Progetto basi di dati

Gestionale Palestra arti marziali

Silvio Cici
Matr.:1049165

Davide Trevisan
Matr.:1070686

Abstract

Il progetto che si propone consiste nella realizzazione di una base di dati per la gestione di una palestra di arti marziali. Viene creata una pagina web per permettere la consultazione e l'interazione col database sia ai maestri che agli allievi. Si da la possibilità agli allievi di creare un account con il quale è possibile iscriversi ad un corso e di consultare dati utili per loro, quali ad esempio gli orari delle prossime lezioni, la lista dei maestri e le gare disponibili.
Ai maestri è designata la parte più corposa dell'interfaccia web, in quanto essi possono svolgere le azioni più interessanti, come ad esempio aggiungere altri maestri, preiscrivere allievi alle gare, aggiungere corsi, aggiungere lezioni a corsi, promuovere un amatore ad agonista.

Analisi dei requisiti

Il progetto vuole modellare una base di dati che gestisca una piccola palestra di arti marziali tramite un sito web. Gli **utenti** del sito sono caratterizzati da un username e da una password, entrambi utilizzati per il login, da un codice fiscale, che li identifica univocamente. Sono dotati inoltre dei dati anagrafici principali quali nome, cognome e data di nascita. Gli utenti della palestra si dividono in maestri e allievi.
I **maestri** oltre ai dati già presentati sono dotati di un numero di telefono per poterli contattare in caso di necessità. I maestri possono insegnare una o più **discipline**, identificate da un nome, tra quelle della palestra presa (nel caso preso in oggetto del progetto la scelta è tra (BJJ, Karate e Judo)).
Per ogni **allievo** oltre ai dati anagrafici già presentati è necessario anche conoscere se è in regola coi pagamenti o meno.
Ogni allievo, per ogni disciplina, può appartenere alla lista degli **agonisti**, identificati mediante il codice fiscale e la disciplina in cui si è agonista, mentre è annoverato tra i **non agonisti** nelle discipline in cui non è agonista**.** I non agonisti sono identificati alla medesima maniera degli agonisti. I non agonisti possono essere amatori se hanno partecipato ad almeno un corso della disciplina in questione, anche non presso la palestra presa in oggetto dal progetto.
Ad un maestro può essere affidata la responsabilità dell'organizzazione di **corsi**, che sono identificati univocamente tramite un id. I corsi sono destinati ad una precisa fascia d'età, possono essere riservati solamente agli agonisti o aperti a qualsiasi allievo e possono avere un prezzo.
I corsi sono suddivisi in **lezioni**, e sono identificate mediante la loro data di inizio e fine, oltre che mediante l'id del corso a cui appartengono e il luogo in cui si svolge. L'iscrizione ad un corso per amatori comporta il passaggio allo status di amatore per coloro che erano non agonisti e non amatori, mentre ad un corso per agonisti si possono iscrivere solo agonisti.
Le **palestre** presso cui hanno luogo le varie lezioni sono identificate univocamente mediante la loro ubicazione, espressa mediante l'indirizzo, e possono essere palestre prese in affitto o di proprietà della palestra, per cui bisogna tener traccia anche di quanto dovuto eventualmente al proprietario.
Il maestro è inoltre responsabile della preiscrizione di un agonista alle **gare**, quando e se ritiene l'allievo idoneo ad affrontarla.
Una gara ha un nome ed è identificata dall'ubicazione, dal giorno in cui si svolge e dalla disciplina in oggetto; inoltre partecipare ad una gara può avere dei costi, se non è vicina, e quindi si vuole tener conto del costo stimato per la trasferta.

Progettazione Concettuale

Lista delle classi:

Gli attributi delle classi sono principalmente NOT NULL, perchè sono indispensabili per il database.
Quelli che possono essere NULL invece sono:
'NumTel' della classe Maestri; 'Insegnante' della classe Corsi per permettere un eventuale passaggio della gestione di un corso da un maestro a un altro in caso si dovesse cancellare l'attuale maestro e quindi il corso fosse momentaneamente senza un gestore, senza dover eliminare tutto il corso con le sue lezioni per poi reinserirlo; 'Prezzo' sempre della classe Corsi; 'Costo_trasferta' della classe gare; 'affitto' della classe Palestre.
Le classi presenti nel progetto sono:

- ***utenti:*** classe fondamentale per la gestione del login iniziale nel sito

Attibuti:
- username character (20) unique per permettere un login univoco
- password character (20)
- codicefiscale character (26) primary key
- Nome character(20)
- Cognome character(20)
- DataNascita date

utenti si divide in:

1. ***Maestri:*** modella i meastri dell'associazione, in qualità di utenti amministratori
   Attibuti:
   - NumTel character(15)

2. ***Allievo:*** modella gli allievi della palestra, in qualità di utenti del lato client
   Attibuti:
   - Pagamenti_in_regola BOOLEAN

- ***Discipline:*** contiene le discipline (nel caso preso in oggetto: BJJ, Judo, Karate)

Attibuti:
- Nome character(20) primary key

- ***Agonisti:*** contiene la lista delle discipline in cui un allievo è agonista
  Attibuti:
  non ha attributi propri (eredita il nome della disciplina e il codice fiscale dell'allievo)

- ***Non_Agonisti:*** contiene la lista delle discipline in cui un allievo non è agonista
  Attibuti:
  - Amatore BOOLEAN

- ***Corsi:*** modella i corsi insegnati nella palestra
  Attibuti:
  - Prezzo INTEGER
  - id_corso integer AUTO_INCREMENT primary key
  - fascia_eta character(20)
  - agonistico BOOLEAN

- ***Gare***: elenca le gare per ogni singola disciplina

Attibuti:
- Ubicazione character(100)
- Giorno datetime
- Costo_trasferta int
- NomeGara character(40)

- ***Palestre***: contiene le palestre a disposizione dell'associazione

Attibuti:
- Ubicazione character(100) primary key
- affitto integer

- ***Lezione***: modella le lezioni di ogni singolo corso

Attibuti:
- inizio datetime
- fine datetime

Lista delle associaizoni:

- ***divisioneA:*** Agonisti-Allievo

Molteplicità: **1:N** l'agonista è al più un allievo, un allievo può essere agonista in più discipline.
Totalità: **1:0** l'agonista è sempre un allievo, l'allievo può non essere agonista.

- ***divisioneNA:*** Non_agonisti-Allievo

Molteplicità: **1:N** il non-agonista è al più un allievo, un allievo può essere non-agonista in più discipline.
Totalità: **1:0** il non-agonista è sempre un allievo, l'allievo può non essere agonista.

