Relazione sul progetto di Basi di Dati

Scuola sub Tizio&Caio

Vanin Giacomo,

Matricola 1026988

Abstract

Il progetto modella la gestione di una scuola sub. In particolare vengono gestiti i corsi offerti dalla scuola con le relative lezioni ed esami, le immersioni fatte da chi frequenta la scuola, le riunioni dei soci e gli eventi organizzati, le attrezzature con possibilità di noleggio. Si vuole poter ottenere informazioni riguardanti il numero di sub che partecipano ad una certa immersione, il numero delle attrezzature noleggiate e quelle ancora disponibili, un calendario delle lezioni di un certo corso. Si vuole poter fare una ricerca avanzata su alcune tabelle e si deve dare la possibilità di intervenire sul database.

1. Analisi dei requisiti

Si vuole realizzare una base di dati e la relativa applicazione web che modelli alcune classi per gestire una scuola sub e in particolare i suoi corsi con le relative lezioni, gli eventi e le immersioni programmate.

Di ogni corso interessa il nome (come sigla) del corso, la data dell'esame il codice del corso che lo identifica univocamente, e il costo in € che comprende tutte le spese. Inoltre interessa sapere quali persone frequentano il corso.

Delle lezioni interessano l'id della lezione, la data e l'ora, il luogo dove si svolgerà o si è svolta la lezione e il corso a cui la lezione è legata.

Di tutte le persone interessa nome, cognome, data di nascita, e-mail e n° di telefono. Le persone possono essere soci o sub; dei primi interessa la data in cui è diventato socio, la quota versata e lo sconto sulle attrezzature; degli altri interessa il n° dell'ultimo brevetto conseguito e in n° di immersioni che ha fatto. Inoltre possono esserci persone che non sono né soci né sub e soci che sono sub e viceversa. Di tutti e tre interessa l'id che li identifica.

Degli eventi interessa l'id che lo identifica, il nome dell'evento, la data, il luogo dove si è svolto o si svolgerà l'evento, i soci che hanno organizzato l'evento e le persone che vi hanno partecipato o che vi parteciperanno.

Delle riunioni si vuole sapere il codice, la data in cui si è svolta o si svolgerà ogni riunione e i soci che vi hanno partecipato.

Delle attrezzature della scuola si vuole sapere il tipo (ad esempio muta, maschera, GAV, ecc.), il prezzo di listino per il noleggio, chi ha noleggiato un certo tipo di attrezzatura e il prezzo a cui l'ha noleggiata (varia a seconda che chi noleggia sia socio oppure no).

Dei brevetti interessa il n° identificativo, il nome di tale brevetto (sigla) e la massima profondità a cui permette di andare lo specifico brevetto. Inoltre interessa sapere chi è il sub possessore di tale brevetto.

Delle immersioni interessa sapere chi sono i sub che hanno partecipato a tale immersione, la data e l'ora dell'immersione, la città dove si è affrontata l'immersione, il luogo (Mare o Lago), da dove parte l'immersione, il costo per poter affrontare l'immersione e la profondità massima a cui si può arrivare in quell'immersione. Un'immersione è identificata da un codice.

Per i sub ci sono delle condizioni particolari. Una persona può essere considerata sub se possiede almeno un brevetto e se ha fatto almeno un'immersione. Pertanto quando si inserisce nel database un nuovo sub si deve anche aggiornare la tabella brevetti con almeno un suo brevetto e la tabella immersioni con almeno una sua immersione.

Di ogni utente del sistema si deve conoscere l'username (unico per ogni utente) e l'hash della password. Ogni utente deve poter effettuare delle ricerche avanzate, inserire nuovi dati, modificare ed eliminare dati nel database rispettando le condizioni imposte.

2. Progettazione concettuale

2.1 Descrizione testuale delle classi

2.1.1 Persone

La classe persone contiene le informazioni essenziali sugli individui frequentanti la scuola sub. In questa classe non è specificato se gli individui siano sub o soci; vengono semplicemente riportati il nome, e alcune informazioni per poter distinguere una persona dall'altra.

Attributi

- Nome: *varchar(45)* - nome della persona
- Cognome: *varchar(45)* - cognome della persona
- Datanascita: *date* - la sua data di nascita
- Email: *varchar(45)* - la sua mail
- Telefono: *varchar(10)* - il suo numero di telefono

2.1.2 Sub

La classe rappresenta le persone che sono dei sub.

Attributi

- Totimmersioni *int* - totale delle immersioni da quando è sub

2.1.3 Soci

La classe rappresenta le persone che sono anche soci.

Attributi

- Dataiscrizione *date* - data di quando è diventato socio
- Quota *decimal(10,2)* - soldi che dà per il mantenimento della scuola
- Scontoatt *varchar(45)* - percentuale di sconto sulle attrezzature

2.1.4 Corsi

La classe tiene conto dei corsi offerti dalla scuola sub. Sono presenti i corsi che si stanno svolgendo, che si sono già svolti, ma anche quelli che devono ancora iniziare e di cui non si sa la data d'esame.

Attributi

- Nome *varchar(45)* - sigla del corso
- Esame *datetime* - data e ora dell'esame
- Costo *decimal(10,2)* - prezzo del corso

2.1.5 Lezioni

La classe contiene le informazioni sulle lezioni relative ai corsi in svolgimento o già svolti.

Attributi

- Data *date* - data della lezione
- Ora *time* - ora di inizio della lezione
- Luogo *(Scuola, Piscina, Mare, Lago)* - luogo in cui si svolge la lezione

2.1.6 Eventi

La classe raccoglie tutti gli eventi organizzati dalla scuola sub sia quelli già avvenuti, sia quelli in programma.

Attributi

- Nome *varchar(45)* - nome dell'evento
- Data *date* - data dell'evento
- Luogo *varchar(45)* - luogo in cui si svolge o si svolgerà l'evento

2.1.7 Riunioni

La classe raccoglie le riunioni fatte dai soci della scuola sub.

Attributi

- Data *date* - data della riunione

2.1.8 Attrezzaturescuola

La classe contiene le attrezzature che possiede la scuola sub e che possono essere noleggiate.

Attributi

- Tipo *varchar(45)* - il tipo di attrezzatura
- Prezzonoleggio *decimal* - prezzo del noleggio da listino

2.1.9 Brevetti

La classe tiene conto dei brevetti che un sub possiede.

Attributi

- Brevetto *varchar(45)* - il numero di riconoscimento identificativo del brevetto
- Nome *varcahr(45)* - sigla che indica il tipo di brevetto
- Maxprofondita *int* - la massima profondità raggiungibile con quel brevetto

2.1.10 Immersioni

La classe contiene le immersioni fatte dai sub che frequentano la scuola sub.

