Progetto Basi di Dati

creato da:
Alberto de Agostini
mat: 579021

ABSTRACT

Il progetto mira alla modellazione e alla gestione di un Campionato di pallavolo.
E' stata realizzata una interfaccia web minimale per l'interazione con gli utenti (giocatori o non), in base al tipo di utente sono possibili diverse operazioni (i giocatori possono 'chiudere' le partite effettuate).
Esiste poi l'utente admin che ha pieni poteri sul DB.

ANALISI REQUISITI

Come detto in precedenza il progetto mira alla gestione di un campionato di pallavolo (quasi identico ad un campionato di qualsiasi altro sport, con minime modifiche andrebbe bene per altri sport), la gestione di esso e' realizzata tramite l'interfaccia web in cui i giocatori registrati 'mandano avanti' il campionato.
Una delle e' **giocatore**, di un giocatore ci interessa il nome, il cognome, l'indirizzo, l'ingaggio, il suo ruolo preferito ed e' identificato da il suo codice fiscale,
attraverso questo ultimo gli **utenti** che si registrano al sito possono mostrare di essere il giocatore inserendo tale CF nel sito, una volta inserito viene collegato l'utente al giocatore nel database e automaticamente vengono rese possibli azioni come inserire il punteggio delle partite effettuate.
Ogni giocatore gioca in una **squadra**, questa entita' e' forse la principale del database, essa e' identificata da il nome, in piu' conosceremo la **societa'** di cui fa parte e la **palestra** in cui gioca.
Ad ogni squadra e' connesso anche il **personale squadra** che identifica tutte quelle persone che lavorano in essa, questi si suddividono in 'manager','allenatore','presidente'.
Un altra entita' molto importante e' **partita**, ogni partita e' identificata da un ID, poi contiene la data, la **squadra** di casa, la **squadra** in trasferta, la **palestra** in cui viene giocata, il risultato e **l'arbitro** che la diretta.
Si e' scelto di dividere le partite in **partite da giocare** e **partite giocate,** le partite giocate contengono il risultato di essa, mentre quelle da giocare non hanno i campi per il risultato, effettuando questa suddivisione risulta piu' facile l'aggiornamento della **classifica**, essa infatti contiene per ogni **squadra** il punteggio, i set vinti, quelli persi le partite vinte e quelle perse.

SCHEMA CONCETTUALE

lista delle classi:

- admin: modella il tipo di utente amministratore.
  - username varchar
  - password varchar

- arbitro: modella l'arbitro che dirige le varie partite.

    - CF char(16)
    - nome varchar
    - cognome varchar

- classifica: rappresenta la classifica del campionato.
    - squadra varchar
    - punteggio smallint unsigned
    - pGiocate smallint unsigned
    - pVinte smallint unsigned
    - pPerse smallint unsigned
    - setVinti smallint unsigned
    - setPersi smallint unsigned

- giocatore: rappresenta un giocatore di una squadra del campionato.
    - CF codice fiscale char(16)
    - nome varchar
    - cognome varchar
    - indirizzo varchar
    - ingaggio int unsigned
    - ruolo varchar

- palestra: modella una palestra in cui gioca qualche squadra.
    - Indirizzo varchar
    - nome varchar
    - capienza int unsigned

- partita: modella una partita del campionato.
    - Id int
    - data date
    - locali varchar
    - ospiti varchar
    - palestra varchar
    - arbitro char(16)

la partita si suddivide in 2 sottoclassi
- partitadagiocare che ha gli stessi attributi
- partitagiocata
    - setlocali smallint unsigned
    - setospiti smallint unsigned

- personalesquadra rappresenta le varie persone che lavorano nella squadra
    - |CF char(16)
    - nome varchar
    - cognome varchar
    - Generalizzazione per il ruolo svolto

    - manager
    - presidente
    - allenatore

- societa: rappresenta una societa che possiede una squadra
    - nome varchar

- sponsor: sponsor di una squadra
    - nome varchar
    - importo_donazione int unsigned

- squadra: rappresenta una squadra del campionato
    - nome varchar
    - societa varchar
    - indirizzopalestra varchar

- utente: modella un utente che si registra nel sito
    - username varchar
    - password varchar

lista delle associazioni

utente – giocatore: registrato come
ad ogni utente puo' essere associato un giocatore, ad ogni giocatore puo' essere associato uno ed un solo utente. 1:1.
Un giocatore puo' non essere registrato al sito quindi non e' totale da giocatore a utente, inoltre un utente puo' non essere un giocatore quindi non e' totale in nessun senso.

giocatore – squadra: contratto
Ogni giocatore gioca in una squadra, in una squadra giocano piu' giocatori. 1:N.
Un giocatore puo' non giocare in nessuna squadra mentre in ogni squadra devono esserci giocatori.

Squadra – sponsor: sponsorizzata da
ogni squadra puo' essere sponsorizzata da piu' sponsor, uno sponsor puo' sponsorizzare piu' squadre. N:M.
Non totale in entrambi i sensi.

Squadra – personalesquadra: lavora in
ogni squadra puo' avere piu' personale che ci lavora, un personale e' di una sola squadra. N:1.
Una squadra puo' non avere personale (squadra autogestita) e un personale puo' non lavorare in nessuna squadra.

