Relazione Basi di Dati

Alessandro Filira - afilira
Matricola: 1023662

Padova, 20 Giugno 2014

Sommario

Abstract

Il progetto si pone di gestire una base di dati relativa l'amministrazione della segreteria di una società sportiva, in particolare di una società calcistica. Si proverà comunque a tenere il sistema il più generico possibile per poter essere sfruttato da più parti.
Le operazioni principali su tale base di dati sono la creazione di record anagrafici per gli atleti, allenatori, dirigenti e collaboratori con l'intera gestione di categorie e quote associative.
Maggiori informazioni riguardanti la strutturazione delle operazioni che si vogliono implementare sono presenti nella descrizione dei requisiti.

Descrizione dei Requisisti

Il progetto consiste nella creazione di una base di dati che permetta a più società sportive di gestire i propri atleti e responsabili attraverso un applicativo web.
Ogni società potrà vedere e gestire solo i giocatori della propria realtà in questo modo più società useranno la stessa base di dati ma potranno agire solamente su ciò che riguarda loro direttamente.

L'entità principale da amministrare sono le persone, in particolar modo gli atleti dei quali è importante sapere nome, cognome, codice fiscale, data di nascita, luogo di nascita, provincia di nascita, indirizzo di residenza codice fiscale e cellulare.
Ogni atleta può avere uno o due genitori dei quali si vuole conoscere Cognome e nome, codice fiscale, cellulare, telefono fisso, professione, email.
Per ogni tesserato si può indicare a chi intestare la fattura.

Per ogni atleta si vuole anche sapere quali documenti ha consegnato in modo da poter avviare la procedura di tesseramento: ad esempio per essere tesserati servono due fototessere, certificato medico, e altri moduli che potrebbero variare di anno in anno.
Perciò si cercherà di sviluppare una base dati che lasci piena libertà all'utente dell'applicativo web in modo che possa gestire totalmente anche questa fase.

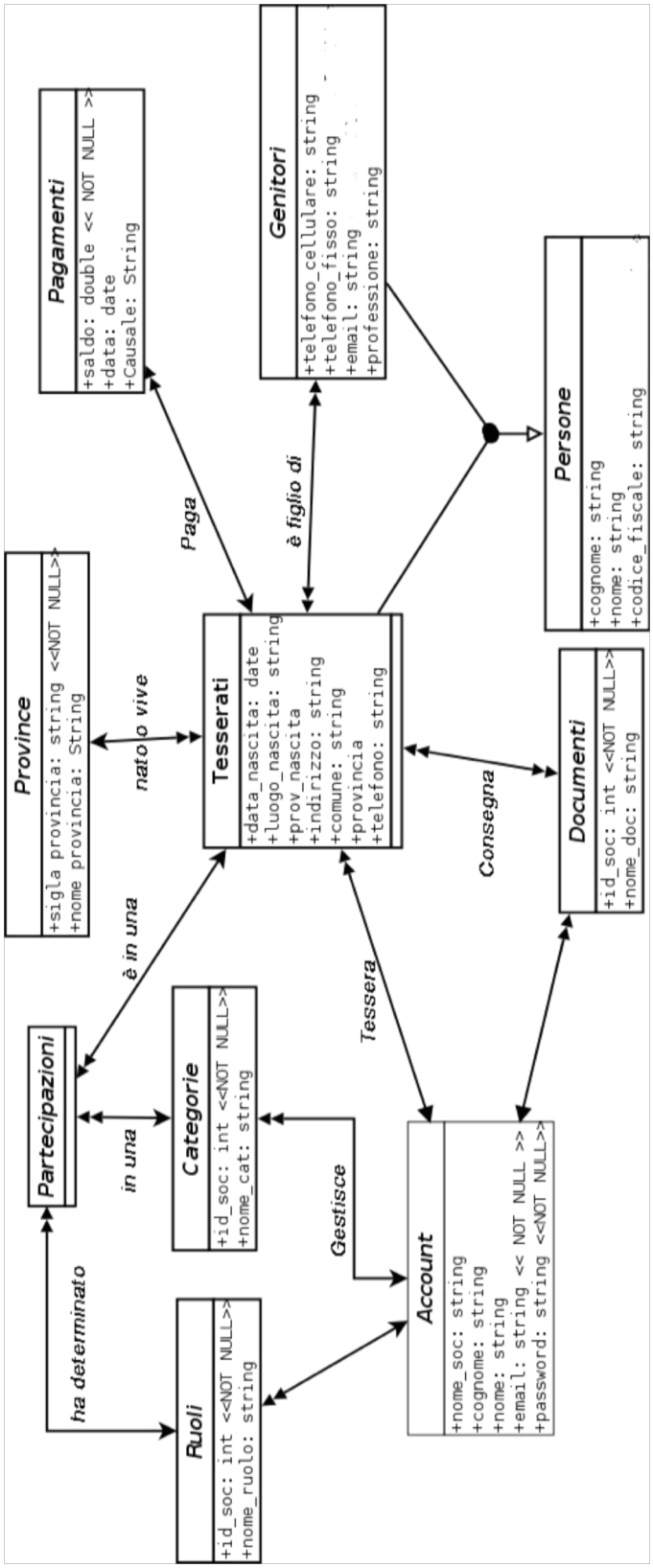
Ogni persona può essere inserita in una categoria nella quale avrà un ruolo che potrà essere allenatore, dirigente, atleta, medico, ecc. I vari ruoli saranno gestiti dall'amministratore in modo che in futuro se dovesse essere definito un nuovo ruolo, l'utente potrà essere facilmente inserirlo ed eliminare i ruoli non più necessari.