Attributi

- Data *date* - data dell'immersione
- Ora *time* - ora dell'immersione
- Citta *varchar(45)* - la città in cui si è fatta o si farà l'immersione
- Luogo *(Mare, Lago)* - indica qual è il tipo di acqua dell'immersione
- Tipo (*Diurna, Notturna)* - il tipo di immersione
- Inizioimm *(Spiaggia, Gommone, Barca)* - indica il punto di partenza
- Costo *decimal* - indica il prezzo
- Profonditamax *int* - indica la profondità massima del fondale

2.2 Descrizione testuale delle associazioni

2.2.1 Persone - Eventi: "Partecipano a"

Molteplicità N:M una persona può partecipare a uno o più eventi, un evento può essere partecipato da una o più persone.

Totalità: parziale verso persone/parziale verso eventi una persona può partecipare a nessun evento, un evento può essere partecipato da nessuna persona (se si deve ancora svolgere).

2.2.2 Persone - Corsi: "Frequentano"

Molteplicità N:M una persona può frequentare uno o più corsi, un corso può essere frequentato da una o più persone.

Totalità: parziale verso persone/parziale verso corsi una persona può frequentare nessun corso, un corso può essere frequentato da nessuno (se non ancora iniziato).

2.2.3 Corsi- Lezioni: "Divisi in"

Molteplicità 1:N un corso può essere diviso in più lezioni, una lezione divide un solo corso.

Totalità: totale verso corsi/parziale verso lezioni un corso può essere diviso in nessuna lezione (se non ancora iniziato) o più, una lezione deve dividere per forza un corso.

2.2.4 Soci- Eventi: "Organizzano"

Molteplicità N:M un socio può organizzare più eventi, un evento può essere organizzato da più soci.

Totalità: parziale verso eventi/totale verso soci un evento deve essere organizzato da almeno un socio, un socio può organizzare nessun evento.

2.2.5 Soci - Riunioni: "Partecipano a"

Molteplicità N:M un socio può partecipare a più riunioni, una riunione può essere partecipata da più soci.

Totalità: parziale verso riunioni/parziale verso soci una riunione può essere partecipata da nessun socio (se non ancora avvenuta), un socio può partecipare a nessuna riunione.

2.2.6 Sub - Immersioni: "Partecipano a"

Molteplicità N:M un sub può partecipare a più immersioni, un'immersione può essere partecipata da più sub.

Totalità: totale verso sub/totale verso immersioni un sub deve aver fatto almeno un'immersione, un'immersione deve essere stata fatta da almeno un sub.

2.2.7 Sub - Brevetti: "Possiedono"

Molteplicità 1:N un sub può possedere più brevetti, un brevetto è posseduto da un solo sub.

Totalità: totale verso sub/totale verso brevetti un sub deve possedere almeno un brevetto (per essere considerato tale), un brevetto deve essere posseduto da un sub.

2.2.8 Sub - AttrezzatureScuola: "Noleggiano"

Molteplicità: 1:N un sub può noleggiare più attrezzature, un'attrezzatura può essere noleggiata da un solo sub(per volta).

Totalità: parziale verso Sub/parziale verso AttrezzatureScuola un sub può non aver noleggiato nessuna attrezzatura, un'attrezzatura può non essere stata noleggiata da nessuno.

2.3 Descrizione delle gerarchie

È presente un'unica gerarchia che ha per superclasse la classe Persone e come sottoclassi le classi Soci e Sub.

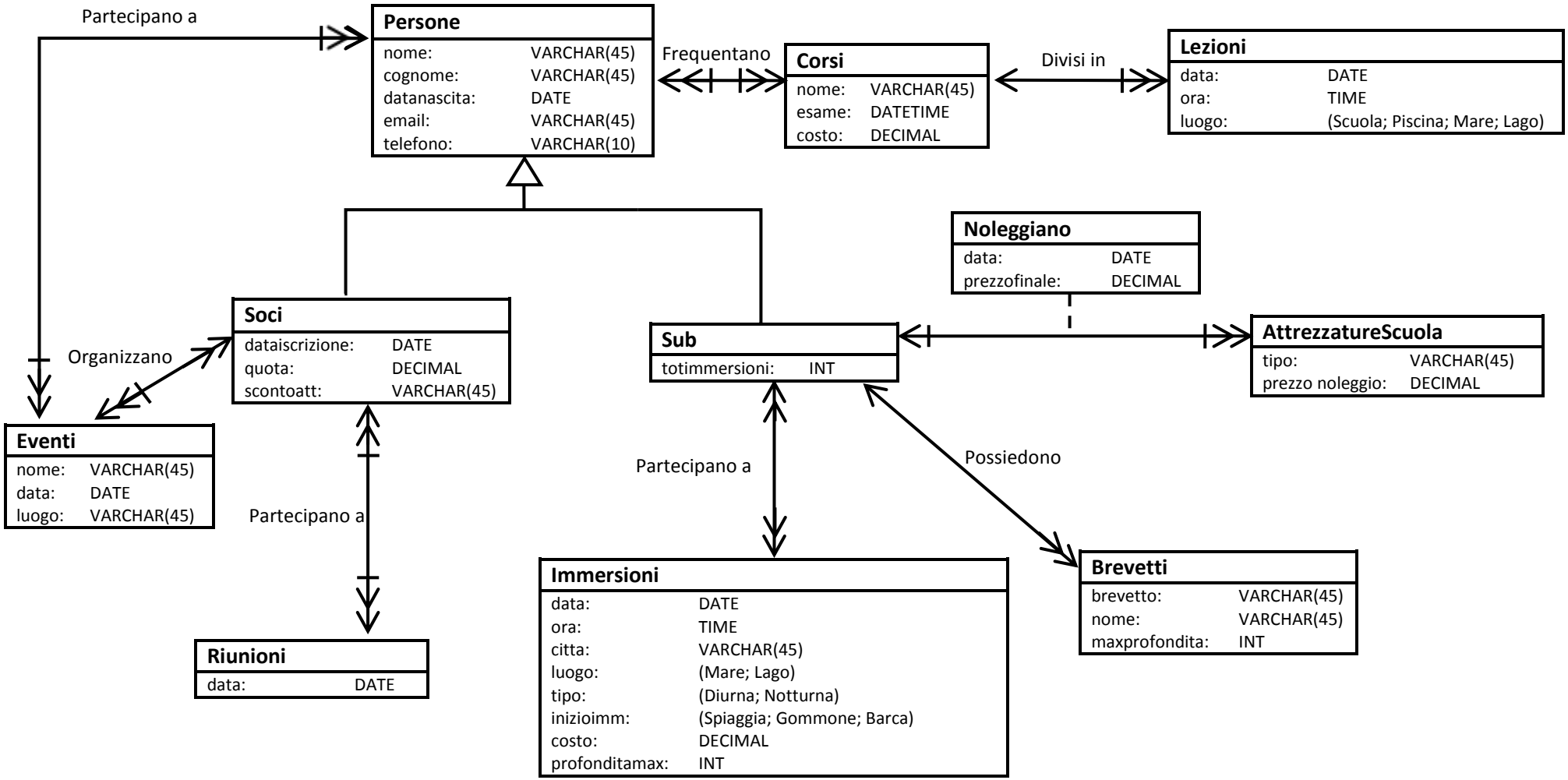
La gerarchia ha soltanto le sottoclassi scorrelate, infatti non rispetta i vincoli di:

- Disgiunzione, perché ci possono essere sub che sono anche soci.
- Copertura, perché possono esserci persone frequentanti la scuola che non sono né soci né sub.