- ***iscrizione:*** Allievo-Corsi

Molteplicità: **N:M** un allievo può essere iscritto a più corsi, un corso può avere più allievi iscirtti.
Totalità: **0:0** un allievo può non essere ancora iscritto a nessuno corso, un corso può non avere iscritti.

- ***preiscrizione:*** Agonisti-Gare

Molteplicità: **N:M** un agonista può essere preiscritto a più gare, una gara può avere più agonisti preiscritti.
Totalità: **0:0** un agonista può non essere preiscritto a nessuna gara, una gara può non avere preiscritti.

- ***gestione:*** Maestri-Corsi

Molteplicità: **N:1** un maestro più insegnare in più corsi, un corso pió avere al massimo un maestro.
Totalità: **0:0** un maestro può non star insegnando in nessun corso, un corso può non avere momentaneamente un maestro.

- ***praticaA:*** Agonisti-Discipline

Molteplicità: **1:N** un atleta può essere definito agonista a seconda di ogni singola disciplina

che pratica, una disciplina può avere più agonisti.
Totalità: **1:0** un atleta è agonista se lo è almeno in almeno una disciplina, una disciplina può non avere agonisti.

- ***praticaNA:*** Non_agonisti-Discipline

Molteplicità:**1:N** un atleta può essere non agonista a seconda delle discipline che pratica, una disciplina può avere più non-agonisti.
Totalità: **1:0** un atleta è non agonista se non è agonista in almeno una disciplina, una disciplina può non avere non-agonisti.

- ***insegnamento:*** Maestri-Discipline

Molteplicità: **N:M** un maestro può insegnare più discipline, una disciplina può avere più maestri.
Totalità: **1:1** un maestro deve insegnare almeno una disciplina altrimenti non è un maestro, una disciplina ha almeno un insegnante per default, altrimenti non potrebbe essere insegnata nella palestra.

- ***calendario:*** Gare-Discipline

Molteplicità: **1:N** ogni gara è definita per una disciplina, una disciplina può avere più gare.
Totalità: **1:0** ogni gara è definita per una disciplina, una disciplina può non avere gare in programma.

- ***relativi:*** Corsi-Discipline

Molteplicità: **1:N** ogni corso è definito per una disciplina, una disciplina può avere più corsi.
Totalità: **1:0** ogni corso è definito per una disciplina, una disciplina può non avere corsi.

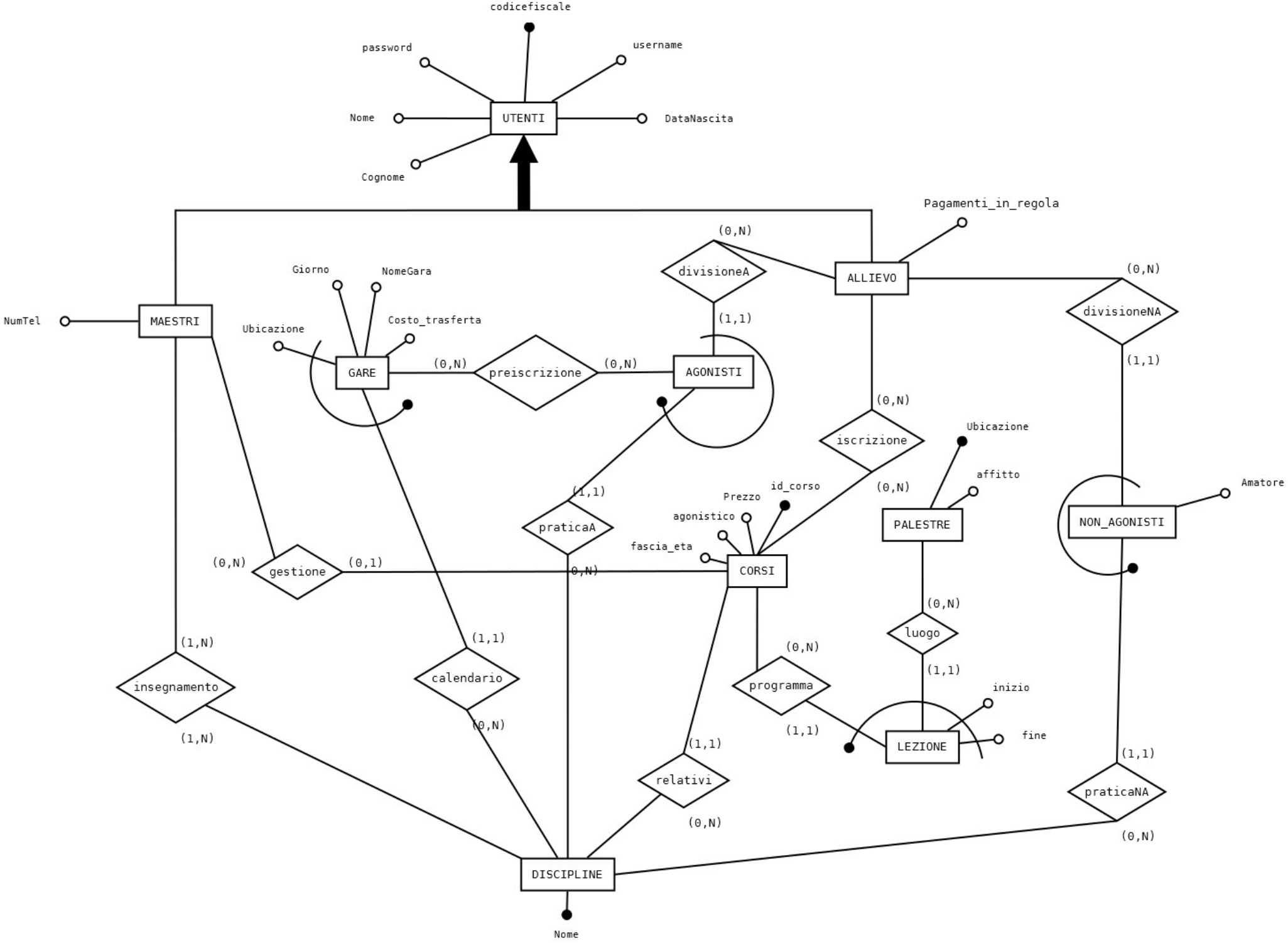
- ***programma:*** Corsi-Lezione

Molteplicità: **N:1** un corso può avere più lezioni, una lezione appartiene sempre a un corso.
Totalità: **0:1** un corso può non avere ancora lezioni, una lezione appartiene sempre a un corso.