Direttamente collegato con i tesserati, c'è la gestione delle quote associative di cui si vuole sapere il costo e se sono state pagate o meno.
È anche prevista la possibilità di effettuare sconti per tesserati con che presentano problemi economici oppure eliminare la quota associativa per allenatori e dirigenti che svolgono un servizio all'interno della società.
C'è quindi la necessità di sapere se un tesserato ha pagato la quota per intero o solamente in parte e quanto è l'ammontare del dovuto.

Le operazioni tipiche sono la creazione, la modifica e la cancellazione delle varie entità, sarà inoltre implementata una ricerca tra i record degli atleti per velocizzare e rendere più immediato l'utilizzo dell'applicativo.

Progettazione concettuale

Schema Concettuale



Lista delle classi

Account

- Nome_società: *string*
- Cognome: *string*
- Nome: *string*
- Email: *string*
- Password: *string*

Persone

- Cognome: *string*
- Nome: *string*
- Codice_fiscale: *string*

- **Tesserati**
  - Data_nascita: *date*
  - Luogo_nascita: *string*
  - Provinica_nascita: *string*
  - Indirizzo_residenza: *string*
  - Comune: *string*
  - Provincia: *string*
  - Cellulare: *string*

- **Genitori**
  - Sesso: enum{M,F}
  - Telefono_cellulare: string
  - Telefono_fisso: string
  - Professione: string
  - Email: string

Categorie

- Nome_cat: string

Ruolo

- Nome_ruolo: string

Documenti

- Nome_doc: string

Provincie

- Siglia_provincia: string
- Nome provincia: string

Pagamenti

- Saldo: double
- Data pagamento: date
- Causale: string

Partecipazione (classe senza attributi)

Lista delle associazioni

Account-Tesserati

Ogni account ha 0 o molti tesserati.
Ogni tesserato appartiene a una sola società.
Molteplicità N : 1
Totalità: Totale verso tesserati, Parziale verso account

Tesserato-Provincia

Ogni tesserato è nato in una sola provincia.
Più tesserati possono essere nati nella stessa provincia.
Molteplicità N:1
Totalità: Totale verso provincia, Parziale verso tesserato

Tesserati-Genitori

Ogni tesserato ha 0, 1 o 2 genitori
Ogni genitore ha più figli
Molteplicità N:M
Totalità: Parziale verso genitori e verso tesserati poiché potrebbero esserci genitori senza figli e tesserati senza genitori.

Tesserato-Pagamenti

Ogni tesserato ha più pagamenti
Un pagamento fa riferimento ad un solo tesserato
Molteplicità N:1
Totalità: Totale verso pagamenti poiché ogni tesserato avrà almeno un pagamento, anche se questo sarà impostato a 0.

Account-Categorie, Account-Ruoli, Account-Documenti

Una società ha più categorie, ruoli e documenti.
Un ruolo, una categoria e un documento si riferisce ad una sola socità
Molteplicità N:1
Totalità: Totale su account perché una categoria, ruolo o documento fa sempre riferimento a un account mentre parziale su documenti poiché potrebbero esserci account senza documenti.