2.4 Descrizione dei vincoli di integrità aggiuntivi

Non sono presenti vincoli di integrità che non siano stati catturati dallo schema concettuale.

Schema concettuale



3. Progettazione logica

3.1 Descrizione dello schema relazionale

- **Persone (<u>idpersona</u>: int, nome: varchar, cognome: varchar, datanascita: date, email: varchar, telefono: varchar)**
  - PK (Idpersona)
  - UNIQUE (email, telefono)
  - NOT NULL (nome, cognome, datanascita)
- **Eventi (<u>idevento</u>: int, nome: varchar, data: date, luogo: varchar)**
  - PK (idevento)
  - NOT NULL (nome, data, luogo)
- **PersoneEventi (<u>persona</u>*: int, <u>evento</u>*: int)**
  - PK (persona, evento),
  - persona FK (Persone),
  - evento FK (Eventi)
- **Soci (<u>idsocio</u>*: int, dataiscrizione: date, quota: decimal, scontoatt: varchar)**
  - PK (idsocio),
  - idsocio FK (Persone),
  - NOT NULL (dataiscrizione, quota, scontoatt)
- **Sub (<u>idsub</u>*: int, totimmersioni: int)**
  - PK (idsub),
  - idsub FK (Persone)
- **Corsi (<u>codcorso</u>: int, nome: varchar, esame: datetime, costo: decimal)**
  - PK (codcorso),
  - NOT NULL (nome, costo)
- **PersoneCorsi (<u>persona</u>*: int, <u>corso</u>*: int)**
  - PK (persona, corso),
  - persona FK (Persone),
  - corso FK (Corsi)

- **Lezioni (<u>idlez</u>: int, data: date, ora: time, luogo: enum {'Scuola', 'Piscina', 'Mare', 'Lago'}, corso*: int)**
  - PK (idlez),
  - corso FK (Corsi),
  - NOT NULL (data, ora, corso)
- **EventiSoci (<u>evento</u>*: int, <u>socio</u>*: int)**
  - PK (evento, socio),
  - evento FK (Eventi),
  - socio FK (Soci)
- **Brevetti (<u>brevetto</u>: varchar, nome: varchar, maxprofondita: int, sub*: int)**
  - PK (brevetto),
  - sub FK (Sub),
  - NOT NULL (nome, sub)
- **Immersioni (<u>codimm</u>: int, data: date, ora: time, citta: varchar, luogo: enum {'Mare', 'Lago'}, tipo: enum {'Diurna', 'Notturna'}, inizioimm: enum {'Spiaggia', 'Gommone', 'Barca'}, costo: decimal, profonditamax: int)**
  - PK (codimm),
  - NOT NULL (data, ora, citta, costo, profonditamax)
- **ImmersioniSub (<u>immersione</u>*: int, <u>sub</u>*: int)**
  - PK (immersione, sub),
  - immersione FK (Immersioni),
  - sub FK (Sub)
- **AttrezzatureScuola (<u>codatt</u>: int, tipo: varchar, prezzonoleggio: decimal)**
  - PK (codatt),
  - NOTNULL (tipo, prezzonoleggio)
- **Noleggi (sub*: int, <u>att</u>*: int, data: date, prezzofinale: decimal)**
  - PK (att),
  - sub FK (Sub),
  - att FK (AttrezzatureScuola),

  - NOT NULL (sub, data, prezzofinale)

- **Riunioni (<u>codriunione</u>: int, data: date)**
  - PK (codriunione),
  - NOT NULL (data)
- **SociRiunioni (<u>socio</u>*: int, <u>riunione</u>*: int)**
  - PK (socio, riunione),
  - socio FK (Soci),
  - riunione FK (Riunioni)
- **Users (email: varchar, username: varchar, password: varchar)**
  - UNIQUE (email, username),
  - NOT NULL (email, username, password)

3.2 Considerazioni sulle scelte progettuali

3.2.1 Classe Users

La classe Users non è in relazione con nessun'altra classe del database quindi si è scelto di non visualizzarla nello schema concettuale né in quello relazionale.