- ***luogo:*** Lezione-Palestre

Molteplicità: **1:N** una lezione può avvenire solo in una palestra alla volta, in una palestra possono avvenire più lezioni.
Totalità: **1:0** una lezione può avvenire solo in una palestra alla volta, una palestra può non avere lezioni in programma.

A seguire lo schema di quanto detto:


codicefiscale
password
username
Nome
UTENTI
DataNascita
Cognome
Pagamenti_in_regola
(0,N)
divisioneA
ALLIEVO
(0,N)
divisioneNA
Giorno
NomeGara
NumTel
MAESTRI
(1,1)
(1,1)
Ubicazione
Costo_trasferta
GARE
(0,N)
preiscrizione
(0,N)
AGONISTI
(0,N)
iscrizione
Ubicazione
affitto
(1,1)
Prezzo
id_corso
Amatore
praticaA
(0,N)
PALESTRE
NON_AGONISTI
agonistico
fascia_eta
(0,N)
gestione
(0,1)
0,N)
CORSI
(0,N)
luogo
(1,N)
(1,1)
(1,1)
insegnamento
calendario
(0,N)
programma
(1,1)
inizio
fine
(1,N)
(0,N)
(1,1)
relativi
LEZIONE
(1,1)
praticaNA
(0,N)
(0,N)
DISCIPLINE
Nome

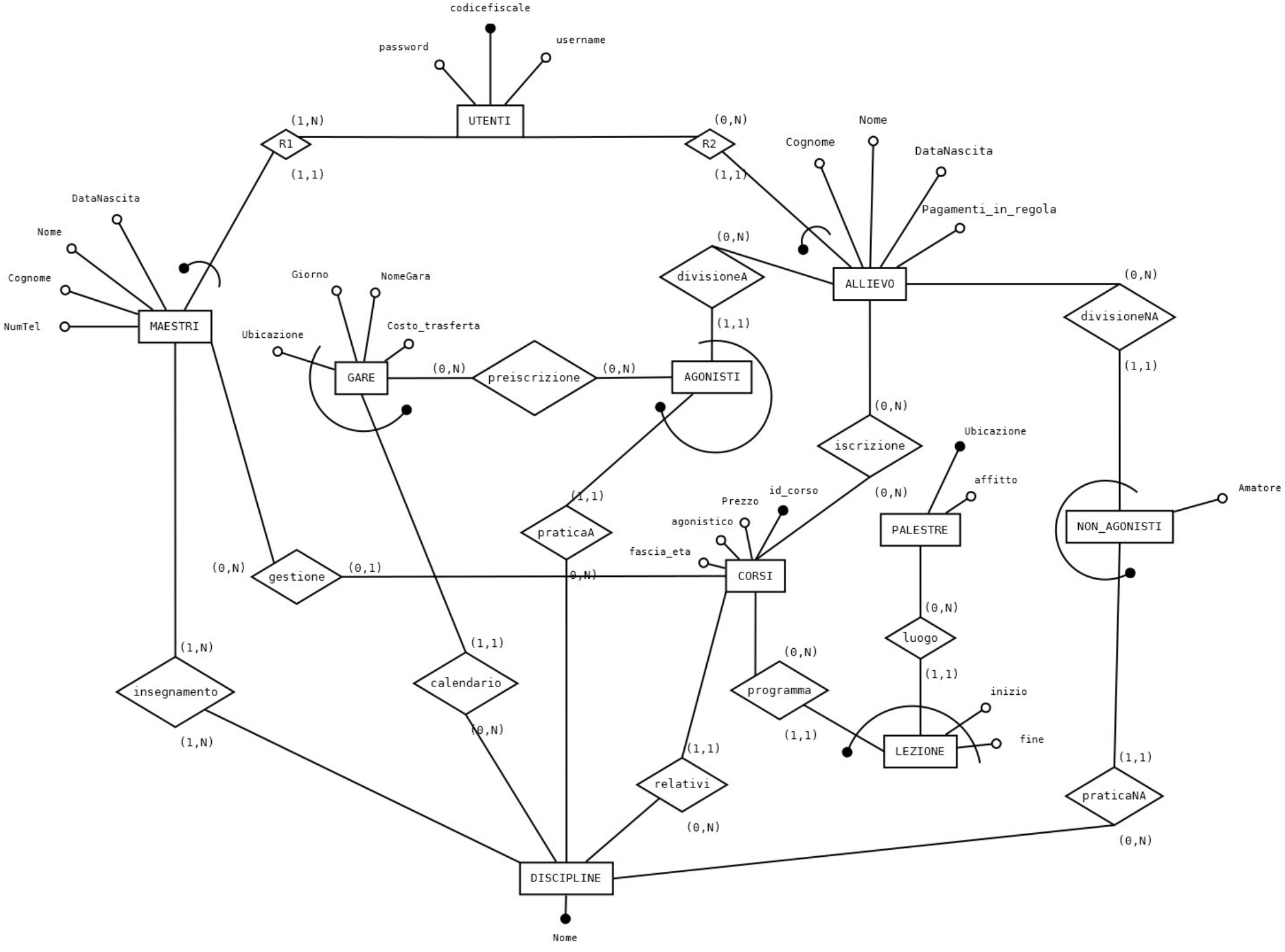
Vincoli non esprimibili:

- Un allievo può iscriversi ad un corso solo se appartiene alla fascia d'età a cui esso è destinato
- L'iscrizione ad un corso per agonisti è subordinata alla presenza del medesimo allievo tra gli agonisti della disciplina in oggetto.
- Un allievo o è agonista o è non agonista in una stessa disciplina;
- L'inizio della lezione deve essere antecedente alla sua fine;
- Un non agonista non amatore in una disciplina diventa amatore al momento della sua iscrizione ad un corso della stessa disciplina.
- Un allievo al momento della registrazione al sito ha sempre i pagamenti in regola.