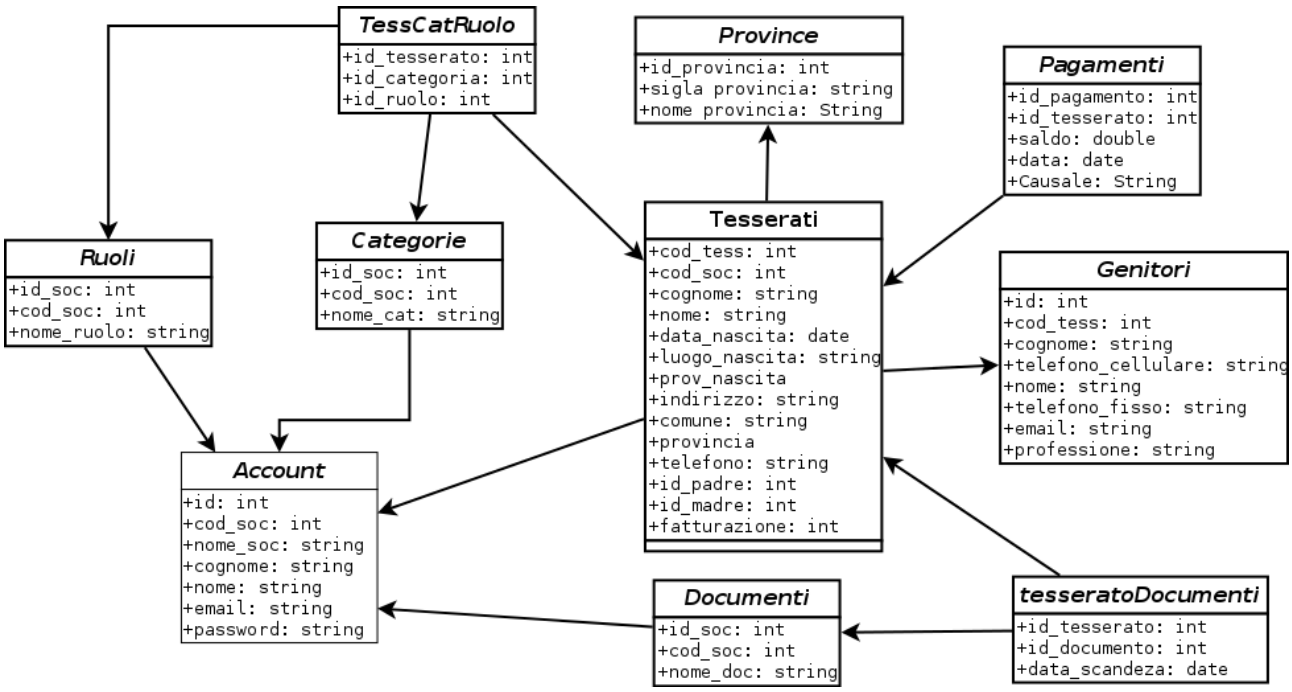
Tesserato-Documento

Un tesserato può consegnare più documenti
E più documenti fanno riferimento al medesimo tesserato.
Molteplicità N:M
Totalità: Totale verso documenti, Parziale verso tesserati
Attributi

- Data scadenza documento

Progettazione logica

Schema Logico (relazioni e campi)



Gerarchie

La sola gerarchia presente è nella classe delle Persone ed è una gerarchia con partizionamento verticale. Visto il grande numero di attributi diversi si è fatta la scelta di suddividere la classe Persone in due classi: Genitori e Tesserati avendo così le seguenti tabelle:

Tesserati

- Cod_tess int <<PK>>(Auto Increment)
- Cod_società: int <<FK(account)>>
- Cognome: string
- Nome: string
- Data_nascita: date <<NOT NULL>>
- Luogo Nascita:string
- Prov_nascita: <<FK (provincie)>>
- Codice_fiscale :string <<NOT NULL>>
- Indirizzo: string
- Comune: string
- Provincia : int <<FK (provincie)>>
- Telefono: string
- Id_padre : int <<FK (genitori)>>
- Id_madre : int <<FK (genitori)>>
- fatturazione : int <<FK (genitori)>>

Genitori

Per semplificare la gestione dei genitori è stato introdotto anche qui il cod_soc in modo da avere in maniera più immediata i genitori di una certa società.
Ciò è stato fatto anche per evitare che ci fossero genitori senza figli che rimanessero nel database senza poter essere ri-associati.

- id: int <<PK >>(Auto Increment)
- cod_soc <<FK (account) >>
- Cognome: string
- Nome:string
- Sesso: enum {M,F}
- Codice Fiscale: string
- Telefono_cellulare: string
- Telefono_fisso :string
- Professione: string
- Email: string

Account

- Id: int <<PK>> (Auto Increment)
- Cod_società: int
- Nome società: string
- Cognome: string
- Nome: string
- Email: string
- Password: string

Categorie

- Id_cat: int <<PK>>(Auto Increment)
- Cod_soc: int <<FK (account)>>
- Nome categoria: string

Ruoli

- Id_ruolo: int <<PK>>(Auto Increment)
- Cod_soc: int <<FK (account)>>
- Nome_ruolo: string

Documenti

- Id_doc: int <<PK>>(Auto Increment)
- Cod_soc: int <<FK (account)>>
- Nom_doc: string

Pagamenti

- Id_pagamento: int <<PK>>(Auto increment)
- Id_tesserato: int <<FK (tesserati)>>
- Saldo: string

- Data_pagamento:date
- Causale: string

Provincie

- Id_provicia: int <<PK>> (Auto increment)
- Sigla_provincia: string
- Nome_provincia.string

Chiavi sintetiche

Per comodità sono state chiavi sintetiche in ogni tabella in particolare nelle persone, tesserati e genitori in cui si poteva semplicemente usare il codice fiscale, essendo identificativo della persona.
È stato inoltre deciso che il codice fiscale non è chiave unica in quanto un genitore potrebbe essere in due società diverse poiché ha figli iscritte in società diverse. Ragionamento analogo è stato fatto per i giocatori in quanto potrebbe succedere che una società non elimini il proprio giocatore quando egli non è più iscritto, rimanendo così nel database. Si sarebbe potuto comunque mettere come chiave cod_soc e codice_fiscale ma per comodità si è preferito usare una id con numero progressivo.