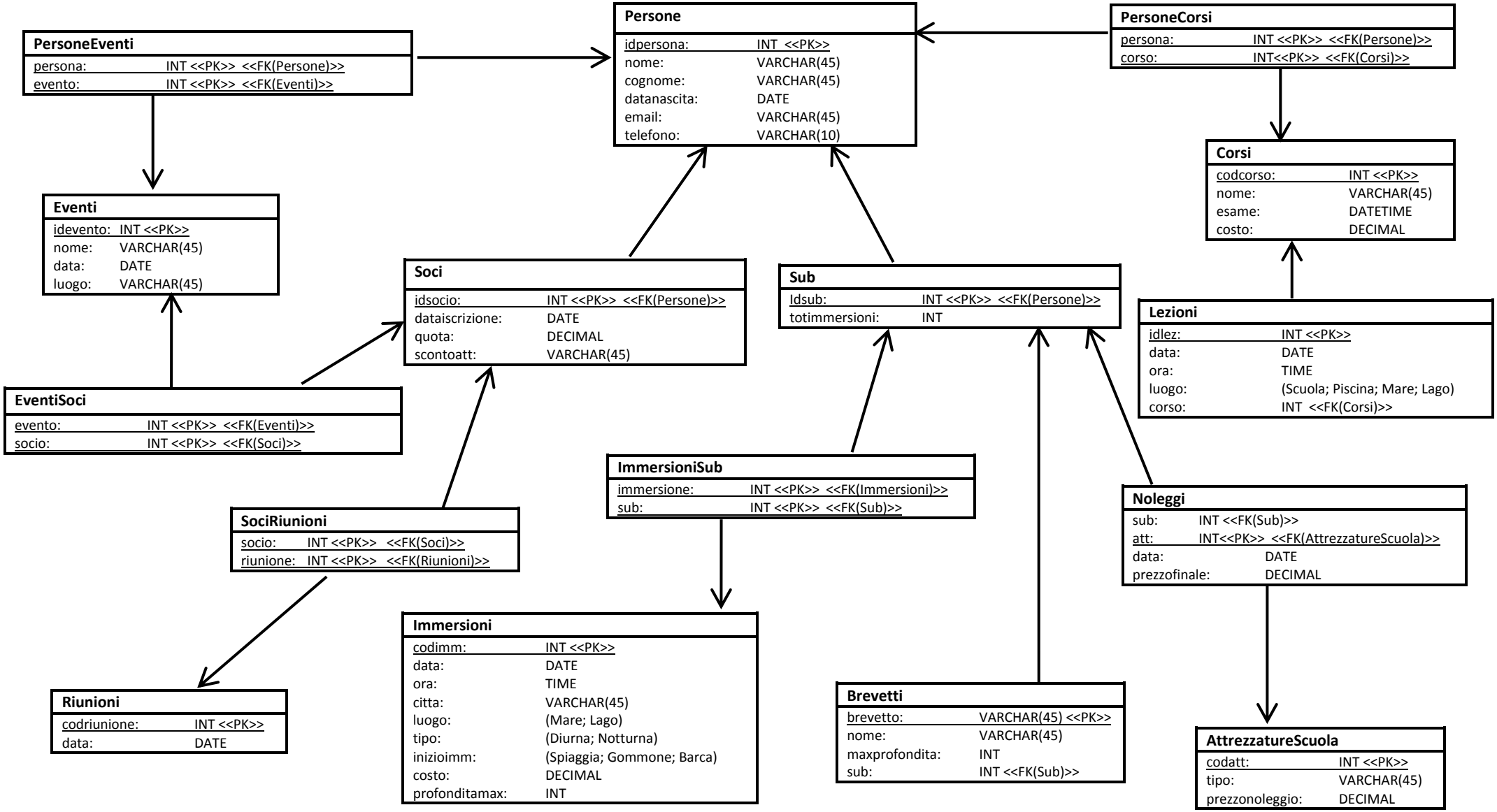
3.2.2 Trasformazione delle gerarchie

La gerarchia, che aveva come superclasse la classe Persone e come sottoclassi la classe Sub e la classe Soci nel modello concettuale, è stata resa nel modello relazionale con un partizionamento verticale. I motivi di tale scelta sono stati:

- Non si è scelto un partizionamento orizzontale perché ci sarebbe stato il problema del collocamento delle persone che non sono né soci né sub.
- Non si è scelta la soluzione a tabella unica perché per la maggior parte delle persone inserite nella tabella molti campi sarebbero stati NULL e quindi inutili.

3.2.3 Vincoli semantici catturati dallo schema

- Il campo email della classe Users deve essere formato da lettere o numeri - @ -lettere -.it o .com.
- Il campo prezzonoleggio della classe AttrezzatureScuola deve essere uguale per attrezzature dello stesso tipo.
- Il campo prezzofinale della classe Noleggi deve corrispondere al campo prezzonoleggio dell'attrezzatura, scontato eventualmente della percentuale indicata in scontoatt se la persona che noleggia l'attrezzatura è un socio.


Persone
idpersona: INT <<PK>>
nome: VARCHAR(45)
cognome: VARCHAR(45)
datanascita: DATE
email: VARCHAR(45)
telefono: VARCHAR(10)
PersoneEventi
persona: INT <<PK>> <<FK(Persone)>>
evento: INT <<PK>> <<FK(Eventi)>>
PersoneCorsi
persona: INT <<PK>> <<FK(Persone)>>
corso: INT<<PK>> <<FK(Corsi)>>
Corsi
codcorso: INT <<PK>>
nome: VARCHAR(45)
esame: DATETIME
costo: DECIMAL
Eventi
idevento: INT <<PK>>
nome: VARCHAR(45)
data: DATE
luogo: VARCHAR(45)
Soci
idsocio: INT <<PK>> <<FK(Persone)>>
dataiscrizione: DATE
quota: DECIMAL
scontoatt: VARCHAR(45)
Sub
Idsub: INT <<PK>> <<FK(Persone)>>
totimmersioni: INT
Lezioni
idlez: INT <<PK>>
data: DATE
ora: TIME
luogo: (Scuola; Piscina; Mare; Lago)
corso: INT <<FK(Corsi)>>
EventiSoci
evento: INT <<PK>> <<FK(Eventi)>>
socio: INT <<PK>> <<FK(Soci)>>
ImmersioniSub
immersione: INT <<PK>> <<FK(Immersioni)>>
sub: INT <<PK>> <<FK(Sub)>>
Noleggi
sub: INT <<FK(Sub)>>
att: INT<<PK>> <<FK(AttrezzatureScuola)>>
data: DATE
prezzofinale: DECIMAL
SociRiunioni
socio: INT <<PK>> <<FK(Soci)>>
riunione: INT <<PK>> <<FK(Riunioni)>>
Immersioni
codimm: INT <<PK>>
data: DATE
ora: TIME
citta: VARCHAR(45)
luogo: (Mare; Lago)
tipo: (Diurna; Notturna)
inizioimm: (Spiaggia; Gommone; Barca)
costo: DECIMAL
profonditamax: INT
Riunioni
codriunione: INT <<PK>>
data: DATE
Brevetti
brevetto: VARCHAR(45) <<PK>>
nome: VARCHAR(45)
maxprofondita: INT
sub: INT <<FK(Sub)>>
AttrezzatureScuola
codatt: INT <<PK>>
tipo: VARCHAR(45)
prezzonoleggio: DECIMAL