Progettazione logica

Gerarchia

La gerarchia presente nel modello concettuale avente Utente come superclasse e Allievo e Maestro come sottoclassi è stata resa nel modello relazionale tramite una soluzione ibrida tra il collasso verso il basso e la sostituzione con relazione della gerarchia. Le ragioni della scelta sono dovute al fatto che accorpare i figli nel padre sarebbe risultato praticamente impossibile, in quanto maestri e allievi hanno attributi e relazioni unici e quindi non era corretto adottare questa soluzione, poiché moltissimi accessi sono contestuali ai figli. Accorpare interamente il padre nei figli era una soluzione possibile, ma si è preferito mantenere l'entità utente con in essa i campi username, password e codice fiscale, mentre si son spostati ad allievo e maestro i campi nome, cognome e data di nascita, usando come chiave il codice fiscale, con vincolo di integrità referenziale verso quello contenuto in utente. Le ragioni di questa scelta sono dovute al fatto che sarebbe stato inutile accollarsi il peso in maestri e allievi dei campi username e password, poichè inutilizzati a parte un unico accesso durante la verifica delle credenziali, a differenza degli altri campi la cui frequenza di accesso è di molto superiore. Si è quindi scelto di tenere la tabella contenente tutti gli utenti al fine di alleggerire sia le tabelle sia le interrogazioni, ottenendo pure un guadagno sulle prestazioni. Inoltre accorpando nei figli gli attributi sopra elencati si è voluto dare dignità alle entità e allievo e maestro e al significato semantico che esse assumono durante tutto il progetto rispetto ad utente, che è più astratta: se fossero restati nel padre, per poter avere il numero di telefono e i dati anagrafici di un maestro si sarebbe dovuto ricorrere un join! nella soluzione adottata è invece possibile ottenere quel dato con una semplice selezione e ciò è migliore sotto tutti i punti di vista. Lo schema diventerebbe quindi come questo (permangono i vicoli non esprimibili già citati):


codicefiscale
password
username
UTENTI
(1,N)
R1
(1,1)
(0,N)
R2
(1,1)
Nome
Cognome
DataNascita
Pagamenti_in_regola
ALLIEVO
(0,N)
divisioneA
(1,1)
(0,N)
divisioneNA
(1,1)
DataNascita
Nome
Cognome
NumTel
MAESTRI
Giorno
NomeGara
Ubicazione
Costo_trasferta
GARE
(0,N)
preiscrizione
(0,N)
AGONISTI
(0,N)
iscrizione
(0,N)
Ubicazione
affitto
PALESTRE
Amatore
NON_AGONISTI
(1,1)
praticaA
Prezzo
id_corso
agonistico
fascia_eta
CORSI
(0,N)
gestione
(0,1)
0,N)
(0,N)
luogo
(1,1)
(1,N)
insegnamento
(1,N)
(1,1)
calendario
(0,N)
(0,N)
programma
(1,1)
inizio
fine
LEZIONE
(1,1)
relativi
(0,N)
(1,1)
praticaNA
(0,N)
DISCIPLINE
Nome

Schema relazionale:

UTENTI (username, password, codicefiscale)

MAESTRI (Nome, Cognome, DataNascita, NumTel, codfiscale)
Vincoli di integrità referenziale:
- codfiscale si riferisce a codicefiscale della tabella UTENTI.

DISCIPLINE (Nome)

INSEGNAMENTO (Insegnante, Disciplina)
Vincoli di integrità referenziale:
- Insegnante si riferisce a codfiscale della tabella MAESTRI.
- Disciplina si riferisce a Nome della tabella DISCIPLINE.

ALLIEVO (CodiceFiscale, Nome, Cognome, DataNascita, Pagamenti_in_regola)
Vincoli di integrità referenziale:
- codicefiscale si riferisce a codicefiscale della tabella UTENTI.

AGONISTI (Disciplina, Allievo)
Vincoli di integrità referenziale:
- Disciplina si riferisce a Nome della tabella DISCIPLINE.
- Allievo si riferisce a CodiceFiscale della tabella ALLIEVO.

NON_AGONISTI (Disciplina, Allievo, Amatore)
Vincoli di integrità referenziale:
- Disciplina si riferisce a Nome della tabella DISCIPLINE.
- Allievo si riferisce a CodiceFiscale della tabella ALLIEVO.

CORSI (Insegnante, Disciplina, Prezzo, id_corso, fascia_eta, agonistico)
Vincoli di integrità referenziale:
- Insegnante si riferisce a codfiscale della tabella MAESTRI.
- Disciplina si riferisce a Nome della tabella DISCIPLINE.

ISCRIZIONE (Allievo, Corso)
Vincoli di integrità referenziale:
- Allievo si riferisce a CodiceFiscale della tabella ALLIEVO.
- Corso si riferisce a id_corso della tabella CORSI.

GARE (Ubicazione, Giorno, Disciplina, Costo_trasferta, NomeGara)
Vincoli di integrità referenziale:
- Disciplina si riferisce a Nome della tabella DISCIPLINE.

PREISCRIZIONI (Allievo, UbicazioneGara, GiornoGara, Disciplina)
Vincoli di integrità referenziale:
- Allievo si riferisce ad Allievo della tabella AGONISTI.
- UbicazioneGara, GiornoGara, Disciplina si riferiscono a Ubicazione, Giorno, Disciplina della tabella GARE.

PALESTRE (Ubicazione, affitto)

LEZIONE (inizio, fine, palestra, corso)

Vincoli di integrità referenziale:

- palestra si riferisce a Ubicazione della tabella PALESTRE.
- corso si riferisce a id_corso della tabella CORSI.