Associazioni

TessCatRuolo

- Id_tesserato: int <<PK>> <<FK (tesserati) >>
- Id_categoria: int <<PK>> <<FK (categorie)>>
- Id_ruolo: int <<PK>> <<FK (ruoli)>>

TesseratoDocumenti

- Id_tesserato: int <<PK>><<FK (tesserati) >>
- Id_documento: int <<PK>><<FK (documenti) >>
- Data_scandeza: date

Implementazione della base di dati

Il codice sotto riportato crea la base di dati descritta in precedenza.

```
DROP TABLE IF EXISTS account;
DROP TABLE IF EXISTS categorie;
DROP TABLE IF EXISTS documenti;
DROP TABLE IF EXISTS genitori;
DROP TABLE IF EXISTS pagamenti;
DROP TABLE IF EXISTS province;
DROP TABLE IF EXISTS ruoli;
DROP TABLE IF EXISTS TessCatRuolo;
DROP TABLE IF EXISTS tesserati;
DROP TABLE IF EXISTS tesseratoDocumenti;
```

```
CREATE TABLE `account` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cod_soc` int(11) NOT NULL,
  `nome_soc` varchar(100) NOT NULL,
  `cognome` varchar(100) NOT NULL,
  `nome` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;


CREATE TABLE `categorie` (
  `id_cat` int(11) NOT NULL AUTO_INCREMENT,
  `cod_soc` int(11) NOT NULL,
  `nome_cat` varchar(100) NOT NULL,
  PRIMARY KEY (`id_cat`),
  KEY `nome_cat` (`nome_cat`),
  KEY `cod_soc` (`cod_soc`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=22 ;


CREATE TABLE `documenti` (
  `id_doc` int(11) NOT NULL AUTO_INCREMENT,
  `cod_soc` int(11) NOT NULL,
  `nome_doc` varchar(100) NOT NULL,
  PRIMARY KEY (`id_doc`),
  KEY `cod_soc` (`cod_soc`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;


CREATE TABLE `genitori` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cod_soc` int(11) NOT NULL,
  `cognome` varchar(100) NOT NULL,
  `nome` varchar(100) NOT NULL,
  `sesso` enum('M','F') NOT NULL,
  `codice_fiscale` varchar(100) NOT NULL,
  `telefono_cellulare` varchar(100) NOT NULL,
  `telefono_fisso` varchar(100) NOT NULL,
  `professione` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `cod_soc` (`cod_soc`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=258 ;


CREATE TABLE `pagamenti` (
  `id_pagamento` int(11) NOT NULL AUTO_INCREMENT,
  `id_tesserato` int(11) NOT NULL,
  `saldo` double NOT NULL,
  `data_pagamento` date NOT NULL,
```

```
  `causale` varchar(100) NOT NULL,
  PRIMARY KEY (`id_pagamento`),
  KEY `id_tesserato` (`id_tesserato`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1467 ;


CREATE TABLE `province` (
  `id_province` int(16) NOT NULL AUTO_INCREMENT,
  `sigla_provincia` varchar(5) NOT NULL,
  `nome_provincia` varchar(128) NOT NULL,
  PRIMARY KEY (`id_province`),
  KEY `id_province` (`id_province`),
  KEY `sigla_province` (`sigla_provincia`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=109 ;


CREATE TABLE `ruoli` (
  `id_ruolo` int(11) NOT NULL AUTO_INCREMENT,
  `cod_soc` int(11) NOT NULL,
  `nome_ruolo` varchar(100) NOT NULL,
  PRIMARY KEY (`id_ruolo`),
  KEY `cod_soc` (`cod_soc`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=13 ;


CREATE TABLE `TessCatRuolo` (
  `id_tesserato` int(11) NOT NULL,
  `id_categoria` int(11) NOT NULL,
  `id_ruolo` int(11) NOT NULL,
  PRIMARY KEY (`id_tesserato`,`id_categoria`,`id_ruolo`),
  KEY `id_tesserato` (`id_tesserato`),
  KEY `id_categoria` (`id_categoria`),
  KEY `id_ruolo` (`id_ruolo`),
  KEY `id_ruolo_2` (`id_ruolo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE `tesserati` (
  `cod_tess` int(11) NOT NULL AUTO_INCREMENT,
  `cod_soc` int(11) NOT NULL,
  `cognome` varchar(100) NOT NULL,
  `nome` varchar(100) NOT NULL,
  `data_nascita` date NOT NULL,
  `luogonascita` varchar(100) NOT NULL,
  `prov_nascita` int(11) DEFAULT NULL,
  `codice_fiscale` varchar(100) NOT NULL,
  `indirizzo` varchar(100) NOT NULL,
  `comune` varchar(100) NOT NULL,
  `provincia` int(100) DEFAULT NULL,
  `telefono` varchar(100) DEFAULT NULL,
  `id_padre` int(11) DEFAULT NULL,
  `id_madre` int(11) DEFAULT NULL,
  `fatturazione` int(11) DEFAULT NULL,
  PRIMARY KEY (`cod_tess`),
  KEY `cod_soc` (`cod_soc`),
```

```
  KEY `id_padre` (`id_padre`),
  KEY `id_madre` (`id_madre`),
