PROGETTO BASI DI DATI

WHITE STALLION

Boseggia Marco – Matricola: 1044608

Chinellato Maria Giovanna – Matricola: 1049511

ABSTRACT

Il progetto ha lo scopo di fornire un database e un interfaccia web che permettano la gestione di uno o più centri ippici da parte di una azienda. In particolare, il database raccoglie informazioni riguardanti i dipendenti, gli utenti, i cavalli e i concorsi ai quali cavalli e cavalieri dei maneggi partecipano ed hanno partecipato. L'interfaccia web, invece, permette agli utenti di registrarsi per ottenere informazioni generali sui centri ippici, come la loro ubicazione, e per fare richiesta di iscrizione alle lezioni di equitazione. Gli utenti registrati ed iscritti possono, inoltre, consultare gli orari delle loro lezioni, fare richiesta per cambiare orario di lezione e, se utenti agonistici, iscriversi ai concorsi. Gli istruttori sono anch'essi registrati e possono consultare il “calendario” delle loro lezioni. Infine, gli amministratori possono accedere alle informazioni relative a istruttori e utenti, approvare o rifiutare le richieste ricevute, aggiungere (in caso di acquisto) o rimuovere (in caso di vendita o decesso) i cavalli e assumere o licenziare i dipendenti. Le operazioni tipiche sono la creazione o la modifica dei dati relativi alle persone e ai cavalli e l'aggiunta di concorsi ai quali i cavalieri possono iscriversi.

ANALISI DEI REQUISITI

Per la realizzazione della base di dati e della relativa interfaccia web sono state concepite le seguenti entità principali: centri ippici, utenti registrati e cavalli. Più precisamente di ogni centro ippico ci interessa conoscere:

* l’indirizzo e-mail, che lo identifica univocamente;
* il nome;
* la città in cui ha sede;
* l’indirizzo postale;
* il numero di telefono;

Gli utenti che la base di dati gestisce sono esclusivamente quelli registrati e si dividono in dipendenti ed allievi. La classe dipendenti si ramifica ulteriormente in amministratori, istruttori e stallieri, mentre la classe allievi comprende gli allievi agonistici, gli allievi in mezza fida e gli allievi senza fida. Di ogni utente registrato ci interessa:

* il codice fiscale, che lo identifica univocamente;
* il nome e il cognome;
* la data di nascita, l'età dell'utente deve essere superiore ai sette anni;
* la città di residenza;
* il CAP della città di residenza;
* l’indirizzo postale;
* l'indirizzo e-mail;
* il numero di telefono;
* la password per l’accesso all’interfaccia web.

Più precisamente di ogni dipendente vogliamo memorizzare:

* il centro ippico in cui lavora;
* lo stipendio.

Le informazioni che vogliamo ricordare di ogni allievo sono:

* il centro ippico a cui è iscritto;
* l’istruttore a cui è associato;
* l’orario delle lezioni.

Gli allievi agonistici hanno un abbonamento “fida totale”, questo abbonamento implica il legame biunivoco tra cavallo e allievo, mentre gli allievi mezza fida possono prendersi cura di un solo cavallo ed il cavallo è condiviso da al massimo due allievi. Di questi allievi, quindi, ci interessa memorizzare il cavallo a cui sono associati.

Di ogni cavallo ci interessa:

* il codice del passaporto, che lo identifica univocamente;
* il nome;
* la razza;
* l’anno di nascita;
* le specialità in gara;
* il centro ippico a cui appartiene;

Le altre entità che compongono la base di dati sono: concorsi, richieste, città e assicurazioni.

I concorsi si suddividono in concorsi interni, organizzati dai centri ippici gestiti dal database, ed esterni, organizzati da altri centri ippici.

Le informazioni relative ai concorsi sono:

* la descrizione della tipologia del concorso;
* la data di inizio concorso;
* la data di fine concorso;
* il primo premio.

Dei concorsi interni ci interessa:

* il centro ippico che li organizza.

Dei concorsi esterni vogliamo sapere:

* l’indirizzo postale del centro ippico in cui a luogo il concorso;
* il nome del centro ippico organizzatore;
* la città;
* il numero di telefono del centro ippico;
* l’indirizzo e-mail del centro ippico.

Ad un concorso possono iscriversi solamente gli allievi agonistici.

Di ogni richiesta vogliamo memorizzare:

* il tipo di richiesta (ad esempio iscrizione, cambio orario, ecc);
* la descrizione (ad esempio se la richiesta è un'iscrizione allora in questo campo verrà memorizzato il centro ippico in cui l'utente vuole iscriversi);
* l’utente che ha fatto la richiesta e l’amministratore che l’ha valutata;
* la data di conferma della richesta, se la richiesta è stata rifiutata allora la data verrà lasciata NULL.