Squadra – societa: appartiene
ogni squadra e' posseduta da una societa', una societa' puo' possedere piu' squadre. 1:N.

Squadra – palestra: casa
una squadra ha una palestra come 'casa'. In una palestra giocano piu' squadre in casa. 1:N.
Totale in entrambi i sensi.

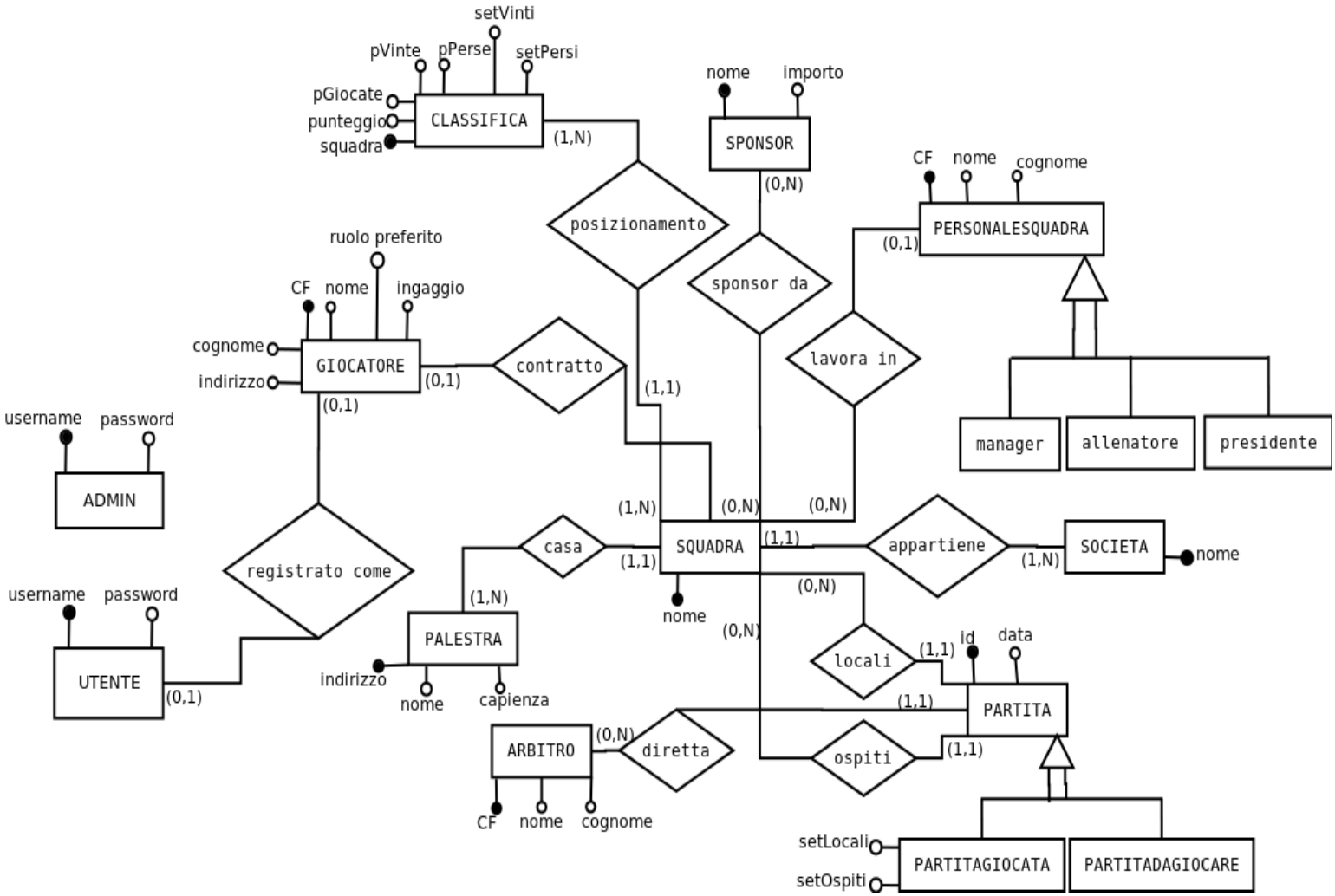
Squadra – partita: locali
una squadra gioca piu' partite come locale. In una partita c'e' una squadra locale. N:1.
Non totale da squadra a partita.

Squadra – partita: ospiti
identica all'associazione sopra, in questo caso si rappresenta la squadra che gioca come ospite.

Squadra – classifica: posizionamento
una squadra e' posizionata in una classifica, nella classifica ci sono tutte le squadre iscritte. 1:N.
Associazione totale.

Partita – arbitro: diretta
una partita e' diretta da un arbitro, un arbitro puo' dirigere piu' partite. 1:N.
Totale da partita a arbitro, non inversamente.