Implementazione

A seguito è riportata l'implementazione in Mysql di quanto detto:

```
CREATE DATABASE IF NOT EXISTS Palestra;


CREATE TABLE IF NOT EXISTS Palestra.utenti(
username character(20) unique NOT NULL,
password character(20) NOT NULL,
codicefiscale character(20) primary key) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Maestri(
Nome character(20) NOT NULL,
Cognome character(20) NOT NULL,
DataNascita date NOT NULL,
NumTel character(15),
codfiscale character (16) primary key,
FOREIGN KEY (codfiscale) REFERENCES palestra.utenti(codicefiscale) ON DELETE
CASCADE ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Discipline(
Nome character(20) PRIMARY KEY) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS Palestra.Insegnamento(
Insegnante character(16) NOT NULL,
Disciplina character(20) NOT NULL,
PRIMARY KEY (Insegnante,Disciplina),
FOREIGN KEY (Insegnante)
REFERENCES Palestra.Maestri(codfiscale) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (Disciplina)
REFERENCES Palestra.Discipline(Nome) ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Allievo(
CodiceFiscale character(16) PRIMARY KEY,
Nome character(20) NOT NULL,
Cognome character(20) NOT NULL,
DataNascita date NOT NULL,
Pagamenti_in_regola BOOLEAN NOT NULL,
FOREIGN KEY (codicefiscale) REFERENCES palestra.utenti(codicefiscale) ON DELETE
CASCADE ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Agonisti(
Disciplina character(20) not null,
Allievo character(16) not null,
primary key(Disciplina,Allievo),
```

```
foreign key (Disciplina) references Palestra.Discipline(Nome) ON DELETE CASCADE ON
UPDATE CASCADE,
foreign key (Allievo) references Palestra.Allievo(CodiceFiscale) ON DELETE CASCADE ON
UPDATE CASCADE)ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Non_Agonisti(
Disciplina character(20) not null,
Allievo character(16) not null,
Amatore BOOLEAN NOT NULL,
primary key(Disciplina,Allievo),
foreign key (Disciplina) references Palestra.Discipline(Nome) ON DELETE CASCADE ON
UPDATE CASCADE,
foreign key (Allievo) references Palestra.Allievo(CodiceFiscale) ON DELETE CASCADE ON
UPDATE CASCADE)ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Corsi(
Insegnante character(16),
Disciplina character(20) NOT NULL,
Prezzo INTEGER,
id_corso integer AUTO_INCREMENT primary key,
fascia_eta character(20) NOT NULL,
agonistico BOOLEAN NOT NULL,
FOREIGN KEY (Insegnante) references Palestra.Maestri(codfiscale) ON UPDATE CASCADE ON
DELETE SET NULL,
FOREIGN KEY (Disciplina) references Palestra.Discipline(Nome) ON UPDATE CASCADE ON
DELETE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Iscrizione(
Allievo character (16) NOT NULL,
Corso integer NOT NULL,
PRIMARY KEY (Allievo,Corso),
FOREIGN KEY (Allievo) references Palestra.Allievo(CodiceFiscale) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Corso) references Palestra.Corsi(id_corso) ON DELETE CASCADE ON
UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Gare(
Ubicazione character(100) NOT NULL,
Giorno datetime NOT NULL,
Disciplina character(20) NOT NULL,
Costo_trasferta int,
NomeGara character(40) NOT NULL,
PRIMARY KEY (Ubicazione,Giorno,Disciplina),
FOREIGN KEY (Disciplina) references Palestra.Discipline(Nome) ON DELETE CASCADE ON
UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Preiscrizioni(
```

```
Allievo character(16) not null,
UbicazioneGara character(100) not null,
GiornoGara datetime not null,
Disciplina character(20) not null,
primary key (Allievo,UbicazioneGara,GiornoGara,Disciplina),
foreign key (Allievo) references Palestra.agonisti(Allievo) ON DELETE CASCADE ON UPDATE
CASCADE,
foreign key(UbicazioneGara,GiornoGara,Disciplina) references
Palestra.Gare(Ubicazione,Giorno,Disciplina) ON DELETE CASCADE ON UPDATE
CASCADE)ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Palestre(
Ubicazione character(100) primary key,
affitto integer) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Palestra.Lezione(
inizio datetime not null,
fine datetime not null,
palestra character(100) not null,
corso integer not null,
primary key (inizio,fine,palestra,corso),
foreign key (palestra) references Palestra.Palestre(Ubicazione) ON DELETE CASCADE ON
UPDATE CASCADE,
foreign key (corso) references Palestra.corsi(id_corso) ON DELETE CASCADE ON UPDATE
CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Funzioni

Nella base di dati sono state implementate le seguenti funzioni di cui si fornisce il codice, contenute nel file functiontrigger.sql:

- contalezioni, funzione che dato l'id di un corso restituisce come intero il numero di lezioni dello stesso:

```
DELIMITER $$
DROP FUNCTION IF EXISTS contalezioni;
$$
CREATE FUNCTION contalezioni (idcorso int(11))
RETURNs integer
BEGIN
DECLARE numlezioni integer;
SELECT COUNT(*) INTO numlezioni FROM palestra.lezione WHERE
corso=idcorso;
RETURN numlezioni;
END $$
$$
```

- guadagnocorso, che dato l'id di un corso restituisce come intero il guadagno ottenuto dallo stesso, senza contare gli utenti che si sono cancellati

```
DROP FUNCTION IF EXISTS guadagnocorso;
$$
CREATE FUNCTION guadagnocorso(idcorso int(11))
RETURNS integer
BEGIN
DECLARE prezzocorso integer;
DECLARE numiscritti integer;
SELECT COUNT(*) INTO numiscritti FROM palestra.iscrizione WHERE
corso=idcorso;
SELECT prezzo INTO prezzocorso FROM palestra.corsi WHERE id_corso=idcorso;
RETURN prezzocorso*numiscritti;
END $$
DROP FUNCTION IF EXISTS fasciaanni;
 $$
```

- fasciaanni, che dato un codice fiscale, fornisce la fascia d'età dell'allievo sotto forma di stringa; le fasce d'età sono -11,12-15,16-35,+35

```
CREATE FUNCTION fasciaanni (cf CHARACTER(20))
RETURNs CHARACTER(20)
BEGIN
DECLARE anni integer;
DECLARE nascita date;
select DataNascita into nascita from palestra.allievo where codicefiscale=cf;
select  TIMESTAMPDIFF(YEAR,(nascita),CURDATE( )) into anni;
   CASE
     WHEN (anni>=0 and anni<=11) THEN RETURN('-11');
```

```
        WHEN (anni>11 and anni<=15) THEN return('12-15');
        WHEN (anni>15 and anni<=35) THEN return('16-35');
        WHEN (anni>35 and anni<150) THEN return('+35');
      END CASE;
    END
    $$
```

si noti come sia stato messo un tetto massimo e minimo ragionevole per l'età, in modo da non accettare date palesemente false.

Queries

di seguito verranno mostrate alcune delle query più interessanti con il relativo codice e output; lo stato del database usato per questi esempi non è quello che viene fornito in allegato, anche se è molto simile. Le queries sono state implementate nel progetto nel codice PHP, quindi si è scelto di sostituire le variabili PHP con dei valori opportuni.

- seleziona tutti gli allievi della palestra, con i rispettivi campi, ordinati per cognome e nome:

```
SELECT *
FROM palestra.allievo
ORDER BY cognome,nome ASC
```

risultato:

CodiceFiscale	Nome	Cognome	DataNascita	Pagamenti_in_regola
AGOKAR90G11C111Q	agonistakarate	bianchi	1990-03-20	1
AMAJDO90G11C111Q	amatorejudo	blu	1982-02-13	1
AGOJDO90G11C111Q	agonistajudo	bruni	1990-01-11	1
ALLESE93L12C111Q	allievoesempio	deluca	1994-11-11	1
AMAKAR90G11C111Q	amatorekarate	freschi	1998-02-11	1
AGOBJJ90G11C111Q	agonistabjj	rossi	1990-06-11	1
AMABJJ90G11C111Q	amatorebjj	sartorello	1992-03-12	1

- seleziona il codice fiscale dell'utente avente username=”Sodets” e password “password”

```
SELECT codicefiscale
FROM palestra.utenti
WHERE username='Sodets'  AND password='password';
```

risultato:

codicefiscale
CCISLV93L18C111Q

- Seleziona il nome del maestro, il cognome del maestro, la disciplina, il prezzo, l'id dei corsi per amatori disponibili per ALLESE93L12C111Q (allievoesempio):