4. Definizione dello schema logico in MySql

```
CREATE TABLE Persone (
idpersona       INT PRIMARY KEY,
nome            VARCHAR(45) NOT NULL,
cognome         VARCHAR(45) NOT NULL,
datanascita     DATE NOT NULL,
email           VARCHAR(45) UNIQUE,
telefono        VARCHAR(10) UNIQUE
) ENGINE=InnoDB;

CREATE TABLE Soci (
idsocio         INT PRIMARY KEY,
dataiscrizione  DATE NOT NULL,
quota           DECIMAL(10,2) NOT NULL,
scontoatt       VARCHAR(45) NOT NULL,
FOREIGN KEY (idsocio) REFERENCES Persone (idpersona) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Sub (
idsub                   INT PRIMARY KEY,
totimmersioni   INT,
FOREIGN KEY (idsub) REFERENCES Persone (idpersona) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Corsi (
codcorso        INT PRIMARY KEY,
nome            VARCHAR(45) NOT NULL,
esame           DATETIME,
costo           DECIMAL(10,2) NOT NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE PersoneCorsi (
persona		INT,
corso		INT,
PRIMARY KEY (persona, corso),
FOREIGN KEY (persona) REFERENCES Persone (idpersona) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (corso) REFERENCES Corsi (codcorso) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Lezioni (
idlez		INT PRIMARY KEY,
data		DATE NOT NULL,
ora	TIME NOT NULL,
luogo		ENUM('Scuola', 'Piscina', 'Mare', 'Lago'),
corso		INT NOT NULL,
FOREIGN KEY (corso) REFERENCES Corsi (codcorso) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Eventi (
idevento	INT PRIMARY KEY,
nome		VARCHAR(45) NOT NULL,
data		DATE NOT NULL,
luogo		VARCHAR(45) NOT NULL
) ENGINE=InnoDB;

CREATE TABLE EventiSoci (
evento		INT,
socio		INT,
PRIMARY KEY (evento, socio),
FOREIGN KEY (evento) REFERENCES Eventi (idevento) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
FOREIGN KEY (socio) REFERENCES Soci (idsocio) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Brevetti (
brevetto            VARCHAR(45) PRIMARY KEY,
nome                VARCHAR(45) NOT NULL,
maxprofondita       INT,
sub                 INT NOT NULL,
FOREIGN KEY (sub) REFERENCES Sub (idsub) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Immersioni (
codimm INT PRIMARY KEY,
data            DATE NOT NULL,
ora             TIME NOT NULL,
citta           VARCHAR(45) NOT NULL,
luogo           ENUM('Mare', 'Lago'),
tipo            ENUM('Diurna', 'Notturna'),
inizioimm       ENUM('Spiaggia', 'Gommone', 'Barca'),
costo           DECIMAL(10,2) NOT NULL,
profonditamax   INT NOT NULL
) ENGINE=InnoDB;

CREATE TABLE ImmersioniSub (
immersione      INT,
sub             INT,
PRIMARY KEY (immersione,sub),
FOREIGN KEY (immersione) REFERENCES Immersioni (codimm) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (sub) REFERENCES Sub (idsub) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE AttrezzatureScuola (
Codatt                  INT PRIMARY KEY,
tipo                    VARCHAR(45) NOT NULL,
prezzonoleggio          DECIMAL(10,2) NOT NULL
) ENGINE=InnoDB;

CREATE TABLE Noleggi (
sub             INT NOT NULL,
att             INT PRIMARY KEY,
data            DATE NOT NULL,
prezzofinale    DECIMAL(10,2) NOT NULL,
FOREIGN KEY (sub) REFERENCES Sub (idsub) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (att) REFERENCES AttrezzatureScuola (codatt) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE PersoneEventi (
evento          INT,
personaINT,
PRIMARY KEY (persona, evento),
FOREIGN KEY (persona) REFERENCES Persone (idpersona) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (evento) REFERENCES Eventi (idevento) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Riunioni (
codriunione     INT PRIMARY KEY,
data            DATE NOT NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE SociRiunioni (
socio           INT,
riunione        INT,
PRIMARY KEY(socio, riunione),
FOREIGN KEY (socio) REFERENCES Soci (idsocio) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (riunione) REFERENCES Riunioni (codriunione) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Users (
email           varchar(45) NOT NULL,
username        varchar(45) NOT NULL,
password        varchar(45) NOT NULL,
UNIQUE KEY email (email)
UNIQUE KEY username (username)
) ENGINE=InnoDB;
```

5. Funzioni

Di seguito vengono proposte le funzioni utilizzate nel progetto.

5.1 Funzione per l'individuazione della massima profondità raggiungibile da un sub

```
DELIMITER $$
CREATE FUNCTION max_prof (sub INT) RETURNS int (11)

BEGIN
DECLARE m_p INT;
SELECT MAX (b.maxprofondita) INTO m_p FROM Brevetti b WHERE b.sub=sub;
RETURN m_p;
END$$
DELIMITER ;
```

Questa semplice funzione restituisce la variabile m_p in cui è memorizzata la massima profondità a cui può arrivare il sub passato come parametro alla funzione.

5.2 Funzione che calcola la spesa totale di una persona

```
DELIMITER $$

CREATE FUNCTION Spesa_totale_annua (id_persona INT) RETURNS decimal (10,2)


BEGIN


DECLARE tot_spesa DECIMAL (10,2);

DECLARE t_cc DECIMAL (10,2);

DECLARE t_ci DECIMAL (10,2);

DECLARE t_cn DECIMAL (10,2);

DECLARE q_s DECIMAL (10,2);


SELECT IFNULL (SUM (cpc.costo), 0) INTO t_cc FROM Corsiperscomp cpc WHERE cpc.persona=id_persona;


SELECT IFNULL (SUM (isc.costo), 0) INTO t_ci FROM  Immsubcomp isc JOIN Persone p ON isc.sub=p.idpersona
WHERE p.idpersona=id_persona;


SELECT IFNULL (SUM (n.prezzofinale), 0) INTO t_cn FROM Persone p JOIN Sub su ON p.idpersona=su.idsub JOIN
Noleggi n ON su.idsub=n.sub WHERE p.idpersona=id_persona;


SELECT IFNULL (SUM (s.quota), 0) INTO q_s FROM Persone p JOIN Soci s ON p.idpersona=s.idsocio WHERE
p.idpersona=id_persona;


SET tot_spesa=t_cc+t_ci+t_cn+q_s;

RETURN tot_spesa;

END$$

DELIMITER ;
```

Questa funzione restituisce la variabile tot_spesa in cui è memorizzata la spesa totale della persona passata come parametro alla funzione. La spesa totale è calcolata sommando le spese per i corsi, le spese per le immersioni, la quota associativa e le spese per il noleggio delle attrezzature.

6. Procedure

Di seguito vengono proposte le procedure utilizzate nel progetto.

6.1 Procedura che seleziona i sub che hanno un certo tipo di brevetto

```
DELIMITER $$

CREATE PROCEDURE div_brev (IN brev VARCHAR (45))

BEGIN

SELECT b.sub, p.nome, p.cognome

FROM brevetti b JOIN persone p ON b.sub=p.idpersona

WHERE b.nome=brev;

END$$

DELIMITER ;
```

Questa semplice procedura seleziona i sub che hanno un brevetto con lo stesso nome del parametro passato alla procedura.

6.2 Procedura che seleziona le diverse spese di una persona

```
DELIMITER $$


CREATE PROCEDURE spese_div (IN id_persona INT)

BEGIN

DECLARE t_cc DECIMAL (10,2);

DECLARE t_ci DECIMAL (10,2);

DECLARE t_cn DECIMAL (10,2);

DECLARE q_s DECIMAL (10,2);


SELECT IFNULL (SUM (cpc.costo), 0) INTO t_cc

FROM Corsiperscomp cpc

WHERE cpc.persona=id_persona;


SELECT IFNULL (SUM (isc.costo), 0) INTO t_ci

FROM Immsubcomp isc JOIN Persone p ON isc.sub=p.idpersona

WHERE p.idpersona=id_persona;


SELECT IFNULL (SUM (n.prezzofinale), 0) INTO t_cn
```

```
FROM Persone p JOIN Sub su ON p.idpersona=su.idsub JOIN Noleggi n ON su.idsub=n.sub

WHERE p.idpersona=id_persona;


SELECT IFNULL (SUM (s.quota), 0) INTO q_s

FROM Persone p JOIN Soci s ON p.idpersona=s.idsocio

WHERE p.idpersona=id_persona;


CREATE TEMPORARY TABLE spese (

spesecorsi DECIMAL(10,2),

speseimmersioni DECIMAL(10,2),

spesenoleggi DECIMAL(10,2),

spesasocio DECIMAL(10,2)

);


INSERT INTO spese VALUES (t_cc, t_ci, t_cn, q_s);

SELECT * FROM spese;

END$$

DELIMITER ;
```

Questa procedura seleziona le diverse spese della persona con id che corrisponde al parametro passato alla procedura. Vengono selezionate prima singolarmente e poi memorizzate in una tabella temporanea.