MODELLO E-R



Le chiavi esterne non sono state inserite nell diagramma per non complicarne troppo la lettura, le elenco qui sotto:
UTENTE CF REFERENCES GIOCATORE CF
GIOCATORE squadra REFERENCES SQUADRA nome
SQUADRA palestra REFERENCES PALESTRA indirizzo
SQUADRA societa REFERENCES SOCIETA nome
PERSONALESQUADRA squadra REFERENCES SQUADRA nome
SQUADRA squadra REFERENCES SQUADRA nome
PARTITA locali REFERENCES SQUADRA nome
PARTITA ospiti REFERENCES SQUADRA nome
PARTITA arbitro REFERENCES ARBITRO CF
PARTITA palestra REFERENCES PALESTRA indirizzo
CLASSIFICA squadra REFERENCES SQUADRA nome

PROGETTAZIONE LOGICA

L'entita' personalesquadra ho scelto di modellarla come una entita' unica, raggruppando i vari tipi di personale all'interno di un attributo chiamato ruolo.
Nessun cambio per le associazioni.

L'entita partita ho scelto di dividerla in 2 entita', partitadagiocare e partitagiocata.
Esse differiscono per soltanto 2 attributi infatti partitagiocata contiene il risultato della partita (setLocali e setOspiti), tuttavia risulta essere lo schema piu' chiaro e piu' facile la modellazione e il mantenimento del DB per simulare l'avanzamento del campionato.
Entrambe queste nuove entita' hanno le stesse associazioni che aveva l'entita' partita.

Il diagramma rimane lo stesso apparte i cambi riguardanti le generalizzazioni appena descritti.

BASE DI DATI (CODICE SQL)

Riporto qui sotto il codice SQL scritto per la creazione delle tabelle.
La tabella ERROR non descritta in precedenza serve solo per la gestione di qualche errore o per debug, puo' essere utile anche all'amministratore.

```
DROP TABLE IF EXISTS CLASSIFICA;
DROP TABLE IF EXISTS PARTITADAGIOCARE;
DROP TABLE IF EXISTS PARTITAGIOCATA;
DROP TABLE IF EXISTS PERSONALESQUADRA;
DROP TABLE IF EXISTS UTENTE;
DROP TABLE IF EXISTS GIOCATORE;
DROP TABLE IF EXISTS SPONSORSQUADRA;
DROP TABLE IF EXISTS SPONSOR;
DROP TABLE IF EXISTS SQUADRA;
DROP TABLE IF EXISTS ARBITRO;
DROP TABLE IF EXISTS PALESTRA;
DROP TABLE IF EXISTS SOCIETA;
DROP TABLE IF EXISTS ADMIN;
DROP TABLE IF EXISTS ERROR;


CREATE TABLE IF NOT EXISTS ADMIN(
  `username` VARCHAR(20) NOT NULL,
  `password` VARCHAR(20) NOT NULL,
  PRIMARY KEY(`username`)
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS ERROR(
  `msg` VARCHAR(255)
```

```
)ENGINE=InnoDB;


CREATE TABLE IF NOT EXISTS SOCIETA(
  `nome`  VARCHAR(25) NOT NULL,
  PRIMARY KEY(`nome`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS PALESTRA(
  `indirizzo`  VARCHAR(50) NOT NULL,
  `nome`  VARCHAR(25) NOT NULL,
  `capienza`  SMALLINT UNSIGNED,
  PRIMARY KEY(`indirizzo`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS ARBITRO(
  `CF`  CHAR(16) NOT NULL,
  `nome`  VARCHAR(25) NOT NULL,
  `cognome`  VARCHAR(25) NOT NULL,
  PRIMARY KEY(`CF`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS SPONSOR(
  `nome`  VARCHAR(25) NOT NULL,
  `importo_donazione`   INT UNSIGNED,
  PRIMARY KEY(`nome`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS SQUADRA(
  `nome`   VARCHAR(25) NOT NULL,
  `societa`  VARCHAR(25),
  `indirizzoPal`  VARCHAR(50),
  PRIMARY KEY(`nome`),
  FOREIGN KEY(`societa`)
  REFERENCES SOCIETA(`nome`)
  ON DELETE SET NULL
  ON UPDATE CASCADE,
  FOREIGN KEY(`indirizzoPal`)
  REFERENCES PALESTRA(`indirizzo`)
  ON DELETE SET NULL