  KEY `telefono` (`telefono`),
  KEY `fatturazione` (`fatturazione`),
  KEY `provincia` (`provincia`),
  KEY `prov_nascita` (`prov_nascita`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1276 ;


CREATE TABLE `tesseratoDocumenti` (
  `id_tesserato` int(11) NOT NULL,
  `id_documento` int(11) NOT NULL,
  `data_scadenza` date DEFAULT NULL,
  PRIMARY KEY (`id_tesserato`,`id_documento`),
  KEY `id_documento` (`id_documento`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



ALTER TABLE `categorie`
  ADD CONSTRAINT `category` FOREIGN KEY (`cod_soc`) REFERENCES `account` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `documenti`
  ADD CONSTRAINT `doc` FOREIGN KEY (`cod_soc`) REFERENCES `account` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `genitori`
  ADD CONSTRAINT `genitoriAcc` FOREIGN KEY (`cod_soc`) REFERENCES `account` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `pagamenti`
  ADD CONSTRAINT `pagamenti` FOREIGN KEY (`id_tesserato`) REFERENCES `tesserati`
(`cod_tess`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `ruoli`
  ADD CONSTRAINT `ruoli` FOREIGN KEY (`cod_soc`) REFERENCES `account` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `TessCatRuolo`
  ADD CONSTRAINT `categoriaPlayer` FOREIGN KEY (`id_categoria`) REFERENCES `categorie`
(`id_cat`),
  ADD CONSTRAINT `id_ruolo` FOREIGN KEY (`id_ruolo`) REFERENCES `ruoli` (`id_ruolo`),
  ADD CONSTRAINT `id_tesserato` FOREIGN KEY (`id_tesserato`) REFERENCES `tesserati`
(`cod_tess`);


ALTER TABLE `tesserati`
  ADD CONSTRAINT `fatt` FOREIGN KEY (`fatturazione`) REFERENCES `genitori` (`id`),
```

```
  ADD CONSTRAINT `madre` FOREIGN KEY (`id_madre`) REFERENCES `genitori` (`id`) ON
DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `padre` FOREIGN KEY (`id_padre`) REFERENCES `genitori` (`id`) ON
DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `prov_casa` FOREIGN KEY (`provincia`) REFERENCES `province`
(`id_province`),
  ADD CONSTRAINT `prov_nascita` FOREIGN KEY (`prov_nascita`) REFERENCES `province`
(`id_province`),
  ADD CONSTRAINT `tesserato` FOREIGN KEY (`cod_soc`) REFERENCES `account` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `tesseratoDocumenti`
  ADD CONSTRAINT `doc_tess` FOREIGN KEY (`id_documento`) REFERENCES `documenti`
(`id_doc`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `tesseratoDoc` FOREIGN KEY (`id_tesserato`) REFERENCES `tesserati`
(`cod_tess`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Query

Lista Atleti

La prima query permette di avere una lista con tutti i giocatori della società con cui si fa il login. Vengono visualizzati in una tabella un checkbox per eliminare il giocatore, un numero contatore, cognome, nome, data di nascita, indirizzo, comune e provincia, telefono e un link che manda alla pagina della scheda atleta.
Oltre alla semplice lista ,si ha la possibilità di ordinare i record tramite i link a capo delle colonne Cognome, Nome, Data di nascita. Viene passato in GET un parametro che viene letto e tramite script php e attraverso alcune istruzioni condizionali genera la form.
Dalla sessione viene letto il cod_soc che fa riferimento alla società con cui si fa il login.

Query Base:
“SELECT p.sigla_provincia, t.cod_tess,t. cognome ,t.nome ,t.data_nascita, t.indirizzo, t.comune, t.telefono
FROM tesserati t
LEFT JOIN province p **ON** p.id_province=t.provincia
WHERE t.cod_soc=".$_SESSION['user']['cod_soc'];

Per quanto riguarda l’ordinamento vengono aggiunte le seguenti righe a seconda della scelta fatta

ORDER BY t.cognome **ASC I DESC**
ORDER BY t.nome **ASC I DESC**
ORDER BY t.data_nascita **ASC I DESC**

Nella stessa pagina è possibile fare una ricerca per cognome, nome, anno di nascita e provincia.

Per fare ciò vi è un menù a tendina per selezionare il campo di ricerca e un campo di input in cui digitare il testo.
Le Query per la ricerca sono non fanno altro che aggiungere una condizione alla query di partenza, $val è il valore passato dal form di ricerca.

