# Progetto Basi di Dati 2023-24
# "UNIGE SOCIAL SPORT"
# Parte I

*[Numero gruppo: 43]*

*[Nome, Cognome e Matricola dei componenti:*

*Eugenio Vassallo 5577783, Mario Madalin Biberia 5608210, Umeshan Panchabalasingham 5606614]*

*[Per i gruppi che hanno effettuato la peer review le sezioni 1. e 2. devono essere consegnate in versione finale e consistenti con la progettazione relativa ai passi successivi.]*

## 1. REQUISITI RISTRUTTURATI

*[Riportare in questa sezione i requisiti ristrutturati in modo da eliminare ambiguità, evidenziando le modifiche effettuate]*

*1) Come vengono gestiti gli utenti: gestiamo due tipologie di utenti: semplici e premium. Gli utenti possono iscriversi agli eventi sportivi e, se premium, organizzare eventi sportivi e creare tornei e squadre.*

*2) Come vengono gestiti gli eventi sportivi: Gli eventi sportivi sono organizzati dagli utenti premium e possono coinvolgere altri utenti come giocatori o arbitri. Ogni evento appartiene a una categoria sportiva specifica e si svolge in un impianto del CUS Genova.*

*3) Gli impianti sportivi del CUS Genova ospitano gli eventi sportivi organizzati sulla piattaforma.*

*4) Gli eventi sportivi (intesi come match) appartengono a diverse categorie sportive, ognuna con il proprio regolamento e numero di giocatori.*

*5) Gestione delle iscrizioni: Gli utenti (sia semplici che premium) possono iscriversi agli eventi sportivi in qualità di giocatori o arbitri. Le iscrizioni sono gestite dall'organizzatore dell'evento.*

*6)* Tornei: *gli utenti premium possono organizzare tornei composti da diversi eventi sportivi. L'utente premium che crea il torneo decide il numero degli eventi e la modalità del torneo*

*7)Squadre: gli utenti premium possono creare e gestire squadre sportive. Il creatore di una squadra può manualmente inserire membri o in alternativa un utente può candidarsi ad una squadra In formazione . la*

*formazione è conclusa quando si raggiunge il numero massimo consentito di giocatori o quando lo decide il creatore della squadra*

*8) Esito evento sportivo: L'organizzatore di un evento sportivo può registrare i risultati delle partite.*

*9) Valutazioni fra utenti: gli utenti possono valutare le prestazioni degli altri giocatori dopo la conclusione di un evento. Il punteggio attribuito nella valutazione va da 1 a 10, sono consentiti anche commenti*