7. Trigger

Di seguito vengono proposti i trigger utilizzati nel progetto.

7.1 Trigger che verifica il numero di brevetti di un sub prima della cancellazione

```
DELIMITER $$


CREATE TRIGGER verifica_brev BEFORE DELETE ON Brevetti

FOR EACH ROW

BEGIN

DECLARE quanti INT;
```

```
DECLARE ecco INT;

SELECT COUNT(*) INTO quanti FROM Brevetti WHERE sub = OLD.sub;

IF quanti <2 THEN

SIGNAL SQLSTATE '10000'

SET MESSAGE_TEXT='Un sub rimarrebbe senza brevetto.';

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica che il brevetto da cancellare non sia l'unico del sub che lo possiede. In caso contrario fa scattare un errore che blocca la cancellazione.

7.2 Trigger che verifica il numero di brevetti di un sub prima dell'aggiornamento

```
DELIMITER $$

CREATE TRIGGER verifica_brev1 BEFORE UPDATE ON Brevetti

FOR EACH ROW

BEGIN

DECLARE quanti INT;

DECLARE ecco INT;

IF OLD.sub != NEW.sub

THEN SELECT COUNT(*) INTO quanti FROM Brevetti WHERE sub = OLD.sub;

IF quanti <2 THEN

SIGNAL SQLSTATE '10001'

SET MESSAGE_TEXT='Un sub rimarrebbe senza brevetto.';

END IF;

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica il numero di brevetti di un sub prima dell'aggiornamento. Se un brevetto è posseduto da un sub e si vuole cambiare il sub che lo possiede, prima viene verificato che il brevetto non sia l'unico posseduto da quel sub. In caso contrario fa scattare un errore che blocca l'aggiornamento.

7.3 Trigger che verifica il numero di immersioni di un sub prima della cancellazione

```
DELIMITER $$

CREATE TRIGGER verifica_imm BEFORE DELETE ON ImmersioniSub

FOR EACH ROW

BEGIN

DECLARE quante INT;

DECLARE ecco INT;

SELECT COUNT(*) INTO quante FROM ImmersioniSub WHERE sub = OLD.sub;

IF quante <2 THEN

SIGNAL SQLSTATE '10002'

SET MESSAGE_TEXT='Un sub risulterebbe non aver fatto nessuna immersione.';

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica che l'immersione che si vuole cancellare non sia l'unica del sub che l'ha fatta. In caso contrario fa scattare un errore che blocca la cancellazione.

7.4 Trigger che verifica il numero di immersioni di un sub prima dell'aggiornamento

```
DELIMITER $$

CREATE TRIGGER verifica_imm1 BEFORE UPDATE ON ImmersioniSub

FOR EACH ROW

BEGIN

DECLARE quanti INT;

DECLARE ecco INT;

IF OLD.sub!=NEW.sub

THEN SELECT COUNT(*) INTO quanti FROM ImmersioniSub WHERE sub = OLD.sub;

IF quanti <2 THEN

SIGNAL SQLSTATE '10003'

SET MESSAGE_TEXT='Un sub risulterebbe non aver fatto nessuna immersione.';

END IF;

END IF;

END$$
```

```
DELIMITER ;
```

Questo trigger verifica il numero di immersioni di un sub prima dell'aggiornamento. Se un'immersione è stata fatta da un sub e si vuole cambiare il sub che l'ha fatta, prima viene verificato che l'immersione non sia l'unica fatta da quel sub. In caso contrario fa scattare un errore che blocca l'aggiornamento.

7.5 Trigger che verifica il prezzonoleggio di un'attrezzatura prima dell'inserimento

```
DELIMITER $$

CREATE TRIGGER verifica_pr_nol BEFORE INSERT ON AttrezzatureScuola

FOR EACH ROW

BEGIN

DECLARE p_n INT;

DECLARE t_a INT;

DECLARE ecco INT;

SELECT tipo INTO t_a FROM AttrezzatureScuola WHERE tipo = NEW.tipo GROUP BY tipo;

IF t_a IS NOT NULL THEN SELECT prezzonoleggio INTO p_n FROM AttrezzatureScuola WHERE tipo = NEW.tipo
GROUP BY prezzonoleggio;

IF NEW.prezzonoleggio <> p_n THEN

SIGNAL SQLSTATE '10004'

SET MESSAGE_TEXT='Il prezzonoleggio non corrisponderebbe al prezzonoleggio delle altre attrezzature dello

stesso tipo.';

END IF;

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica che il prezzonoleggio dell'attrezzatura che si vuole inserire corrisponda al prezzonoleggio delle attrezzature dello stesso tipo. In caso contrario fa scattare un errore che blocca l'inserimento.

7.6 Trigger che verifica il prezzonoleggio di un'attrezzatura prima dell'aggiornamento

```
DELIMITER $$

CREATE TRIGGER verifica_pr_nol1 BEFORE UPDATE ON AttrezzatureScuola

FOR EACH ROW

BEGIN

DECLARE p_n INT;
```

```
DECLARE t_a INT;

DECLARE ecco INT;

SELECT tipo INTO t_a FROM AttrezzatureScuola WHERE tipo = NEW.tipo GROUP BY tipo;

IF t_a IS NOT NULL THEN SELECT prezzonoleggio INTO p_n FROM AttrezzatureScuola WHERE tipo = NEW.tipo
GROUP BY prezzonoleggio;

IF NEW.prezzonoleggio <> p_n THEN

SIGNAL SQLSTATE '10005'

SET MESSAGE_TEXT='Il prezzonoleggio non corrisponderebbe al prezzonoleggio delle altre attrezzature dello

stesso tipo.';

END IF;

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica il nuovo prezzonoleggio dell'attrezzatura che si vuole modificare. Se si vuole cambiare il tipo dell'attrezzatura, il nuovo prezzonoleggio deve corrispondere al prezzonoleggio delle attrezzature con lo stesso tipo del nuovo tipo dell'attrezzatura da modificare. Se il tipo rimane invariato, anche il prezzonoleggio deve rimanere invariato. In caso contrario fa scattare un errore che blocca l'aggiornamento. Questo comporta che una volta inserito il prezzo per un tipo di attrezzatura non è più possibile modificarlo.

7.7 Trigger che verifica il prezzofinale di un'attrezzatura in noleggio prima dell'inserimento