```
Nome: “AND t.nome LIKE "'.$val."%"”
Cognome: “AND t.cognome LIKE "'.$val."%"”
Anno di Nascita: " AND t.data_nascita BETWEEN "'.$val."-01-01' AND "'.$val."-12-31'"
Provincia: " AND p.sigla_provincia LIKE "'.$val."'
```

È inoltre possibile eliminare un tesserato o più tesserati dal database e per fare ciò si fa utilizzo di una semplice query e un ciclo foreach
La query per eliminare il tesserato con id=$id è:

```
‘DELETE FROM tesserati WHERE cod_tess='.$id;
```

Gestione Pagamenti

Altre query sono state utilizzate per vedere e verificare gli stati di pagamento delle quote associative.
È stata realizzata una tabella con le persone che devono ancora versare l’intera quota in modo che chi utilizza il sistema possa vedere nell’immediato i debitori

```
SELECT t.cod_tess,t.cognome,t.nome,t.data_nascita,t.codice_fiscale,t.telefono, SUM(p.saldo) as totale
FROM tesserati t
JOIN pagamenti p ON t.cod_tess= p.id_tesserato
WHERE t.cod_soc='.$_SESSION['user']['cod_soc'].'
GROUP BY p.id_tesserato
HAVING (totale)<0
```

Altra query utilizzata per gestire i pagamenti di ogni singola persona per avere lo storico di ogni pagamento è:

```
SELECT t.nome,t.cognome,t.codice_fiscale,p.causale,p.data_pagamento,p.saldo
FROM tesserati t
JOIN pagamenti p ON t.cod_tess= p.id_tesserato
WHERE t.cod_soc='.$_SESSION['user']['cod_soc'].'
AND p.id_tesserato='.$_GET['cod_tes'].'
ORDER BY p.data_pagamento ASC
```

È inoltre possibile aggiungere un pagamento tramite un form inserendo data di pagamento, importo e causale che andrà ad eseguire la seguente query:

```
INSERT INTO pagamenti
VALUES(
        NULL,"
        '.$_GET['cod_tes'].'",
        "'.$_POST['saldo'].'",
        "'.$_POST['data_pagamento'].'",
        "'.$_POST['causale'].'"
)
```

Gestione Ruoli, Gestione Categorie

Per la gestione dei ruoli è stata creata una tabella con identificativo ruolo, identificativo della società che lo crea e nome del ruolo.
È stata data la possibilità di modificare e aggiungere ruoli a proprio piacimento con le seguenti query che leggono da POST e dalla SESSION i valori necessari
Inserimento:

```
INSERT INTO ruoli
VALUES(
"NULL",
"'.$_SESSION['user']['cod_soc'].'",
"'.$_POST['role_name'].'")
```

e modifica:

```
UPDATE ruoli
SET nome_ruolo="'.$_POST['role_name'].'"
WHERE id_ruolo='.$_GET['id']
```

Inoltre per ogni ruolo è possibile visualizzare una tabella con la lista dei tesserati che ricoprono tale ruolo, indicando anche la relativa categoria.

```
SELECT r.nome_ruolo, t.cod_tess, t.cognome, t.nome, t.data_nascita, t.indirizzo, t.comune,
pr.sigla_provincia, t.telefono, c.nome_cat
FROM TessCatRuolo p
JOIN tesserati t ON t.cod_tess= p.id_tesserato
JOIN categorie c ON c.id_cat=p.id_categoria
JOIN ruoli r ON p.id_ruolo= r.id_ruolo
LEFT JOIN province pr ON pr.id_province=t.prov_nascita
WHERE p.id_ruolo='.$_GET['id'].'
AND  t.cod_soc='.$_SESSION['user']['cod_soc'];
```

Vi è inoltre una pagina con tutti i ruoli in cui viene segnato anche il numero di componente per ogni singolo ruolo
Se una persona copre più ruoli verrà conteggiata in entrambi i ruoli.

```
SELECT c.id_ruolo,c.nome_ruolo,count(t.id_ruolo) as totale
FROM ruoli c
LEFT JOIN TessCatRuolo t ON t.id_ruolo=c.id_ruolo
WHERE c.cod_soc='.$_SESSION['user']['cod_soc'].'
GROUP BY c.id_ruolo';
```

Per quanto riguarda le categorie, le query sono analoghe cambiando solo la tabella ruoli con categorie.

Gestione HomePage

In home page sono state fatte delle query che riassumono e forniscono alcune informazioni sulla società gestita.
La query per avere il numero di atleti è stata implementata con una funzione, vedi paragrafo funzioni.

In seguito è stata eseguita una query per conoscere i tesserati cui scade, in mneo di 30 giorni, un documento e quale documento