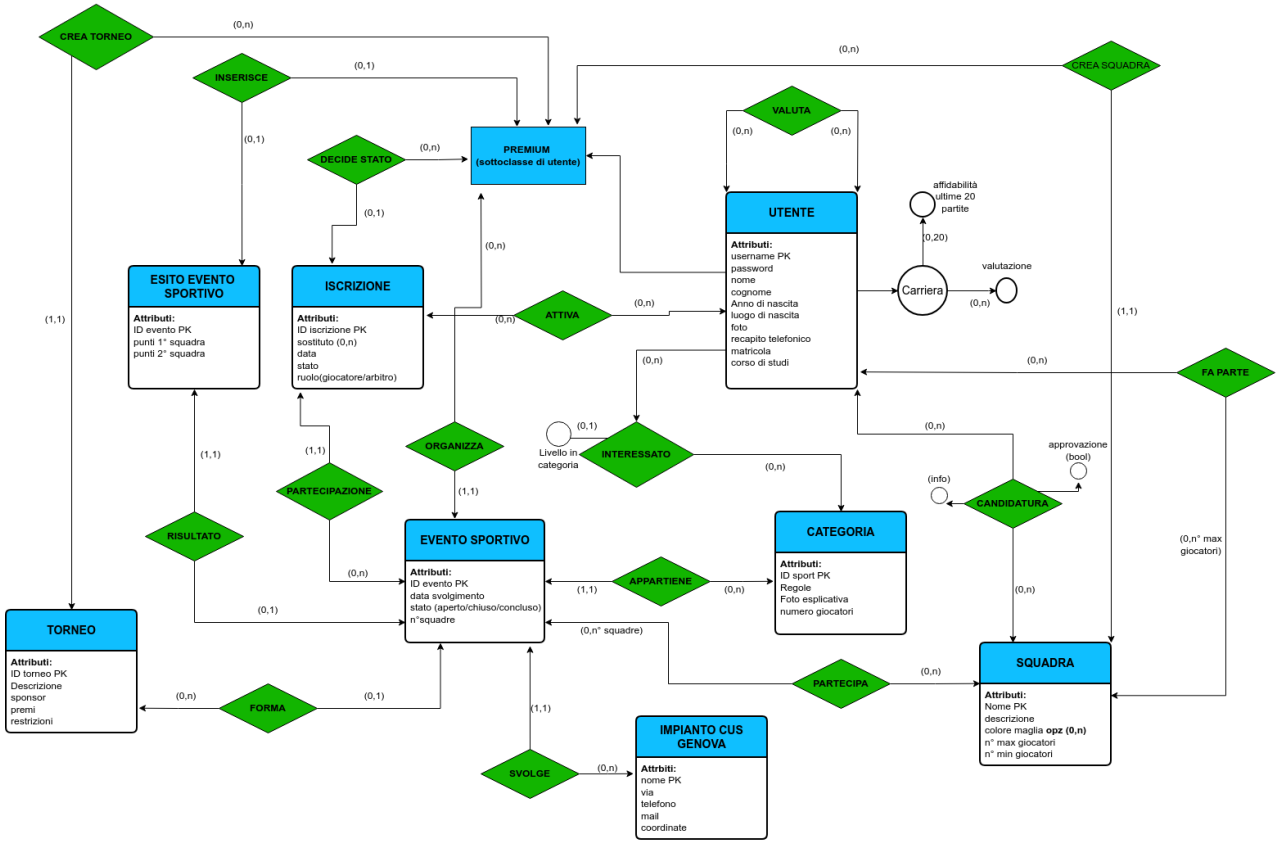
*10) Livello: Ogni utente ha un livello legato a ciascuna categoria sportiva, calcolato in base alle prestazioni nelle partite disputate e ai voti ricevuti. "Il calcolo è legato ad un algoritmo che, partendo da un livello medio pari a 60, lo modifica partita per partita in base a diversi parametri di riferimento (voti ricevuti, esito partita, affidabilità voti, ...)."*

*11) Affidabilità di un utente: Gli utenti possono essere indicati come inaffidabili in base a criteri di partecipazione e puntualità agli eventi. Questo valore è influenzato da: no show, ritardi, sostituzioni nelle ultime 20/10 partite, l'inaffidabilità di un utente è visibile agli utenti premium*

# 2. PROGETTO CONCETTUALE

## SCHEMA ENTITY RELATIONSHIP

*[Riportare in questa sezione il diagramma ER, indicando anche il tipo delle gerarchie di generalizzazioni]*




*(nota: abbiamo usato questa notazione suggerita da Draw.io perchè secondo noi più compatta e leggibile rispetto alla notazione vista a lezione)*

*La superclasse Utente rappresenta tutti gli utenti della piattaforma*

*La sottoclasse Premium rappresenta gli utenti premium, che hanno funzionalità aggiuntive come la possibilità di organizzare eventi sportivi, creare tornei, squadre e altri poteri decisionali in più*

*È un tipo di gerarchia totale(ogni utente deve essere o premium o base) e disgiunta (un utente non può essere sia utente base che utente premium)*

## DIZIONARIO DATI – DOMINI DEGLI ATTRIBUTI

***(Nome entità + attributi e dominio)***

***Utente:***

| |
|---|
| *username: VARCHAR(50), PRIMARY KEY*<br>*password: VARCHAR(50), NOT NULL*<br>*nome: VARCHAR(50)* |

*cognome: VARCHAR(50)*
*anno_nascita: INTEGER, CHECK (anno_nascita > 1900 AND anno_nascita <= EXTRACT(YEAR FROM CURRENT_DATE))*
*luogo_nascita: VARCHAR(100)*
*foto: BYTEA*
*telefono: VARCHAR(20)*
*matricola: VARCHAR(20)*
*corso_studi: VARCHAR(100)*