```
DELIMITER $$

CREATE TRIGGER verifica_noleggi BEFORE INSERT ON Noleggi

FOR EACH ROW

BEGIN

DECLARE sconto_p INT;

DECLARE pr_no INT;

DECLARE pr_f INT;

DECLARE ecco INT;

SELECT scontoatt INTO sconto_p FROM Soci WHERE idsocio = NEW.sub;

SELECT prezzonoleggio INTO pr_no FROM AttrezzatureScuola WHERE codatt = NEW.att;

IF sconto_p IS NOT NULL THEN

SET pr_f = pr_no-(pr_no*sconto_p/100);
```

```
ELSE

SET pr_f = pr_no;

END IF;

IF NEW.prezzofinale <> pr_f THEN

SIGNAL SQLSTATE '10006'

SET MESSAGE_TEXT='Il prezzofinale dell\'attrezzatura noleggiata non e\' corretto.';

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica che il prezzofinale dell'attrezzatura a noleggio che si vuole inserire corrisponda al prezzonoleggio, eventualmente scontato se il sub che ha noleggiato quell'attrezzatura è anche socio. In caso contrario fa scattare un errore che blocca l'inserimento.

7.8 Trigger che verifica il prezzofinale di un'attrezzatura in noleggio prima dell'aggiornamento

```
DELIMITER $$

CREATE TRIGGER verifica_noleggi1 BEFORE UPDATE ON Noleggi

FOR EACH ROW

BEGIN

DECLARE sconto_p INT;

DECLARE pr_no INT;

DECLARE pr_f INT;

DECLARE ecco INT;

SELECT scontoatt INTO sconto_p FROM Soci WHERE idsocio = NEW.sub;

SELECT prezzonoleggio INTO pr_no FROM AttrezzatureScuola WHERE codatt = NEW.att;

IF sconto_p IS NOT NULL THEN

SET pr_f = pr_no-(pr_no*sconto_p/100);

ELSE

SET pr_f = pr_no;

END IF;

IF NEW.prezzofinale <> pr_f THEN

SIGNAL SQLSTATE '10007'
```

```
SET MESSAGE_TEXT='Il prezzofinale dell\'attrezzatura noleggiata non e\' corretto.';

END IF;

END$$

DELIMITER ;
```

Questo trigger verifica che il prezzofinale dell'attrezzatura a noleggio che si vuole modificare corrisponda al prezzonoleggio, eventualmente scontato se il sub che ha noleggiato quell'attrezzatura è anche socio. In caso contrario fa scattare un errore che blocca l'aggiornamento.

8. View

8.1 Immsubcomp

```
CREATE VIEW Immsubcomp (immersione, sub, costo) AS

select isu.immersione, isu.sub, i.costo

from ImmersioniSub isu join Immersioni i on isu.immersione=i.codimm;
```

8.2 Corsiperscomp

```
CREATE VIEW Corsiperscomp (persona, corso, costo) AS

select pc.persona, pc.corso, c.costo

from PersoneCorsi pc join Corsi c on pc.corso=c.codcorso;
```

Queste view permettono di reperire più facilmente la spesa di un sub per le sue immersioni e di una persona per i corsi che ha frequentato. Infatti vengono usate nella funzione Spesa_totale_annua e nella procedura spese_div.

9. Query

Di seguito vengono proposte alcune delle query più significative utilizzate nel progetto.

9.1 Persone che non sono né soci né sub

```
SELECT p.*

FROM (Persone p LEFT JOIN Soci s ON p.idpersona=s.idsocio) LEFT JOIN Sub su ON p.idpersona=su.idsub

WHERE s.idsocio IS NULL AND su.idsub IS NULL
```

Questa query restituisce i valori della tabella Persone i cui id non corrispondono a nessun idsub della tabella Sub e nessun idsocio della tabella Soci.

9.2 Attrezzature disponibili

```
SELECT a.tipo, COUNT(*) AS Quanti, a.prezzonoleggio

FROM AttrezzatureScuola a LEFT JOIN Noleggi n ON a.codatt = n.att

WHERE n.att IS NULL GROUP BY a.tipo, a.prezzonoleggio;
```

Questa query restituisce il tipo, la quantità e il prezzo delle attrezzature della scuola disponibili, cioè quelle che non sono presenti nella tabella Noleggi.

9.3 Query esempio

```
SELECT *

FROM Immersioni i

WHERE i.profonditamax<=20 AND i.luogo='Mare' AND i.tipo='Diurna' AND i.costo<=15.00
```

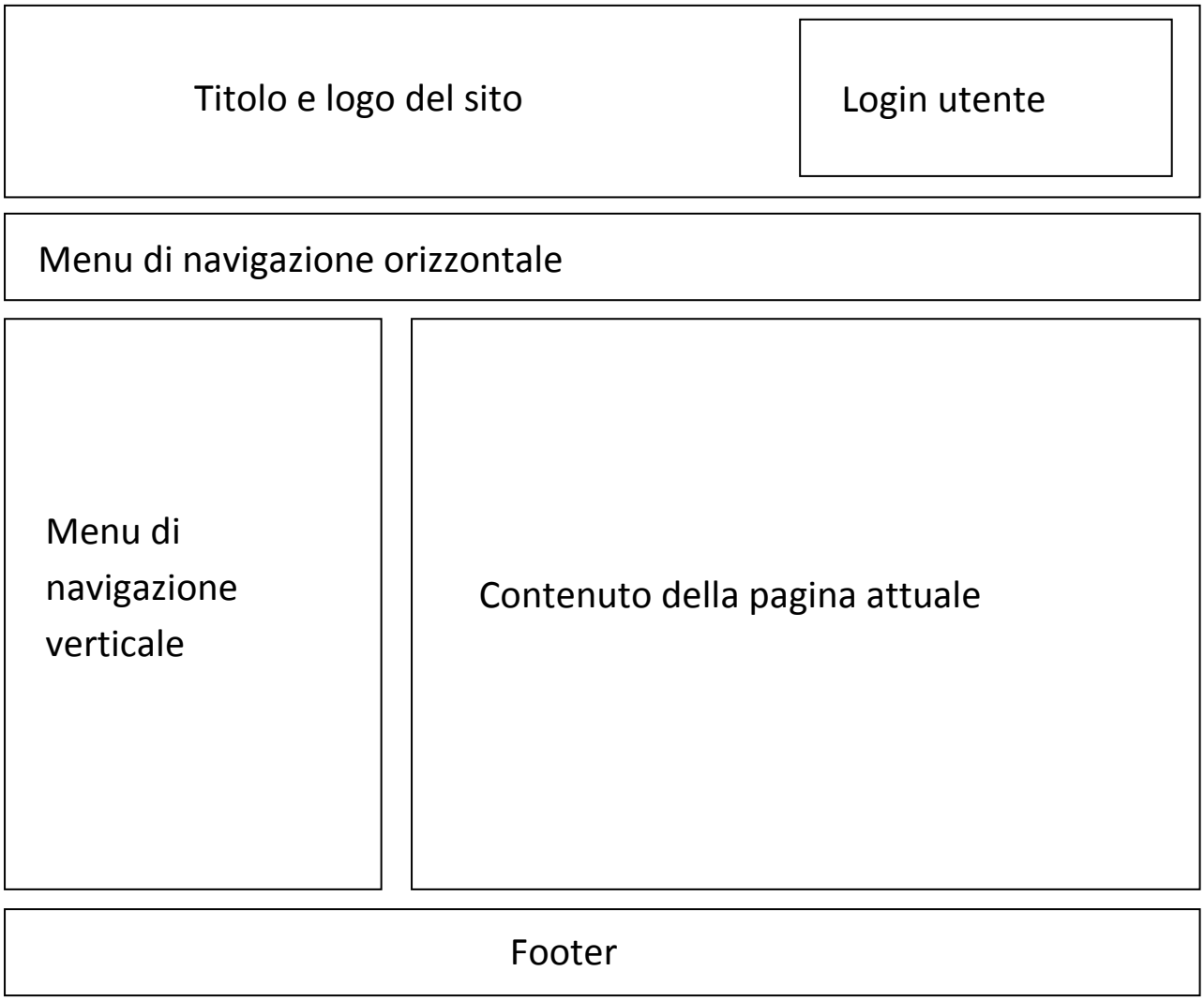
Questa query restituisce le immersioni con profondità non superiore a 20m, fatte al mare di giorno, con prezzo che non supera i 15 €.