```
SELECT t.cognome, t.nome, d.data_scadenza,doc.nome_doc
FROM tesseratoDocumenti d
JOIN tesserati t ON d.id_tesserato=t.cod_tess
JOIN documenti doc ON d.id_documento=doc.id_doc
WHERE t.cod_soc='.$_SESSION['user']['cod_soc'].'
HAVING (d.data_scadenza-CURDATE())<30 AND (d.data_scadenza-CURDATE())>0
```

L'ultima sezione e query della homepage è relativa ai compleanni ed è così costituita

```
SELECT t.cognome, t.nome, YEAR(CURDATE())-YEAR(t.data_nascita) AS anni
FROM tesserati t
WHERE  MONTH(t.data_nascita)='.$oggi_mese.'
AND DAYOFMONTH(t.data_nascita)='.$oggi_giorno.'
AND t.cod_soc='.$_SESSION['user']['cod_soc'];
```

Funzioni

Numero di tesserati

Per avere sempre a portata di mano il numero di tesserati in una determinata società è stata implementata una funzione che ricevuto come parametro l'id della società restituisce il numero di tesserati di tale società.

```
DELIMITER $$

CREATE FUNCTION `contaTesserati`(`id` INT) RETURNS int(11)
BEGIN
DECLARE numTess INT;
SELECT count(t.cod_tess) INTO numTess FROM tesserati t WHERE t.cod_soc=id;
RETURN numTess;

END$$
DELIMITER ;
```

Totale quote

La funzione totale quote torna il totale delle quote pagate dai tesserati di una stessa società.
Si riporta il codice che la genera

```
DELIMITER $$

CREATE FUNCTION `totale_quote`(`cod_soc` INT) RETURNS int(11)
BEGIN
DECLARE totQuote FLOAT;
SELECT SUM(p.saldo) as tot INTO totQuote
FROM pagamenti p
JOIN tesserati t ON t.cod_tess=p.id_tesserato
WHERE t.cod_soc=cod_soc AND p.saldo>0;
RETURN totQuote;
END$$

DELIMITER;
```

Trigger

Gestione Pagamenti

Per gestire al meglio i pagamenti ed evitare che l'utente commetta errori nell'inserimento dei record è stato implementato un trigger che modifica il saldo e la causale qualora il tesserato abbia già finito di pagare la quota generata.
È però possibile generare un debito inserendo un valore negativo in modo da poter accettare i pagamenti successivi fino al pareggio dalle somma.

```
DROP TRIGGER IF EXISTS `pagamenti`;
DELIMITER //
CREATE TRIGGER `pagamenti` BEFORE INSERT ON `pagamenti`
 FOR EACH ROW BEGIN
DECLARE sumPagamenti FLOAT;
SELECT SUM(p.saldo) as tot INTO sumPagamenti FROM pagamenti p WHERE
p.id_tesserato=new.id_tesserato;
IF (sumPagamenti=0 && new.saldo>0) THEN
SET new.saldo=0;
SET new.causale="Gia Saldato";
END IF;
END
//
DELIMITER ;
```

Eliminazione Genitori

Per evitare che un tesserato rimanga "orfano" e che quindi non abbia nè un padre né una madre nel sistema si è deciso di implementare un trigger che evitasse l'eliminazione di un genitore se questi era padre o madre di qualche tesserato.