*Categoria:*

*ID_sport: SERIAL, PRIMARY KEY*
*regole: TEXT*
*numero_giocatori: INTEGER, CHECK (numero_giocatori > 0)*
*foto_esplicativa: BYTEA*

*Impianto CUS Genova:*

*nome: VARCHAR(100), PRIMARY KEY*
*via: VARCHAR(100)*
*telefono: VARCHAR(20)*
*email: VARCHAR(100)*
*coordinate: POINT (latitudine, longitudine)*

*Evento Sportivo:*

*ID_evento: SERIAL, PRIMARY KEY*
*data_svolgimento: DATE*
*stato: VARCHAR(10), CHECK (stato IN ('APERTO', 'CHIUSO', 'CONCLUSO'))*
*categoria: INTEGER, REFERENCES Categoria(ID_sport)*
*impianto: VARCHAR(100), REFERENCES Impianto_CUS_Genova(nome)*
*n_squadre: INTEGER, CHECK (n_squadre > 0)*

*Iscrizione:*

*ID_iscrizione: SERIAL, PRIMARY KEY*
*sostituto: BOOLEAN*
*data: DATE*
*stato: VARCHAR(10), CHECK (stato IN ('CONFERMATO', 'RIFIUTATO'))*
*ruolo: VARCHAR(15), CHECK (ruolo IN ('giocatore', 'arbitro'))*

*Esito Evento Sportivo:*

*ID_evento: INTEGER, PRIMARY KEY, REFERENCES Evento_Sportivo(ID_evento)*
*punti_1_squadra: INTEGER, CHECK (punti_1_squadra >= 0)*
*punti_2_squadra: INTEGER, CHECK (punti_2_squadra >= 0)*

*Torneo:*

| |
|---|
| *ID_torneo: SERIAL, PRIMARY KEY* |
| *descrizione: TEXT* |
| *sponsor: VARCHAR(100)* |
| *premi: VARCHAR(100)* |
| *restrizioni: TEXT* |

*Squadra:*

| |
|---|
| *nome: VARCHAR(100), PRIMARY KEY* |
| *descrizione: TEXT* |
| *colore_maglia: VARCHAR(50), NULL (opzionale)* |
| *n_max_giocatori: INTEGER, CHECK (n_max_giocatori > 0)* |
| *n_min_giocatori: INTEGER, CHECK (n_min_giocatori > 0 AND n_min_giocatori <= n_max_giocatori)* |