  ON UPDATE CASCADE
)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS SPONSORSQUADRA(
  `sponsor`  VARCHAR(25) NOT NULL,
  `squadra`  VARCHAR(25) NOT NULL,
  PRIMARY KEY(`squadra`,`sponsor`),
  FOREIGN KEY(`squadra`) REFERENCES SQUADRA(`nome`)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY(`sponsor`) REFERENCES SPONSOR(`nome`)
  ON DELETE RESTRICT ON UPDATE CASCADE
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS GIOCATORE(
  `CF`  CHAR(16) NOT NULL,
  `nome`   VARCHAR(25) NOT NULL,
  `cognome`   VARCHAR(25) NOT NULL,
  `indirizzo`   VARCHAR(50),
  `ingaggio`   INT UNSIGNED,
  `ruoloPreferito`  VARCHAR(20),
  `squadra`   VARCHAR(25),
  PRIMARY KEY(`CF`),
  FOREIGN KEY(`squadra`) REFERENCES SQUADRA(`nome`)
  ON DELETE SET NULL ON UPDATE CASCADE
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS UTENTE(
  `username` VARCHAR(20) NOT NULL,
  `password` VARCHAR(20) NOT NULL,
  `CodFisc`  CHAR(16),
  PRIMARY KEY(`username`),
  FOREIGN KEY(`CodFisc`) REFERENCES GIOCATORE(`CF`)
  ON DELETE SET NULL ON UPDATE CASCADE
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS PERSONALESQUADRA(
  `CF`  CHAR(16) NOT NULL,
  `nome`  VARCHAR(25) NOT NULL,
  `cognome`  VARCHAR(25) NOT NULL,
  `Ruolo`  ENUM('manager','allenatore','presidente'),
  `squadra`   VARCHAR(25),
  PRIMARY KEY (`CF`),
  FOREIGN KEY(`squadra`) REFERENCES SQUADRA(`nome`)
  ON DELETE SET NULL ON UPDATE CASCADE
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS PARTITAGIOCATA(
  `id`  INT UNSIGNED,
```

```
  `data`  DATE NOT NULL,
  `locali`  VARCHAR(25) NOT NULL,
  `ospiti`  VARCHAR(25) NOT NULL,
  `palestra`  VARCHAR(50) NOT NULL,
  `arbitro`  CHAR(16) NOT NULL,
  `setLocali`  SMALLINT UNSIGNED NOT NULL,
  `setOspiti`  SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY(`id`),
  FOREIGN KEY(`locali`) REFERENCES SQUADRA(`nome`),
  FOREIGN KEY(`ospiti`) REFERENCES SQUADRA(`nome`),
  FOREIGN KEY(`arbitro`) REFERENCES ARBITRO(`CF`)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY(`palestra`) REFERENCES PALESTRA(`indirizzo`)
  ON DELETE RESTRICT ON UPDATE CASCADE
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS PARTITADAGIOCARE(
  `id`  INT UNSIGNED,
  `data`  DATE NOT NULL,
  `locali`  VARCHAR(25) NOT NULL,
  `ospiti`  VARCHAR(25) NOT NULL,
  `palestra`  VARCHAR(50) NOT NULL,
  `arbitro`  CHAR(16) NOT NULL,
  PRIMARY KEY(`id`),
  FOREIGN KEY(`locali`) REFERENCES SQUADRA(`nome`)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY(`ospiti`) REFERENCES SQUADRA(`nome`)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY(`arbitro`) REFERENCES ARBITRO(`CF`)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY(`palestra`) REFERENCES PALESTRA(`indirizzo`)
  ON DELETE RESTRICT ON UPDATE CASCADE
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS CLASSIFICA(
  `squadra`  VARCHAR(25) NOT NULL,
  `punteggio`  SMALLINT UNSIGNED NOT NULL DEFAULT 0,
  `pGiocate`  SMALLINT UNSIGNED NOT NULL DEFAULT 0,
  `pVinte`  SMALLINT UNSIGNED NOT NULL DEFAULT 0,
  `pPerse`  SMALLINT UNSIGNED NOT NULL DEFAULT 0,
  `setVinti`  SMALLINT UNSIGNED NOT NULL DEFAULT 0,
  `setPersi`  SMALLINT UNSIGNED NOT NULL DEFAULT 0,
  PRIMARY KEY(`squadra`),
  FOREIGN KEY(`squadra`) REFERENCES SQUADRA(`nome`)
  ON DELETE CASCADE ON UPDATE CASCADE
)
```