```
DROP TRIGGER IF EXISTS `deleteGenitori`;
DELIMITER //
CREATE TRIGGER `deleteGenitori` BEFORE DELETE ON `genitori`
 FOR EACH ROW BEGIN
DECLARE NumeroFigli INT;
SELECT count(t.cod_tess) as figli INTO NumeroFigli FROM genitori g LEFT JOIN tesserati t ON
t.id_padre=g.id ll t.id_madre=g.id WHERE g.id=old.id;
IF (NumeroFigli>0) THEN
CALL ERROR;
END IF;
END
//
DELIMITER ;
```

Interfaccia web

Pagine sviluppate

Config.php
Il file config.php contiene semplicemente uno script php per connettersi al database, il file è strutturato in modo che in cambio di database, password o utente può essere facilmente modificabile.
Tale file verrà richiamato all'inizio di ogni pagina.

content/function.php
Il file function contiene alcune funzioni utili per l'applicazione. Anche questo file come config.php viene incluso in ogni pagina in modo da avere sempre a disposizione le funzioni implementate, in particolare la funzione per generare le form da un array passato alla funzione.

content/home.php
La pagina home.php è la vera e propria home page del sito una volta effettuato il login. Tramite uno script presente nel file header.php si può accedere a tale pagina solo una volta effettuato il login.

header.php
Il file header contiene un controllo sulla sessione per capire se l'utente è loggato oppure no, se così non fosse viene effettuato un redirect alla pagina login.php.
Oltre a questo controllo vengono inclusi i css, eventuali file javascript e visualizzato il menù di ogni pagina.

footer.php
Il footer ha il compito di chiudere la connessione al database e far visualizzare il piè pagina in html.

document.php
Questo file contiene la query che per ogni documento della società visualizza il numero di elementi, ovvero i tesserati che lo hanno consegnato, e due link: uno per poter visualizzare la lista delle persone che lo hanno consegnato e uno per poter modificare il nome del documento.
Inoltre da questa pagina è possibile eliminare uno o più documenti.

edit_document.php
Pagina che permette la modifica del nome del documento selezionato il cui id viene passato come parametro in GET.
Se non viene passato alcun parametro allora si potrà creare un nuovo documento.

edit_player.php
Tale pagina rappresenta la scheda del singolo tesserato nella qualche, oltre a visualizzare tutti i dati, si ha la possibilità di modificarli e aggiornarli.
I dati del tesserato sono i proprio dati anagrafici e quelli dei genitori.

edit_role.php
Modifica del ruolo il cui id è prelevato dal GET, si può modificare il campo nome di tale ruolo.
Se non vi è alcun id nel GET allora la pagina si comporterà come se si dovesse creare un nuovo ruolo.

edit_team.php
Procedura analoga a quella dei documenti e dei ruoli in cui la modifica è limitata al nome della categoria, se $_GET['id'] ha valore altrimenti crea nuova categoria.

genitori.php
Questa pagina contiene la lista dei genitori inseriti nella società sportiva con cui si è fatti il login. Viene visualizzato il il cognome, il nome, il sesso, il telefono, l'indirizzo mail e il numero di figli che ha all'interno della società.
Il numero di figli è utile per capire se si può eliminare o meno tale genitore, come spiegato nella sezione dedicata ai trigger.

index.php
Questa pagina viene utilizzata per la procedura di logout e per dirottare l'utente alla pagina di login se esso non è presente in sessione oppure se è già presente una sessione alla pagina home.php

login.php
La pagina login.php è la prima pagina che si visualizza e serve per accreditarsi alla piattaforma.
È presente una form che passa in POST i valori di username e password dell'utente che tramite la funzione verify_login contenuta nel file funcion.php verifica la correttezza delle credenziali.

mod_pagamenti.php
Pagina che presenta tutto lo storico pagamenti del singolo atleta visualizzandolo in una tabella con data pagamento, importo e causale.
È inoltre presente una somma parziale e una form che permette di generare un altro pagamento.

new_player.php
Scheda personale del singolo tesserato che permette l'inserimento dei dati anagrafici, della categoria, del ruolo, dell'importo della quota e dei dati anagrafici dei genitori.

pagamenti.php
Lista di tutti i tesserati che hanno un debito verso la società.
Tale lista viene presentata tramite una tabella con un link che manda alla pagina dei pagamenti del singolo atleta.
È inoltre possibile filtrare i tesserati per "Chi ha pagato", "chi non ha pagato" e "tutti".

player_doc.php
È la pagina che permette di visualizzare la lista completa dei tesserati che hanno consegnato il documento con id passato come parametro in GET nella pagina.
La lista presenta alcuni dati anagrafici del tesserato, la data di scadenza del documento (se prevista) e un link per andare alla scheda personale del tesserato.

playerrole.php
Lista dei tesserati del ruolo selezionato il cui id è presente in $_GET['id'].
La lista oltre ad alcuni dati anagrafici del tesserato indica la categoria in cui quel tesserato ricopre il ruolo selezionato.

players.php
Lista completa di tutti i tesserati della società senza distinzione di ruoli, categorie, documenti. I vari record sono ordinabili per cognome, nome e data di nascita sia in ordine crescente che decrescente.
È anche presente un cerca per poter trovare con maggiore facilità il tesserato di cui si vogliono le informazioni.
Da questa schermata è possibile eliminare uno o più tesserati dal database della società.

playerteam.php
Analogamente alla pagina playerrole.php e player_doc.php si presenta la lista dei tesserati appartenenti alla categoria selezionata tramite l'id nel GET e il ruolo che i vari tesserati ricoprono nella categoria.

roles.php
Lista dei ruoli con possibilità di eliminarli e modificarli.

teams.php
Lista delle cateria con possibilità di eliminarle e modificarle.

css/style.css
Foglio di stile che contiene le regole css per la vista dell'applicazione.

js/jquery-1.11.1.min.js
Libreria jquery per poter implementare alcune funzioni comode per l'utente.

js/utility.js
Script che contiene le funzioni jquery.

Note conclusive

Per inserire i record nella base dati si è prima sviluppata l'interfaccia utente che permettesse l'inserimento dei tesserati dall'applicativo.
In seguito è stato usato un *tool* in rete che, una volta ricreata la base dati, con le tipologie di attributi, generava in automatico un centinaio di record che rispettassero le regole impostate.
Con questo strumento e qualche sistemazione manuale si è riusciti a popolare il database in maniera rapida e non troppo monotona.
Alcuni dati inseriti sono errati a causa della generazione dei record; come ad esempio codici fiscali, numeri di telefono e province. Ma la quantità di record inseriti credo sia sufficiente per poter eseguire query valide e avere una buona risposta dal sistema.