*Candidatura:*

| |
|---|
| *approvazione: BOOLEAN, CHECK (approvazione IN (TRUE, FALSE))* |

*Valutazione:*

| |
|---|
| *valutazione: INTEGER, CHECK (valutazione BETWEEN 0 AND 10)* |

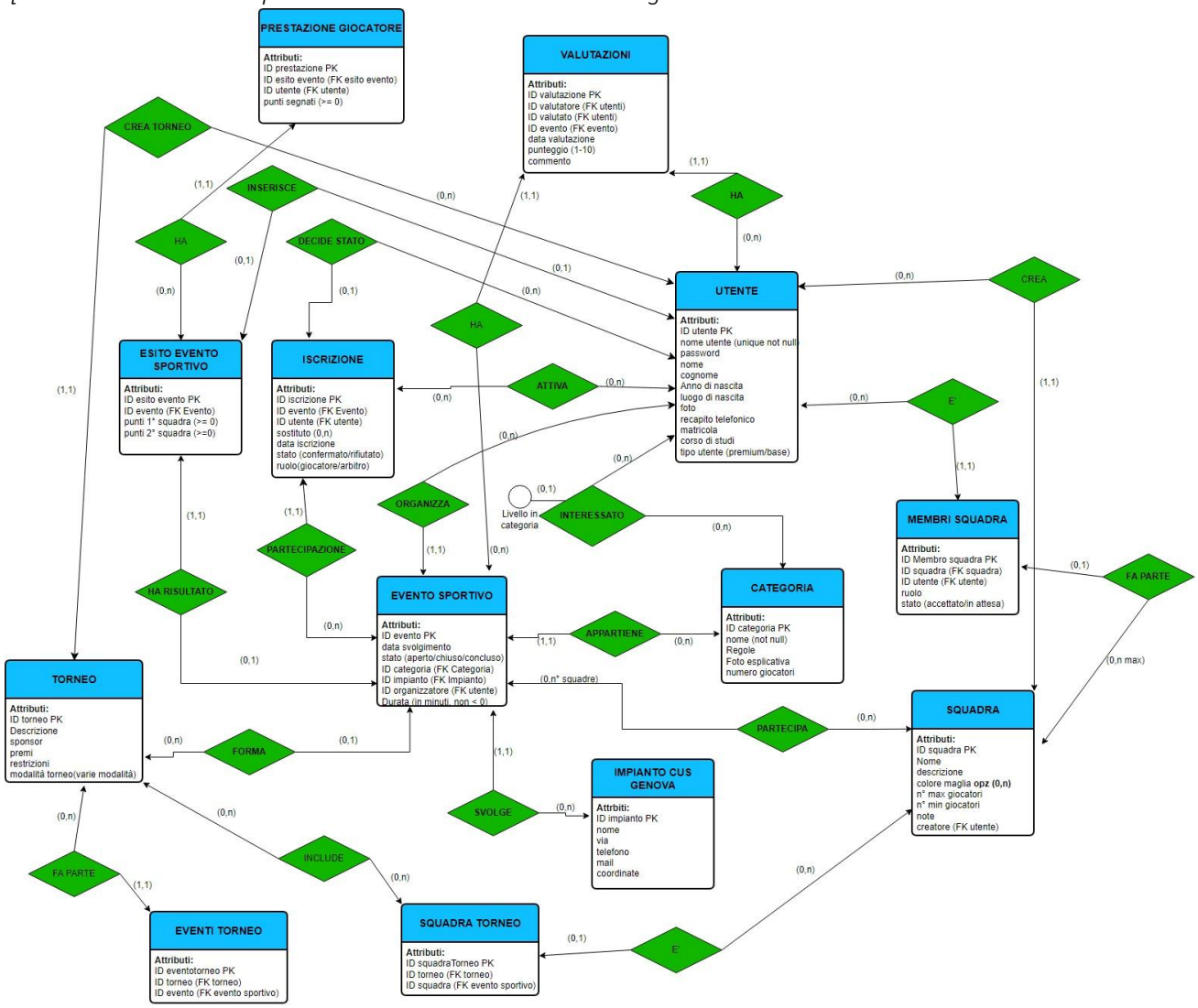
## VINCOLI NON ESPRIMIBILI NEL DIAGRAMMA

1) Numero Massimo di Giocatori per Evento:
Il numero totale di utenti giocatori confermati per un evento sportivo non può eccedere il numero di partecipanti previsti dalla categoria a cui appartiene l'evento.
2) Iscrizione a Eventi Chiusi:
Non è possibile iscriversi a eventi chiusi.
3) Gestione della Sede di un Evento:
Se la sede originale di un evento non è disponibile, viene assegnata una sede alternativa disponibile e meno utilizzata.
4) Aggiornamento del Livello dell'Utente:
Il livello di un utente, che indica il valore di un giocatore in relazione alle partite che ha disputato e ai voti che ha ricevuto, viene aggiornato automaticamente.

# 3. PROGETTO LOGICO

## SCHEMA ER RISTRUTTURATO

*[inserire qui il diagramma ER ristrutturato]*




## DOMINI DEGLI ATTRIBUTI

*[eventuali modifiche dei domini degli attributi e informazioni sui domini di eventuali attributi introdotti]*

*(avendo modificato un po' di cose riscrivo direttamente il dizionario degli attributi + il dominio aggiornati rispetto alla parte 1)*