ENGINE = InnoDB;

TRIGGERS

Di seguito riporto il codice scritto per la definizione dei triggers con una breve descrizione. Molti di questi trigger sono appositi per le procedure definite dopo che l'utente potra' chiamare dall'interfaccia web.

- aggiungiSquadraAllaClassifica
  Questo trigger fa in modo che ogniqualvolta una squadra si iscrive (viene aggiunta al database) automaticamente viene inserita nella classifica.
- CheckUpdatedData
  Questo trigger serve per controllare che si possa effettivamente spostare una partita (tramite procedura ad esempio), se nella data inserita per la partita non si puo' giocare (perche' la palestra e' gia' occupata da un altra partita ad esempio) ne impedisce l'aggiornamento.
- CheckRisultato
  Questo trigger controlla quando ad una partita viene assegnato un risultato se e' un risultato valido.
- CheckScadenza
  Questo trigger controlla che non si possano aggiungere squadre durante un certo periodo (in questo caso fissato dal 15 settembre del corrente anno fino al 01 giugno dell'anno successivo, serve per simulare la chiusura iscrizioni squadre durante il campionato).
- AggClassifica
  Questo ultimo trigger serve per automatizzare la tabella classifica, ogni volta che una partita da giocare viene chiusa essa diventa una partita giocata e in base al risultato inserito la classifica viene aggiornata proprio attraverso questo trigger.

```
/* ogniqualvolta inserisco una squadra la aggiungo alla classifica */
DROP TRIGGER IF EXISTS aggiungiSquadraAllaClassifica;
CREATE TRIGGER aggiungiSquadraAllaClassifica
AFTER insert ON SQUADRA
FOR EACH ROW
INSERT INTO CLASSIFICA values(new.nome, 0, 0, 0, 0, 0, 0);

/* trigger per controllare che la data della partita creata sia accettabile */
/* in caso contrario la partita viene settata per essere giocata tra 2 settimane */
DROP TRIGGER IF EXISTS checkData;
DELIMITER $
CREATE TRIGGER checkData
BEFORE INSERT ON PARTITADAGIOCARE FOR EACH ROW
BEGIN
  DECLARE oldData DATE;
  SET oldData = NEW.data;
  IF (oldData < CURDATE()) THEN
  SET oldData = CURDATE();
  /* se la data della partita non e' valida la metto tra 2 settimane */
```

```
  SET oldData = DATE_ADD(oldData, INTERVAL 2 WEEK);
  SET NEW.data = oldData;
  END IF;
END $
DELIMITER ;

/* come il trigger precedente ma before update invece che before insert */
DROP TRIGGER IF EXISTS checkUpdatedData;
DELIMITER $
CREATE TRIGGER checkUpdatedData
BEFORE UPDATE ON PARTITADAGIOCARE FOR EACH ROW
BEGIN
  DECLARE oldData DATE;
  SET oldData = NEW.data;
  IF (oldData < CURDATE()) THEN
  SET oldData = CURDATE();
/* se la data della partita non e' valida la metto tra 2 settimane */
  SET oldData = DATE_ADD(oldData, INTERVAL 2 WEEK);
  INSERT INTO ERROR VALUES(CONCAT("Data non valida, la partita e' stata spostata in
data ",oldData));
  SET NEW.data = oldData;
  END IF;
END $
DELIMITER ;


/* trigger per controllare la validita' dei punteggi immessi in una partita giocata */
DROP TRIGGER IF EXISTS checkRisultato;
DELIMITER $
CREATE TRIGGER checkRisultato
BEFORE INSERT ON PARTITAGIOCATA
FOR EACH ROW
BEGIN
  DECLARE msg VARCHAR(80);
  SET msg="Punteggio non valido, possibili combinazioni sono 3-0 3-1 3-2 2-3 1-3 0-3";
  IF ((NEW.setLocali + NEW.setOspiti)<3 OR (NEW.setLocali + NEW.setOspiti)>5) THEN
    INSERT INTO ERROR VALUES(msg);
    CALL punteggio_inserito_non_valido ; /* impedisco l'inserimento */
  ELSE
  IF ((NEW.setLocali=2 AND NEW.setOspiti=2)) THEN
    SET msg= "2=2";
    INSERT INTO ERROR VALUES(msg);