Sono presenti anche altre query nel progetto ma sono componibili nell'interfaccia web a seconda di quello che si sta cercando per migliorare l'interazione con l'utente.

10. Interfaccia web

10.1 Organizzazione delle pagine

L'interfaccia web ha la seguente struttura:



In particolare:

- Il menu di navigazione orizzontale è variabile e contiene il percorso per arrivare alla pagina corrente. Ogni passo del percorso è un link alla pagina corrispondente escluso l'ultimo passo. Il primo passo è sempre la home che si riferisce alla pagina index.php.
- Il menu di navigazione verticale è fisso ed è composto da 8 campi e 7 o 8 link alle pagine corrispondenti a seconda che ci si trovi al primo livello del percorso o a uno dei successivi (o nella sezione registrazione o login) rispettivamente. Gli 8 campi sono:
  - Home: homepage dove il sito viene presentato
  - Corsi: elenca dei corsi offerti dalla scuola
  - Noleggi: elenca la quantità e il tipo di attrezzature disponibili
  - Eventi: elenca gli eventi organizzati dalla scuola
  - Immersioni: elenca le immersioni organizzate dalla scuola
  - Riunioni: elenca le riunioni dei soci
  - Ricerca avanzata: ricerca approfondita su attrezzature e immersioni
  - Sezione amministratore: inserimento, cancellazione e modifica di dati nel database, ricerca approfondita sulle persone
- Il contenuto della pagina attuale varia a seconda della sezione e della pagina in cui ci si trova.
- Il footer è fisso e contiene altre informazioni riguardo la scuola

10.2 Funzioni principali

10.2.1 barralogin.php

La funzione presente in barralogin.php gestisce il login di un utente. Si hanno due possibilità:

- Se l'utente non ha ancora effettuato il login, dà la possibilità di farlo o di registrarsi se non si dispone di un account per il sito.
- Se l'utente ha già effettuato l'accesso mostra il nome dell'account utilizzato per l'accesso, il pulsante di logout e il link all'area riservata.

10.2.2 con_db.php

La funzione presente in con_db.php gestisce la connessione con il database. Setta host, username e password ed si connette al server. Se la connessione con il server va a buon fine setta il nome del database da utilizzare. Infine restituisce la variabile $conn che contiene l'identificatore di collegamento MySQL.

10.2.3 corsi.php

La pagina corsi, oltre a presentare i corsi della scuola, dà la possibilità di vedere la lista delle lezioni del corso scelto dall'utente portandolo alla pagina lezioni.php.

10.2.4 cercaadvance.php

Accessibile dalla voce ricerca avanzata nel menu laterale. Offre una ricerca ad interazione con l'utente, effettuabile però solo se l'utente è autenticato.

10.2.5 sezioneamm.php

Accessibile dalla voce sezione amministratore nel menu laterale. Per questa sezione non basta che l'utente sia autenticato, ma deve essere autenticato come amministratore. Da questa pagina si ha accesso a inserimento, cancellazione e modifica dei dati del database, oltre alla ricerca sulle persone che frequentano la scuola sub, non permessa ad utenti comuni.

10.2.6 cancellazione dati dal database (vari file)

Prima di tutto all'utente viene chiesto di scegliere su quale tipo di tabella si vogliono eliminare dei dati. Poi viene chiesto di scegliere quali valori devono avere i dati da eliminare. Come ultimo passo viene mostrata la tabella con i dati che verranno eliminati. Infine, se la cancellazione va a buon fine, viene mostrata la tabella aggiornata.

10.2.7 inserimento dati nel database (vari file)

Anche in questa sezione come prima cosa all'utente viene chiesto di scegliere su quale tipo di tabella si vogliono inserire dei dati. Poi viene chiesto di scegliere i valori per il dato da inserire. Come ultimo passo viene fatto un riepilogo di quelli che saranno i valori del nuovo dato. Infine, se l'inserimento va a buon fine, viene mostrata la tabella aggiornata.

10.2.8 modifica dati del database (vari file)

Anche in questa sezione come prima cosa all'utente viene chiesto di scegliere su quale tipo di tabella si vogliono modificare dei dati. Poi viene chiesto di scegliere il campo da modificare e il nuovo valore. Vengono chiesti i valori dei campi a cui applicare la modifica. Come ultimo passo viene mostrata la tabella con i dati che verranno modificati e il campo che verrà modificato con il nuovo valore. Infine, se la modifica va a buon fine, viene mostrata la tabella aggiornata.

10.2.9 ricerca (vari file)

La ricerca è fatta in modo da interagire con l'utente tramite delle form. Vengono forniti anche esempi di quello che è possibile fare in modo da facilitare l'utente. I risultati della ricerca sono dati sotto forma di tabella.

10.3 Informazioni sull'autenticazione

L'autenticazione dell'utente e il mantenimento dello stato sono stati fatti tramite le sessioni.
Quando un utente effettua il login e questo va a buon fine, viene settata la variabile $_SESSION['username'] che contiene l'username dell'utente che ha effettuato il login. Sulle pagine che hanno delle restrizioni viene fatto un controllo esclusivamente su questa variabile per consentire o meno l'accesso ad esse.