***(Nome entità + attributi e dominio)***

***Utente:***

*ID utente: SERIAL, PRIMARY KEY*
*nome_utente: VARCHAR(50), UNIQUE, NOT NULL*
*password: VARCHAR(50), NOT NULL*
*nome: VARCHAR(50)*
*cognome: VARCHAR(50)*
*anno_nascita: INTEGER, CHECK (anno_nascita > 1900 AND anno_nascita <= EXTRACT(YEAR FROM CURRENT_DATE))*
*luogo_nascita: VARCHAR(100)*
*foto: BYTEA*
*telefono: VARCHAR(20)*
*matricola_studente: VARCHAR(20)*
*nome_corso: VARCHAR(100)*
*tipo_utente: VARCHAR(10), CHECK (tipo_utente IN ('premium', 'semplice'))*

*Categoria:*

*ID Categoria: SERIAL, PRIMARY KEY*
*nome: VARCHAR(50), NOT NULL*
*regolamento: TEXT*
*numero_giocatori: INTEGER, CHECK (numero_giocatori > 0)*
*foto_esempio: BYTEA*

*Impianto CUS Genova:*

*ID impianto: SERIAL, PRIMARY KEY*
*nome: VARCHAR(100), UNIQUE, NOT NULL*
*via: VARCHAR(100), NOT NULL*
*telefono: VARCHAR(20)*
*email: VARCHAR(100)*
*latitudine: DECIMAL(9,6), CHECK (latitudine BETWEEN -90 AND 90)*
*longitudine: DECIMAL(9,6), CHECK (longitudine BETWEEN -180 AND 180)*

*Evento Sportivo:*

*ID evento: SERIAL, PRIMARY KEY*
*data_evento: DATE*
*stato: VARCHAR(10), CHECK (stato IN ('APERTO', 'CHIUSO', 'CONCLUSO'))*
*ID categoria: INTEGER, REFERENCES Categoria(ID categoria)*
*ID impianto: INTEGER, REFERENCES Impianto_CUS_Genova(ID impianto)*
*ID organizzatore: INTEGER, REFERENCES Utente(ID utente)*
*durata_minuti: INTEGER, CHECK (durata_minuti > 0)*

*Iscrizione:*

*ID iscrizione: SERIAL, PRIMARY KEY*
*ID evento: INTEGER, REFERENCES Evento_Sportivo(ID evento)*
*ID utente: INTEGER, REFERENCES Utente(ID utente)*

*data_iscrizione: DATE*
*ruolo: VARCHAR(10), CHECK (ruolo IN ('giocatore', 'arbitro'))*
*stato: VARCHAR(10), CHECK (stato IN ('CONFERMATO', 'RIFIUTATO'))*

*Esito Evento Sportivo:*

*risultato_ID evento: SERIAL, PRIMARY KEY*
*ID evento: INTEGER, REFERENCES Evento_Sportivo(ID evento)*
*punteggio_squadra1: INTEGER, CHECK (punteggio_squadra1 >= 0)*
*punteggio_squadra2: INTEGER, CHECK (punteggio_squadra2 >= 0)*

*Torneo:*

*ID torneo: SERIAL, PRIMARY KEY*
*descrizione: TEXT*
*sponsor: VARCHAR(100)*
*premio: VARCHAR(100)*
*restrizioni_partecipazione: TEXT*
*modalita_torneo: VARCHAR(20), CHECK (modalita_torneo IN ('eliminazione diretta', 'gironi all'italiana', 'mista'))*

*Squadra:*

*ID squadra: SERIAL, PRIMARY KEY*
*nome: VARCHAR(100), UNIQUE, NOT NULL*
*descrizione: TEXT*
*colore_maglia: VARCHAR(50)*
*partecipanti_min: INTEGER, CHECK (partecipanti_min > 0)*
*partecipanti_max: INTEGER, CHECK (partecipanti_max >= partecipanti_min)*
*note: TEXT*
*creato_da: INTEGER, REFERENCES Utente(ID utente)*

*Membro squadra:*

*ID membro squadra: SERIAL, PRIMARY KEY*
*ID squadra: INTEGER, REFERENCES Squadra(ID squadra)*
*ID utente: INTEGER, REFERENCES Utente(ID utente)*
*ruolo: VARCHAR(50)*
*stato: VARCHAR(10), CHECK (stato IN ('accettato', 'in attesa'))*

*Prestazione Giocatore:*

*ID prestazione: SERIAL, PRIMARY KEY*
*esito: INTEGER, REFERENCES esito evento sportivo(ID esito evento)*
*ID utente: INTEGER, REFERENCES Utente(ID utente)*
*punti_segnati: INTEGER, CHECK (punti_segnati >= 0)*