    CALL punteggio_inserito_non_valido ; /* impedisco l'inserimento */
  END IF;
END IF;
END $
DELIMITER ;
```

```
/* trigger per impedire l'inserimento di una squadra se e' oltre la data di termine iscrizioni (1 settembre) */
DROP TRIGGER IF EXISTS checkScadenza;
DELIMITER $
CREATE TRIGGER checkScadenza
BEFORE INSERT ON SQUADRA
FOR EACH ROW
BEGIN
  DECLARE dataOggi DATE;
  DECLARE annoOggi SMALLINT;
  DECLARE scadenza DATE;
  DECLARE riapertura DATE;
  DECLARE msg VARCHAR(50);
  SET dataOggi = CURDATE();
  SET annoOggi = YEAR(dataOggi);
  SET scadenza = CONCAT(annoOggi,"-09-15");
  SET annoOggi = annoOggi+1;                    /* annoOggi diventa annoProssimo */
  SET riapertura = CONCAT(annoOggi,"-06-01");
  SET msg = "Errore: scadenza iscrizioni squadre 1 settembre";
  IF (dataOggi>scadenza AND dataOggi<riapertura) THEN
  INSERT INTO ERROR VALUES(msg);
    CALL non_puoi_iscrivere_squadre_iscrizioni_gia_chiuse ; /* impedisco l'inserimento */
  END IF;
END $
DELIMITER ;

/* trigger per automatizzare l'aggiornamento della classifica, ogni volta che viene
   'chiusa' una partita la classifica si aggiornera' in base al punteggio messo e la
   partita relativa in partitadagiocare viene eliminata */
DROP TRIGGER IF EXISTS aggClassifica;
DELIMITER $
CREATE TRIGGER aggClassifica
AFTER INSERT ON PARTITAGIOCATA
FOR EACH ROW
BEGIN
  DECLARE puntiA SMALLINT;
  DECLARE puntiB SMALLINT;
  DECLARE casi TINYINT UNSIGNED;
  IF NEW.setLocali=3 THEN
    IF NEW.setOspiti<2 THEN
      SET puntiA = 3;
      SET puntiB = 0;
      SET casi=1;
    ELSE
      SET puntiA = 2;
      SET puntiB = 1;
      SET casi=2;
```

```
    END IF;
  ELSE
    IF NEW.setLocali<2 THEN
      SET puntiB = 3;
      SET puntiA = 0;
      SET casi=3;
    ELSE
      SET puntiB = 2;
      SET puntiA = 1;
      SET casi=4;
    END IF;
  END IF;
/* aggiorno effettivamente la classifica */
  CASE casi
  WHEN 1 THEN
  UPDATE CLASSIFICA
  SET
punteggio=punteggio+3,pGiocate=pGiocate+1,pVinte=pVinte+1,setVinti=setVinti+NEW.set
Locali,setPersi=setPersi+NEW.setOspiti
  WHERE squadra=NEW.locali;
  UPDATE CLASSIFICA
  SET
pGiocate=pGiocate+1,pPerse=pPerse+1,setVinti=setVinti+NEW.setOspiti,setPersi=setPersi
+NEW.setLocali
  WHERE squadra=NEW.ospiti;
  WHEN 2 THEN
  UPDATE CLASSIFICA
  SET
punteggio=punteggio+2,pGiocate=pGiocate+1,pVinte=pVinte+1,setVinti=setVinti+NEW.set
Locali,setPersi=setPersi+NEW.setOspiti
  WHERE squadra=NEW.locali;
  UPDATE CLASSIFICA
  SET
punteggio=punteggio+1,pGiocate=pGiocate+1,pPerse=pPerse+1,setVinti=setVinti+NEW.set
Ospiti,setPersi=setPersi+NEW.setLocali
  WHERE squadra=NEW.ospiti;
  WHEN 3 THEN
  UPDATE CLASSIFICA
  SET
pGiocate=pGiocate+1,pPerse=pPerse+1,setVinti=setVinti+NEW.setLocali,setPersi=setPersi
+NEW.setOspiti
  WHERE squadra=NEW.locali;
  UPDATE CLASSIFICA
  SET
punteggio=punteggio+3,pGiocate=pGiocate+1,pVinte=pVinte+1,setVinti=setVinti+NEW.set
Ospiti,setPersi=setPersi+NEW.setLocali
  WHERE squadra=NEW.ospiti;
  WHEN 4 THEN
```

```
  UPDATE CLASSIFICA
  SET
punteggio=punteggio+1,pGiocate=pGiocate+1,pPerse=pPerse+1,setVinti=setVinti+NEW.set
Locali,setPersi=setPersi+NEW.setOspiti
  WHERE squadra=NEW.locali;
  UPDATE CLASSIFICA
  SET
punteggio=punteggio+2,pGiocate=pGiocate+1,pVinte=pVinte+1,setVinti=setVinti+NEW.set
Ospiti,setPersi=setPersi+NEW.setLocali
  WHERE squadra=NEW.ospiti;
  END CASE;
/* elimino la partita appena giocata da quelle da giocare */
  DELETE FROM PARTITADAGIOCARE WHERE locali=NEW.locali AND
ospiti=NEW.ospiti AND data=NEW.data;
END $
DELIMITER ;
```