```
SELECT nome,cognome,disciplina,prezzo,id_corso
FROM corsi LEFT JOIN maestri ON insegnante= codfiscale
WHERE agonistico='0' AND fascia_eta=fasciaanni('ALLESE93L12C111Q')
AND id_corso NOT IN (SELECT corso FROM palestra.iscrizione WHERE
```

allievo='ALLESE93L12C111Q');

risultato:

nome	cognome	disciplina	prezzo	id_corso
Renzo	Ondei	Judo	30	3

come si può vedere si è fatto uso della funzione fasciaanni precedentemente mostrata.

- Seleziona la disciplina, la fascia d'età.l'id del corso, l'inizio, la palestra delle lezioni di AGOKAR90G11C111Q (agonistakarate) che si devono ancora svolgere, mostrando il prezzo del corso e se il corso è per agonisti o meno, il tutto ordinato per data di inizio, disciplina, id del corso:

```
SELECT c.Disciplina,c.fascia_eta,c.id_corso,l.inizio,l.palestra,c.Prezzo,c.agonistico
FROM (((palestra.corsi AS c JOIN palestra.lezione as l on c.id_corso=l.corso)
JOIN palestra.iscrizione AS i ON c.id_corso=i.Corso)
JOIN palestra.allievo AS a ON i.Allievo=a.CodiceFiscale)
WHERE l.inizio>=curdate()
AND i.Allievo='AGOKAR90G11C111Q'
ORDER BY l.inizio,c.Disciplina,c.id_corso ASC
```

risultato:

Disciplina	fascia_eta	id_corso	inizio	palestra	Prezzo	agonistico
Judo	16-35	3	2015-11-27 14:15:00	Via Puccini, Castelfranco Veneto	30	0
Karate	16-35	7	2015-11-28 14:15:00	Via Boito, Castelfranco Veneto	40	1

- Seleziona le gare di AGOBJJ90G11C111Q (agonistabjj) che si devono ancora svolgere, mostrandone il giorno, la disciplina, il nome, l'ubicazione, il costo stimato della trasferta e ordinando il risultato per giorno e disciplina

```
SELECT b.GiornoGara,b.Disciplina,a.NomeGara,b.UbicazioneGara,a.Costo_trasferta
FROM palestra.gare as a
RIGHT JOIN palestra.preiscrizioni AS b
ON a.Ubicazione=b.UbicazioneGara
WHERE a.Giorno=b.GiornoGara
AND a.Disciplina=b.Disciplina AND a.Giorno>=curdate()
AND Allievo='AGOBJJ90G11C111Q'
ORDER BY a.Giorno,a.Disciplina ASC
```

Risultato:

GiornoGara	Disciplina	NomeGara	UbicazioneGara	Costo_trasferta
2015-12-12 09:00:00	BJJ	Milano Challenge	Milano	50

- Seleziona tutti gli allievi di BRSPRD93G11C111Q (Paride Bressan), riportando per ognuno di essi, il nome, il cognome, la data di nascita, e se è in regola coi pagamenti o meno:

```
SELECT DISTINCT nome,cognome,datanascita,pagamenti_in_regola
FROM palestra.iscrizione
LEFT JOIN palestra.corsi ON corso=id_corso
LEFT JOIN palestra.allievo ON codicefiscale=allievo
WHERE insegnante='BRSPRD93G11C111Q'
ORDER BY cognome,nome ASC
```

risultato:

nome	cognome	datanascita	pagamenti_in_regola
agonistakarate	bianchi	1990-03-20	1
amatorekarate	freschi	1998-02-11	1

Trigger

nel database sono stati inoltre implementati i seguenti trigger, contenuti nel file funzionitrigger.sql, di cui si fornisce il codice:

- controllaorario, che controlla se l'orario di inizio della lezione è antecedente a quello di fine. In caso contrario la query di INSERT/UPDATE della lezione fallirà perchè il trigger genererà un errore impostando il valore di inizio a NULL, e inizio è NOT NULL come specificato in fase di creazione della tabella lezione. Ecco il codice:

```
DROP TRIGGER IF EXISTS controllaorario;
 $$
CREATE TRIGGER controllaorario BEFORE INSERT ON palestra.lezione
FOR EACH ROW
BEGIN
IF NEW.inizio>NEW.fine
THEN SET NEW.inizio=NULL;
END IF;
END
$$
```

- checkfascia, che all'inserimento di un record nella tabella iscrizione controlla che l'allievo che tenta di iscriversi a un dato corso appartenga effettivamente alla fascia d'età a cui quel corso è destinato. In caso contrario si genera un errore nella stessa modalità esposta nel trigger precedente:

```
DROP Trigger IF EXISTS checkfascia;
 $$
CREATE TRIGGER checkfascia BEFORE INSERT ON palestra.iscrizione
FOR EACH ROW BEGIN
declare cfallievo CHARACTER(20);
DECLARE fasciallievo CHARACTER(20);
DECLARE fasciacorso CHARACTER(20);
select codicefiscale into cfallievo from palestra.allievo WHERE
codicefiscale=new.allievo;
select fasciaanni(cfallievo) into fasciallievo;
SELECT fascia_eta into fasciacorso from palestra.corsi where id_corso=new.corso;
if fasciallievo!=fasciacorso
THEN
set new.allievo=NULL;
END IF;
END
$$
```

si noti come si è usata anche qui la funzione fasciaanni precedentemente citata;

Interfaccia web

Si noti che l'interfaccia web svolge le funzioni basilari di inserimento. Per altre funzioni più avanzate è richiesto l'accesso al database. Nell'interfaccia web vengono svolti anche molti controlli, perché così è possibile dare un “margine di manovra” ai maestri, rendendo possibili forzature senza compromettere l'integrità della base di dati stessa quali ad esempio l'addossarsi corsi di materie che non si insegnano in maniera “ufficiale”, cosa impedita dall'interfaccia web.

Dato lo scopo didattico del progetto sì è lasciato l'avviso in caso di errore nella query sql.

l'interfaccia web del sito è composta per la maggior parte da pagine con contenuto misto tra HTML e PHP e pagine di PHP puro a cui fanno riferimento i vari action presenti nei form HTML. In due casi, cioè in form.html (che necessita di register.php) e progetto.html(che necessita di login.php), si ha HTML puro, in quanto non vi è necessaria alcuna sicurezza dato che le pagine possono e devono essere accessibili dall'esterno. Inoltre è incluso un file immagine usato per lo sfondo e un file main.css contenente il CSS del sito.
Sono presenti anche due file PHP (tabelle.php e connect.php) contenenti le funzioni necessarie rispettivamente a gestire le funzioni necessarie a creare le tabelle e quelle relative ai parametri di connessione al database.
Le altre pagine presenti oltre le due in HTML già citate sono: per la parte degli allievi Home.php,iscriviti_corso.php (che necessita di actioncorso.php) ,tutte_le_gare.php,i_maestri.php. Per la parte relativa ai maestri/amministratori vi è Homemaestro.php, inserisci_maestro.php (che necessita di insert_maestro.php), preiscrizione_gara.php (che necessita di preiscrivi_gara.php), aggiungi_gara.php (che necessita di insert_gara.php), corsi.php (che necessita di contalezioni.php), agiungi_corso.php (che necessita di insert_corso.php), aggiungi_lezioni.php (che necessita di insert_lezioni.php), agonista.php (che necessita di actionagonista.php) e allievi_palestra.php.

Vista la mole di pagine non si riporterà nella relazione il codice di ognuna esse, rimandando alla visione dei sorgenti allegati nel caso si volesse visualizzare il codice completo, tuttavia si riporteranno in seguito alcuni esempi e soluzioni usate ritenuti significativi:

per la stampa a schermo delle tabelle si è usato il metodo presentato in classe e si sono usate quindi le funzioni contenute nel file tabelle.php:

```
<?php
/* Inizia una tabella html. In input l'array degli
 header delle colonne */
function table_start($row) {
echo "<table class='tabelle'>";
echo "<tr>";
foreach ($row as $field)
echo "<th>$field</th>";
echo "</tr>";
};
/* Stampa un array, come riga di tabella html */
function table_row($row) {
echo "<tr>";
foreach ($row as $field)
if ($field) /* gestione valori nulli! */
        echo "<td>$field</td>";
 else
```

```
        echo "<td>-</td>"; echo "</tr>";};
/* funzione per terminare una tabella html */
function table_end() {
echo "</table>";};
?>
```

per quanto riguarda la sicurezza, si è usata la variabile globale $_Session come “cookie” per l'autorizzazione nelle pagine, e i suoi campi vengono inizializzati nella fase di login, e impedire quindi di poter accedere direttamente tramite URL alle pagine riservate. A questo proposito si riporta il testo di login.php:

```
<?php
session_start();
require 'connect.php';
connect();
$myusername = mysql_real_escape_string($_POST['username']);
$mypassword = mysql_real_escape_string($_POST['password']);
$sql = "SELECT * FROM palestra.utenti WHERE username='$myusername' and
password='$mypassword'";
$result = mysql_query($sql);
$count = mysql_num_rows($result);
$_SESSION['username'] = $myusername;
if ($count == 1) {
   echo"Benvenuto $myusername, hai effettuato il login con successo";

   $sql = "SELECT codicefiscale FROM palestra.utenti WHERE username='$myusername' and
password='$mypassword'";
   $result = mysql_query($sql);
   $row = mysql_fetch_array($result);
   $codice_fiscale = $row[0];
   $_SESSION['codice_fiscale'] = $codice_fiscale;
   $sql = "SELECT * FROM palestra.maestri WHERE codfiscale='$codice_fiscale'"; //vedo se è¨
maestro
   $result = mysql_query($sql);
   $count = mysql_num_rows($result);
   if ($count == 1) {
      $_SESSION['autorizzato'] = "ok";
      header('Refresh: 3; URL=Homemaestro.php');
      die;
   }
   $sql = "SELECT * FROM palestra.allievo WHERE codicefiscale='$codice_fiscale'";
//controprova se è allievo
   $result = mysql_query($sql);
   $count = mysql_num_rows($result);
   if ($count == 1) {
      $_SESSION['autorizzato'] = "allievo";
      header('Refresh: 3; URL=Home.php');
      die;
   }
} else {
   echo"login non riuscito su $myusername , sarai reindirizzato al login tra 3 secondi";
   header('Refresh: 3; URL=Progetto.html');
```

```
}
?>
```

come si nota se l'utente è maestro si pone il valore della chiave “autorizzato” ad “ok”, mentre se è allievo si usa “allievo”, quindi per accertarsi se gli utenti sono autenticati basta ripetere su ogni pagina di maestro il codice:

```
<?php
session_start();
if ($_SESSION['autorizzato']!="ok")
{
  echo "<h1>Area riservata, accesso negato.</h1>";
  echo "Per effettuare il login clicca <a href='progetto.html'>qui</a>";
  die;
}
?>
```

e su ogni pagina di allievo il codice:

```
<?php
session_start();
if ($_SESSION['autorizzato']!="allievo")
{
  echo "<h1>Area riservata agli allievi registrati, accesso negato.</h1>";
  echo "Per effettuare il login o la registrazione clicca <a href='progetto.html'>qui</a>";
  die;
}
?>
```

Inoltre in fase di acquisizione dei dati si è fatto uso delle funzioni php mysql_real_escape_string e di trim per rendere impossibile l'inserimento di stringhe malevole.

a titolo di esempio si riporta anche il codice della pagina che permette l'iscrizione ad un corso di un allievo:

ISCRIVITI_CORSO.PHP:

```
<?php
session_start();
if ($_SESSION['autorizzato'] != "allievo") {
    echo "<h1>Area riservata agli allievi registrati, accesso negato.</h1>";
    echo "Per effettuare il login o la registrazione clicca <a href='progetto.html'>qui</a>";
    die;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it" lang="it">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Iscriviti corso - Arti Marziali Castelfranco Veneto</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

```
        <base href=""/>
        <link href="Main.css" rel="stylesheet" type="text/css"/>
        <link href='http://fonts.googleapis.com/css?family=Shojumaru' rel='stylesheet'
type='text/css'/>

    </head>
    <body></br></br>
        <div class="menu">
            <ul class="lista-menu">
                <li><a href="Home.php">Home</a></li>
                <li>iscrizione corsi</li>
                <li><a href="tutte_le_gare.php">tutte le gare</a></li>
                <li><a href="i_maestri.php">Maestri</a></li>
            </ul>
        </div>

        <div id="lista_corsi">
            <?php
            require "tabelle.php";
            require 'connect.php';
            connect();
            $allievo = $_SESSION['codice_fiscale'];
            $sql = "SELECT disciplina FROM palestra.agonisti WHERE Allievo='$allievo'";
            $result = mysql_query($sql);
            $count = mysql_num_rows($result);
            if ($count >= 1) {
                $select = "SELECT nome,cognome,disciplina,prezzo,id_corso,fascia_eta from corsi left
join maestri on insegnante=codfiscale where agonistico='1' and fascia_eta=fasciaanni('$allievo') and
id_corso not in (select corso from palestra.iscrizione where allievo='$allievo') and (";
                while ($row = mysql_fetch_array($result)) {
                    $select = $select . "disciplina='$row[0]' or ";
                }
                $select = $select . "false)";
                $result = mysql_query($select);
                $count = mysql_num_rows($result);
                if ($count >= 1) {
                    echo '<h2>corsi per agonisti a cui puoi iscriverti</h2>';
                    table_start(array("Nome insegnante", "Cognome insegnante", "disciplina", "prezzo",
"id corso", "fascia d'et&agrave"));
                    while ($row = mysql_fetch_row($result))
                        table_row($row);
                    table_end();
                } else {
                    echo '<h3>nessun corso per agonisti a cui puoi iscriverti</h3>';
                }
            }
            $query = "SELECT nome,cognome,disciplina,prezzo,id_corso,fascia_eta from corsi left join
maestri on insegnante=codfiscale where agonistico='0' and fascia_eta=fasciaanni('$allievo') and
id_corso not in (select corso from palestra.iscrizione where allievo='$allievo')";
            $result = mysql_query($query);
            $count = mysql_num_rows($result);
            if ($count >= 1) {
```

```
            echo '<h2>corsi aperti anche agli amatori</h2>';
            table_start(array("Nome insegnante", "Cognome insegnante", "disciplina", "prezzo", "id
corso", "fascia d'et&agrave"));
            while ($row = mysql_fetch_row($result))
               table_row($row);
            table_end();
         } else {
            echo '<h3>nessun corso per amatori a cui puoi iscriverti</h3>';
         }
         ?>
      </div>
      <div id="iscrizione corso" >
         <h2>Iscriviti ad un corso</h2>
         <form method="post"  action="actioncorso.php" id="fg">
            <input type="text" id="nome" name="idcorso" placeholder="id del corso a cui vuoi
iscriverti" required style="width:15em" />
            <input type="submit"  name="conferma" value="Conferma" />
            <input type="button"  name="annulla" value="Annulla" onclick="location.href =
'Home.php'" />
         </form>
      </div>
   </body>
</html>
```

e il relativo php:

ACTIONCORSO.PHP:

```
<?php

session_start();
require 'connect.php';
connect();
// recupero i campi di tipo "stringa"
$idcorso = trim($_POST['idcorso']);

// verifico se devo eliminare gli slash inseriti automaticamente da PHP
if (get_magic_quotes_gpc()) {
   $idcorso = stripslashes($idcorso);
}

$idcorso = mysql_real_escape_string($idcorso);
$allievo = $_SESSION['codice_fiscale'];
$sql = "SELECT * FROM palestra.agonisti as ag left join palestra.corsi as c on
c.disciplina=ag.disciplina WHERE c.id_corso='$idcorso' and ag.allievo='$allievo' and
c.fascia_eta=fasciaanni('$allievo') and c.agonistico='1'"; //vedo se è un corso per agonisti
$result = mysql_query($sql);
$count = mysql_num_rows($result);
$sql2 = "SELECT * FROM palestra.allievo join palestra.corsi  WHERE id_corso='$idcorso' and
codicefiscale='$allievo' and agonistico='0' and fascia_eta=fasciaanni('$allievo')"; // vedo se è per
non agonisti
```

```
$result2 = mysql_query($sql2);
$count2 = mysql_num_rows($result2);
if (!$count and ! $count2) {
   echo"corso non esistente o non disponibile per te";
   header('Refresh: 5; URL=iscriviti_corso.php');
   die;
}
$codfiscale = $_SESSION['codice_fiscale'];
// preparo la query
$query = "INSERT INTO Palestra.iscrizione (allievo,corso)
                              VALUES ('$codfiscale','$idcorso')";

// invio la query
$result = mysql_query($query);

// controllo l'esito
if (!$result) {
   echo"operazione non riuscita, sarai reindirizzato al form da ricompilare tra 3 secondi";
   header('Refresh: 3; URL=iscriviti_corso.php');
   die("Errore nella query $query: " . mysql_error());
}
$sql = "SELECT noag.disciplina FROM palestra.non_agonisti as noag left join palestra.corsi as c
on c.disciplina=noag.disciplina WHERE c.id_corso='$idcorso' and noag.allievo='$allievo' and
c.agonistico='0'";
$result = mysql_query($sql);
$count = mysql_num_rows($result);
if ($count == 1) {
   $row = mysql_fetch_array($result);
   $disciplina = $row[0];
   $sql = "UPDATE palestra.non_agonisti SET Amatore='1' WHERE
non_agonisti.Disciplina='$disciplina' AND non_agonisti.Allievo='$allievo';";
   $result = mysql_query($sql);
   if (!$result) {
      echo"operazione non riuscita, non è stato possibile renderti amatore”;
      header('Refresh: 3; URL=iscriviti_corso.php');
      die("Errore nella query $query: " . mysql_error());
   }
}
echo("Inserimento effettuato con successo, sarai reindirizzato a breve");
header('Refresh: 3; URL=Home.php');
?>
```

Note e istruzioni per l'installazione:

Il seguente progetto è stato sviluppato e testato su windows 10, con PHP aggiornato alla versione 5.6.12, apache aggiornato alla versione 2.4.16 e mysql aggiornato alla versione 5.6.26, e testato anche su windows 7 e windows 8.1, per cui le indicazioni fornite sono da ritenersi valide solo sulle piattaforme sopraelencate.

Per installare il progetto nel computer in uso è necessario, dopo aver spostato la cartella progetto nella directory di apache dedita allo scopo della visualizzazione delle pagine web, caricare i file contenuti nella cartella “crea e popola” tramite phpmyadmin o qualsiasi altra interfaccia grafica che permetta l'importazione di file sql nel seguente ordine: tabella.sql, functiontrigger.sql, popola.sql. Si segnala che se non si hanno i privilegi necessari per creare database, è necessario rinominarne uno esistente in “Palestra”, cancellare la prima riga di tabella.sql e procedere al caricamento nell'ordine già indicato.

È inoltre necessario modificare il file connect.php come è indicato:

```
<?php

function connect() {
   $nomehost = "localhost";//qui va il nome dell'host
   $nomeuser = "root";//qui va lo username del client mysql del pc in uso
   $password = "admin";//qui va la password sql del pc in uso, se presente
   $dbname = 'Palestra';
   $connessione = mysql_connect($nomehost, $nomeuser, $password);
   $database_select = mysql_select_db($dbname, $connessione);
}

?>
```

si consiglia inoltre per una migliore presentazione, di disabilitare i messaggi di warning su schermo di PHP.