*Evento Torneo:*

| |
|---|
| *ID eventoTorneo: SERIAL, PRIMARY KEY*<br>*ID torneo: INTEGER, REFERENCES Torneo(ID torneo)*<br>*ID evento: INTEGER, REFERENCES Evento_Sportivo(ID evento)* |

*SquadraTorneo:*

| |
|---|
| *ID squadra torneo: SERIAL, PRIMARY KEY*<br>*ID torneo: INTEGER, REFERENCES Torneo(ID torneo)*<br>*ID squadra: INTEGER, REFERENCES Squadra(ID squadra)* |

*Valutazione:*

| |
|---|
| *_ID valutazione: SERIAL, PRIMARY KEY*<br>*ID valutatore: INTEGER, REFERENCES Utente(ID utente)*<br>*ID valutato: INTEGER, REFERENCES Utente(ID utente)*<br>*ID evento: INTEGER, REFERENCES Evento_Sportivo(ID evento)*<br>*data_valutazione: DATE*<br>*punteggio: INTEGER, CHECK (punteggio BETWEEN 0 AND 10)*<br>*commento: TEXT* |

## VINCOLI

*[modifiche all'elenco di vincoli del modello concettuale (nuovi vincoli, eventuali vincoli eliminati o modificati)]*
*(i vincoli sono rimasti invariati)*

## RISTRUTTURAZIONE GERARCHIE

*Abbiamo eliminato l'entità sottoclasse di utente "utente premium", aggregandola semplicemente all'entità utente. Per farlo abbiamo aggiunto un campo dentro ad utente che distingue quelli premium da quelli base, un utente con campo impostato a premium avrà tutti i poteri di un utente premium*