PROCEDURE

Elenco qui sotto il codice scritto per le procedure.

- Spostapartita
  Questa procedura simula la richiesta di spostare una partita
- creapartita
  Tramite questa procedura si crea una partitadagiocare
- chiudipartita
  Tramite questa procedura si chiude una partita, ovvero si toglie dalla tabella partitadagiocare una partita e se ne crea una in partitagiocata con gli stessi attributi piu' il risultato messo
- login
  Serve per provare a fare il login, ritorna tutti i dati necessari alla sessione per l'interfaccia web

```
/* sposta partita serve per modellare la richiesta di spostare una determinatata partita,
   tramite trigger viene controllato che la palestra nella data richiesta sia libera
   in caso opposto viene scritto un errore nella tabella ERROR       */
DROP PROCEDURE IF EXISTS spostaPartita;
DELIMITER $
CREATE PROCEDURE spostaPartita(IN squadraA VARCHAR(25), squadraB
VARCHAR(25), dataVecchia DATE, dataNuova DATE)
BEGIN
  DECLARE idPartita INT UNSIGNED;
  DECLARE idTemp INT UNSIGNED;
  DECLARE palestraP VARCHAR(50);
  SELECT id, palestra INTO idPartita, palestraP FROM PARTITADAGIOCARE WHERE
  locali=squadraA AND ospiti=squadraB AND data=dataVecchia;
  SELECT id INTO idTemp FROM PARTITADAGIOCARE WHERE
```

```
  locali=squadraA AND ospiti=squadraB AND data=dataNuova AND palestra=palestraP;
  IF idPartita IS NOT NULL THEN
    IF idTemp IS NULL THEN
      UPDATE PARTITADAGIOCARE SET data=dataNuova WHERE id=idPartita;
    ELSE
      INSERT INTO ERROR VALUES('palestra gia occupata quel giorno');
    END IF;
  ELSE
    INSERT INTO ERROR VALUES('nessuna partita da spostare in quella data tra le due
squadre');
  END IF;
END;
$
DELIMITER ;

/* chiamando crea partita si inserisce una row sulla tabella partitadagiocare con squadraA
   squadraB in data scelta (se possibile), il resto dei dati e' auto compilato */
DROP PROCEDURE IF EXISTS creaPartita;
DELIMITER $
CREATE PROCEDURE creaPartita(IN squadraA VARCHAR(25), squadraB
VARCHAR(25), dataPartita DATE)
BEGIN
  DECLARE luogoPartita VARCHAR(50);
  DECLARE idPartita INT;
  DECLARE arbitroPartita CHAR(16);
  -- prendo l'indirizzo della squadra di casa --
  SELECT indirizzoPal INTO luogoPartita
  FROM SQUADRA WHERE nome=squadraA;
  -- aumento sempre l'identificatore della partita --
  SELECT MAX(id) INTO idPartita
  FROM PARTITADAGIOCARE;
  SET idPartita=idPartita+1;
  IF idPartita IS NULL THEN
    SET idPartita=0;
  END IF;
-- ad ogni partita assegno un arbitro casuale --
  SELECT CF INTO arbitroPartita
  FROM ARBITRO ORDER BY RAND() LIMIT 1;
  INSERT INTO PARTITADAGIOCARE VALUES(
    idPartita, dataPartita, squadraA, squadraB, luogoPartita, arbitroPartita    );
END;
$
DELIMITER ;


/* chiamando chiudiPartita si va a togliere una entry da PARTITEDAGIOCARE e la si
   inserisce con il risultato richiesto in PARTITEGIOCATE. Qualche trigger si occupera'
   di controllare che i dati immessi siano validi e di aggiornare la classifica */
```

```
DROP PROCEDURE IF EXISTS chiudiPartita;
DELIMITER $
CREATE PROCEDURE chiudiPartita(IN squadraA VARCHAR(25), squadraB
VARCHAR(25), dataPartita DATE, punteggioA SMALLINT UNSIGNED, punteggioB
SMALLINT UNSIGNED)
BEGIN
  DECLARE idPartita INT;
  DECLARE palestraP VARCHAR(25);
  DECLARE arbitroP  CHAR(16);
  DECLARE nuovoId   INT;
  /* recupero tutti i dati che mi servono della partita appena giocata */
  SELECT                                          id, palestra, arbitro INTO idPartita,
palestraP, arbitroP
  FROM PARTITADAGIOCARE
  WHERE locali=squadraA AND ospiti=squadraB AND data=dataPartita;
  IF idPartita IS NOT NULL THEN
  /* id autoincrementante */
    SELECT MAX(id) INTO nuovoId
    FROM PARTITAGIOCATA;
    SET nuovoId=nuovoId+1;
    IF nuovoId IS NULL THEN
      SET nuovoId=0;
    END IF;
/* inserisco i dati come partita giocata (un trigger si occupera' di eliminare la entry relativa
alla partitadagiocare*/
    INSERT INTO PARTITAGIOCATA VALUES(
      nuovoId, dataPartita, squadraA, squadraB, palestraP, arbitroP, punteggioA, punteggioB
    );
    ELSE
      INSERT INTO ERROR VALUES('nessuna partita da chiudere in quella data tra le due
squadre');
   END IF;
END;
$
DELIMITER ;

/* procedura per controllare se un login e' valido, ritorna vari dati dell'utente */
DROP PROCEDURE IF EXISTS login;
DELIMITER $
CREATE PROCEDURE login
(IN nomeUtente VARCHAR(20), IN passwordUtente VARCHAR(20), OUT  esito BOOL,
OUT admin BOOL, OUT giocatore BOOL, OUT usernameU VARCHAR(20), OUT
squadraU VARCHAR(25), OUT codfiscU CHAR(16))
BEGIN
  SET esito=FALSE;
  SET admin=FALSE;
  SELECT COUNT(*)>0 INTO admin FROM ADMIN
  WHERE username=nomeUtente AND password=passwordUtente;
```

```
  IF(admin) THEN
     SET admin=TRUE;
     SET esito=TRUE;
  ELSE
    SELECT COUNT(*)>0 INTO esito FROM UTENTE
    WHERE username=nomeUtente AND password=passwordUtente;
    IF(esito) THEN
      SET esito=TRUE;
      SET usernameU=nomeUtente;
      SELECT CodFisc INTO codfiscU FROM UTENTE
      WHERE username=nomeUtente;
      IF codfiscU IS NOT NULL THEN
        SET giocatore=TRUE;
        SELECT SQUADRA.nome INTO squadraU FROM GIOCATORE,SQUADRA
        WHERE GIOCATORE.CF=codfiscU AND GIOCATORE.squadra=SQUADRA.nome;
      END IF;
    END IF;
  END IF;
END;
$
DELIMITER ;
```

QUERIES

1.
Societa' che ha avuto piu' introiti dagli sponsor:

```
SELECT sum(importo_donazione) as TOTALE, SOCIETA.nome
FROM SOCIETA, SQUADRA, SPONSORSQUADRA,SPONSOR
WHERE SQUADRA.societa=SOCIETA.nome AND
SQUADRA.nome=SPONSORSQUADRA.squadra AND
SPONSOR.nome=SPONSORSQUADRA.sponsor;
```

output:
34000 marsango

2.
L'arbitro che ha diretto piu' partite:

```
SELECT data, locali, ospiti, nome,cognome, count(CF) as numpartite from
PARTITAGIOCATA, ARBITRO where CF=arbitro
GROUP BY CF
HAVING MAX(numpartite)
```

output:
empty (perche' 2 arbitri hanno lo stesso numero di partite ed e' il piu' alto)

3.
Il numero di partite che avra' arbitrato ogni arbitro a fine campionato

```
CREATE VIEW TOTALEPARTITE AS
SELECT * FROM PARTITADAGIOCARE
UNION
SELECT id,data,locali,ospiti,arbitro,palestra FROM PARTITAGIOCATA;

SELECT MAX(numpartite), nome, cognome FROM (SELECT data, locali, ospiti,
nome,cognome, count(CF) as numpartite
FROM TOTALEPARTITE, ARBITRO
where CF=arbitro
GROUP BY CF) AS subselect GROUP BY numpartite HAVING max(numpartite);
```

