF;

RETURN NEW; – Restituisci la riga inserita
END;
$$ LANGUAGE plpgsql;
```

–**Descrizione: funzione trigger che si occupa del controllo delle date all'interno** – della tabella militanza, se uno stesso giocatore ha date di militanza che si – sovrappongono tra di loro viene sollevato un errore che impedisce l'inserimento – della tupla.

```
– Creazione del trigger associato alla funzione
CREATE TRIGGER triggerDateOverleap
BEFORE INSERT ON progetto.Milita
FOR EACH ROW
EXECUTE FUNCTION progetto.inserisciDateOverleap();
```

```
CREATE FUNCTION progetto.inseriscimilitanza() RETURNS TRIGGER AS $$
BEGIN
IF(NEW.ruolo = 'Portiere') THEN
IF(NEW.golsubiti IS NULL) THEN
NEW.golsubiti = 0;
END IF;
END IF;

RETURN NEW;
END; $$ LANGUAGE plpgsql;
–Descrizione: funzione trigger che si occupa del controllo dei gol subiti all'interno
di una nuova tupla inserita dentro la tabella milita

–Creazione del trigger associato
CREATE TRIGGER inserimentoMilitanza
BEFORE INSERT ON progetto.Milita
FOR EACH ROW
EXECUTE FUNCTION progetto.inseriscimilitanza();


CREATE OR REPLACE FUNCTION progetto.datamilitanza() RETURNS
TRIGGER AS $$
BEGIN
IF(NEW.datainizio < (SELECT t.datanascita FROM progetto.giocatore t
WHERE NEW.codfisc = t.codfisc)) THEN
RAISE EXCEPTION 'La data di inizio militanza non può essere
minore della data di nascita del giocatore';
END IF;

RETURN NEW;
END; $$ LANGUAGE plpgsql;

–Descrizione: funzione trigger che controlla se la data di inizio militanza sia
minore della data di nascita del giocatore
–Creazione trigger associato alla funzione
CREATE TRIGGER datamilitanza
BEFORE INSERT ON progetto.Milita
FOR EACH ROW
EXECUTE FUNCTION progetto.datamilitanza()

CREATE OR REPLACE FUNCTION progetto.datatrofeo() RETURNS
TRIGGER AS $$
DECLARE
datan date;
annoint int;
```

```
annotrofeoint int;
BEGIN
SELECT datanascita INTO datan FROM progetto.giocatore WHERE codfisc =
NEW.codf;

annoint := EXTRACT(YEAR FROM datan);
annotrofeoint := CAST(SUBSTR(NEW.anno, 1, 4) AS INTEGER);

IF annoint > annotrofeoint THEN
RAISE EXCEPTION 'La data di assegnazione del trofeo non può essere
minore della data di nascita del giocatore';
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;
–Descrizione: funzione trigger che controlla se la data del trofeo individuale sia
minore della data di nascita del giocatore
–Creazione trigger associato alla funzione
CREATE TRIGGER datatrofeo
BEFORE INSERT ON progetto.trofeoindividuale
FOR EACH ROW
EXECUTE FUNCTION progetto.datatrofeo()

CREATE OR REPLACE FUNCTION progetto.squadramilitanza() RETURNS
TRIGGER AS $$
BEGIN
IF((SELECT t.dataritiro FROM progetto.Giocatore t WHERE t.codfisc =
NEW.codfisc) IS NOT NULL) THEN
RAISE EXCEPTION 'Il giocatore si è ritirato';
END IF;

RETURN NEW;
END; $$ LANGUAGE plpgsql;

–Descrizione: funzione trigger che controlla se la militanza che si vuole inserire
abbia collegata ad essa un giocatore ritirato.
–Creazione trigger associato alla funzione
CREATE TRIGGER ritiratomilita
BEFORE INSERT ON progetto.milita
FOR EACH ROW
EXECUTE FUNCTION progetto.squadramilitanza()
```

6 Creazione ruoli e utenti

```
CREATE ROLE amministratore WITH SUPERUSER;
CREATE USER admin_db WITH PASSWORD 'admin';
```

Descrizione: creazione dello user 'admin_db' ed assegnamento dei privilegi a ques'ultimo, l'amministratore avrà pieni privilegi su tutto il db.

```
CREATE ROLE ist_registrazione WITH LOGIN SUPERUSER PASSWORD
'registrazione';
```

Descrizione: creazione dello user 'ist_registrazione' ed assegnamento dei privilegi a ques'ultimo, questo sarà un ruolo ausiliare utilizzato per la registrazione di nuovi utenti

```
CREATE ROLE utente;
GRANT pg_read_all_data TO utente;
GRANT USAGE, CREATE ON SCHEMA progetto TO utente;
```

Descrizione: creazione dello ruolo utente ed assegnazione dei privilegi a ques'ultimo, tutti gli user che avranno questo ruolo potranno eseguire esclusivamente le interrogazioni

```
CREATE USER asd WITH PASSWORD 'asd';
GRANT ruolo_select to asd;
```

Descrizione: creazione del primo utente del db