## SCHEMA LOGICO

*[schema logico relazionale nella notazione vista a lezione]*

```sql
-- Table definitions
CREATE TABLE Utenti (
    utente_id SERIAL PRIMARY KEY,
    nome_utente VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(50) NOT NULL,
    nome VARCHAR(50),
    cognome VARCHAR(50),
    genere VARCHAR(1) CHECK (genere IN ('M', 'F')),
    anno_nascita INTEGER CHECK (anno_nascita > 1900 AND anno_nascita <= EXTRACT(YEAR FROM CURRENT_DATE)),
    luogo_nascita VARCHAR(100),
    foto BYTEA,
    telefono BIGINT CHECK (telefono >= 0000000000 AND telefono <= 9999999999),
    matricola_studente INTEGER CHECK (matricola_studente >= 0000000 AND matricola_studente <= 9999999),
    tipo_utente VARCHAR(10) CHECK (tipo_utente IN ('premium', 'semplice'))
);

CREATE TABLE Categorie (
    categoria_id SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    regolamento TEXT,
    numero_giocatori INTEGER CHECK (numero_giocatori > 0),
    foto_esempio BYTEA
);

CREATE TABLE Impianti (
    impianto_id SERIAL PRIMARY KEY,
    nome VARCHAR(100) UNIQUE NOT NULL,
    via VARCHAR(100) NOT NULL,
    telefono BIGINT CHECK (telefono >= 0000000000 AND telefono <= 9999999999),
    email VARCHAR(100),
    latitudine DECIMAL(9,6) CHECK (latitudine BETWEEN -90 AND 90),
    longitudine DECIMAL(9,6) CHECK (longitudine BETWEEN -180 AND 180)
);
CREATE TABLE Eventi (
    evento_id SERIAL PRIMARY KEY,
    data_evento DATE,
    stato VARCHAR(10) CHECK (stato IN ('APERTO', 'CHIUSO')),
    categoria_id INTEGER REFERENCES Categorie(categoria_id),
    impianto_id INTEGER REFERENCES Impianti(impianto_id),
    organizzatore_id INTEGER REFERENCES Utenti(utente_id),
    durata_minuti INTEGER CHECK (durata_minuti > 0)
);

CREATE TABLE Iscrizioni (
    iscrizione_id SERIAL PRIMARY KEY,
    evento_id INTEGER REFERENCES Eventi(evento_id),
    utente_id INTEGER REFERENCES Utenti(utente_id),
    data_iscrizione DATE,
    ruolo VARCHAR(10),
    stato VARCHAR(10) CHECK (stato IN ('CONFERMATO', 'RIFIUTATO'))
);

CREATE TABLE Tornei (
    torneo_id SERIAL PRIMARY KEY,
    descrizione TEXT,
    sponsor TEXT,
    premio TEXT,
    restrizioni_partecipazione TEXT,
    modalita_torneo VARCHAR(20) CHECK (modalita_torneo IN ('eliminazione diretta', 'gironi all'italiana', 'mista'))
);

CREATE TABLE Squadre (
    squadra_id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    colore_maglia VARCHAR(50),
    partecipanti_min INTEGER CHECK (partecipanti_min > 0),
    partecipanti_max INTEGER CHECK (partecipanti_max >= partecipanti_min),
    descrizione TEXT,
    note TEXT,
    creato_da INTEGER REFERENCES Utenti(utente_id)
);
```

```sql
CREATE TABLE MembriSquadra (
    membro_squadra_id SERIAL PRIMARY KEY,
    squadra_id INTEGER REFERENCES Squadre(squadra_id),
    utente_id INTEGER REFERENCES Utenti(utente_id),
    ruolo VARCHAR(50),
    stato VARCHAR(10) CHECK (stato IN ('accettato', 'in attesa'))
);

CREATE TABLE RisultatiEvento (
    risultato_evento_id SERIAL PRIMARY KEY,
    evento_id INTEGER REFERENCES Eventi(evento_id),
    punteggio_squadra1 INTEGER CHECK (punteggio_squadra1 >= 0),
    punteggio_squadra2 INTEGER CHECK (punteggio_squadra2 >= 0)
);

CREATE TABLE PrestazioniGiocatore (
    prestazione_id SERIAL PRIMARY KEY,
    risultato_evento_id INTEGER REFERENCES RisultatiEvento(risultato_evento_id),
    utente_id INTEGER REFERENCES Utenti(utente_id),
    punti_segnati INTEGER CHECK (punti_segnati >= 0)
);

CREATE TABLE Valutazioni (
    valutazione_id SERIAL PRIMARY KEY,
    valutatore_id INTEGER REFERENCES Utenti(utente_id),
    valutato_id INTEGER REFERENCES Utenti(utente_id),
    evento_id INTEGER REFERENCES Eventi(evento_id),
    data_valutazione DATE,
    punteggio INTEGER CHECK (punteggio BETWEEN 0 AND 10),
    commento TEXT
);


CREATE TABLE Livelli (
    livello_id SERIAL PRIMARY KEY,
    utente_id INTEGER REFERENCES Utenti(utente_id),
    sport VARCHAR(50) NOT NULL,
    livello INTEGER CHECK (livello BETWEEN 0 AND 100) DEFAULT 60,
    data_aggiornamento DATE DEFAULT CURRENT_DATE
);


CREATE TABLE EventiTorneo (
    evento_torneo_id SERIAL PRIMARY KEY,
    torneo_id INTEGER REFERENCES Tornei(torneo_id),
    evento_id INTEGER REFERENCES Eventi(evento_id)
);

CREATE TABLE SquadreTorneo (
    squadra_torneo_id SERIAL PRIMARY KEY,
    torneo_id INTEGER REFERENCES Tornei(torneo_id),
    squadra_id INTEGER REFERENCES Squadre(squadra_id)
);
```

## VERIFICA DI QUALITÀ DELLO SCHEMA

*[verifica delle forme normali ed eventuali ottimizzazioni applicate tenendo in considerazione il carico di lavoro]*

1) *Tutte le tabelle che abbiamo creato rispettano la 3NF, ovvero tutti gli attributi non chiave sono direttamente dipendenti dalla chiave primaria e non ci sono dipendenze transitive.*

## SCHEMA SQL IN FORMA GRAFICA

*[diagramma che visualizza lo script SQL di creazione dello schema in forma grafica ottenuto con DataGrip (vedi Aulabweb per come crearla dallo script)]*