4.
Tutti gli UTENTI che hanno un rapporto di partite vinte/giocate = 1 (vinte tutte le partite)

```
SELECT * FROM UTENTE,GIOCATORE, SQUADRA,CLASSIFICA WHERE
UTENTE.CodFisc=GIOCATORE.CF AND GIOCATORE.squadra=SQUADRA.nome AND
SQUADRA.nome=CLASSIFICA.squadra AND
pVinte=pGiocate;
```

NOTE

Nel codice SQL consegnato inoltre ci sono valori gia' prefatti per la popolazione di un database completo.
Ci sono diversi utenti gia' creati.
Esempio di utente NON giocatore: USERNAME: mauro PASSWORD: conti
Esempio di utente giocatore: USERNAME: alberto PASSWORD: deagostini
Per accedere alla pagina di amministrazione bisogna loggare con credenziali:
USERNAME: admin PASSWORD: admin

Nella pagina di amministrazione si puo' eseguire qualunque query nel database, in tale pagina di potra' vedere anche il risultato di essa, tuttavia l'esecuzione di qualche query 'distrugge' l'interfaccia web poiche' il fetch del risultato provoca qualche problema, non sono riuscito a risolvere questo problema per mancanza di tempo, tuttavia tutte le query vengono eseguite correttamente indipendentemente da cosa succede nell'interfaccia web.

Nello spazio dell'utente tutto e' gia' pronto per il testing, tuttavia se si vuole partire da 0, dopo aver popolato il database bisogna entrare nella pagina di amministrazione attraverso le credenziali gia' scritte in precedenza e premere il botton INIZIA CAMPIONATO.
Tale bottone chiama una funzione che va a creare tutte le varie combinazioni di squadre con date casuali (settimana per settimana) e arbitro casuale (tra quelli del DB) mettendo ogniuna di queste combinazioni nella tabella PARTITADAGIOCARE.

CODICE PHP:

Riporto qui il codice per l'interfaccia web, quindi html e php, metto solo il codice della pagina index poiche' gli altri risultano essere simili e allungherebbero soltanto la relazione di molto, il codice e' naturalmente disponibile con la consegna del progetto.

```
<?php
                                                    require_once('sessions.php');
?>
<!DOCTYPE HTML>
<html>
<head>
                                                    <link rel="stylesheet" type="text/css"
href="css/style.css">
</head>
<body>
                                                    <div class="header">
                                                     <a class='a-header'
href="index.php"></a>
                                                    </div>

                                                    <div class="nav">
                                                     <ul>
                                                        <li><a
href="index.php">Classifica</a></li>
                                                        <li><a href="inscribeteam.php">Iscrivi
una squadra</a></li>
                                                        <li><a href="calendar.php">Calendario
partite</a></li>
                                                        <li><a href="oldmatches.php">Scorse
partite</a></li>
                                                        <li><a href="addresult.php">Inserisci
un risultato</a></li>
                                                     </ul>
                                                    </div>
                                                    <div class="content">
                                                     <table class="classifica-table">
                                                        <tr>
                                                            <th>Squadra</th>
                                                            <th>Punteggio</th>
                                                            <th>Partite</th>
                                                            <th>Vinte</th>
                                                            <th>Perse</th>
                                                            <th>Set vinti</th>
                                                            <th>Set persi</th>
                                                        </tr>
```

```
<?php
    $_classificaResult = $con->query("SELECT * FROM CLASSIFICA ORDER BY PUNTEGGIO DESC");
    while($row = $_classificaResult->fetch_assoc()){
        $squa=$row["squadra"];
        $punt=$row["punteggio"];
        $part=$row["pGiocate"];
        $vint=$row["pVinte"];
        $pers=$row["pPerse"];
        $setv=$row["setVinti"];
        $setp=$row["setPersi"];
        echo '  <tr>
                <td>'.$row["squadra"].'</td>
                <td>'.$row["punteggio"].'</td>
                <td>'.$row["pGiocate"].'</td>
                <td>'.$row["pVinte"].'</td>
                <td>'.$row["pPerse"].'</td>
                <td>'.$row["setVinti"].'</td>
                <td>'.$row["setPersi"].'</td>
                </tr>';
    }
?>
</table>
</div>
<div class="right-column">
<div class="logger">
    <?php
        if(!isset($_SESSION['username'])) {
//mostrare form login
            $logForm = '<form method="POST" action='.$_SERVER['PHP_SELF'].'>
                <h3 style="text-align:center">Login Form</h3>
                <input type="text" name="username" placeholder="nome utente"></input>
                <input type="password" name="password" placeholder="password"></input>
                <input type="submit" name="login"></input><br>
                Non sei iscritto? <a href="registration.php">registrati</a>
                </form>';
        }
        else{ // mostrare form logout
            $logForm = '<form method="POST" action='.$_SERVER['PHP_SELF'].'>
                <h3 style="text-
```

```
align:center">Benvenuto '.$_SESSION['username'].'</h3>
                                                    <a href="profile.php">Il
tuo profilo</a>
                                                    <input type="hidden"
name="logout"></input><br>
                                                    <input type="submit"
name="logout">Logout</input><br>
                                                    </form>';
                                                }
                                                echo $logForm;
                                            ?>
                                        </div>
                                        <div class="next-matches">
                                            <?php
                                                nextMatches($con);
                                            ?>
                                        </div>
                                    </div>
</body>
</html>
```