Di ogni città vogliamo ricordare:

* il nome;
* la provincia.

Di ogni assicurazione ci interessa:

* l'allievo a cui è riferita;
* la descrizione dell’assicurazione;
* il premio assicurativo;
* la data di scadenza.

All’interfaccia web è consentito l’accesso con cinque diverse modalità: utenti non registrati, utenti registrati, allievi, dipendenti e amministratori. Ogni utente ha accesso a diverse funzionalità a seconda del gruppo a cui appartiene.

Gli utenti non registrati non vengono gestiti dal database ma hanno accesso in sola lettura alle informazioni riguardanti i recapiti dei centri ippici, i profili dei cavalli e la lista dei concorsi. Inoltre, se interessati a partecipare alle lezioni di equitazione, possono registrarsi e, solo successivamente, fare richiesta di iscrizione ad un centro ippico. Tutti gli altri utenti sono gestiti dal database e, in aggiunta a queste funzionalità, possono compiere altre operazioni. Gli utenti registrati possono fare richiesta di iscrizione agli amministratori, visualizzare e modificare il proprio profilo. Gli allievi, inoltre, possono fare altri tipi di richieste, consultare i propri orari di lezione e visualizzare e modificare il proprio profilo. Se sono allievi agonistici, possono iscriversi ai nuovi concorsi in programma. I dipendenti possono visualizzare e modificare il proprio profilo. Gli istruttori, possono anche visualizzare l’orario delle lezioni. Infine gli amministratori hanno i privilegi in lettura e scrittura per quanto riguarda i centri ippici, i concorsi, i dipendenti, gli allievi e i cavalli del proprio maneggio e l’approvazione o il rifiuto delle richieste degli utenti registrati.

PROGETTAZIONE CONCETTUALE

DESCRIZIONE TESTUALE DELLE CLASSI

* Centri ippici: modella un centro ippico

Attributi:

* Email: string «PK», indirizzo e-mail del centro ippico e chiave primaria.
* Nome: string, nome del centro ippico.
* Indirizzo: string, indirizzo postale del centro ippico.
* Telefono: string, numero di telefono del centro ippico.
* UtentiRegistrati: modella un utente registrato generico

Attributi:

* CodFiscale: string «PK», codice fiscale dell’utente e chiave primaria.
* Nome: string, nome dell’utente.
* Cognome: string, cognome dell’utente.
* DataNascita: date, data di nascita dell’utente, l'età dell'utente deve essere superiore ai sette anni. Per garantire questa proprietà verrà creato un trigger.
* CAP: string, codice di avviamento postale.
* Indirizzo: string, indirizzo postale dell’utente.
* Email: string, indirizzo e-mail dell'utente.
* NumTel: string, numero di telefono dell’utente.
* Password: password, password relativa all’utente per l’accesso all’interfaccia web. ???

Sono definite le seguenti sottoclassi di Utenti (con vincolo di partizione):

DIPENDENTI: descrive un dipendente ed ha i seguenti attributi

* Stipendio: double, stipendio relativo al dipendente.

Sono definite le seguenti sottoclassi di Dipendenti con vincolo di copertura: Amministratori, Istruttori, Stallieri. La classe Stallieri non ha alcun attributo.

ALLIEVI: descrive un allievo del centro ippico ed ha i seguenti attributi

* Orario: string (multivalore), orario della lezione dell’allievo. L’orario è descritto dai giorni della settimana in cui l’allievo ha lezione (es.: LU ME VE) e dall’ora in cui svolge l’attività equestre, la stessa per ogni giorno di lezione.

Sono definite le seguenti sottoclassi di Allievi con vincolo di partizione: Agonistici, MezzaFida, SenzaFida.

Attributi di Agonistici:

* Patente: string, patente dell’agonista

La classe SenzaFida non ha alcun attributo.

* Cavalli: modella un cavallo di un centro ippico

Attributi:

* CodPassaporto: string «PK», codice del passaporto del cavallo e chiave primaria.
* Nome: string, nome del cavallo.
* Razza: string, razza di appartenenza del cavallo.
* AnnoNascita: int, anno di nascita del cavallo.
* Tipo: string, disciplina equestre preferita dal cavallo.
* Concorsi: modella un generico concorso equestre

Attributi:

* CodConcorso: string «PK», codice identificativo del concorso e chiave primaria.
* Desc: string, descrizione della tipologia del concorso.
* DataInizio: date, indica la data di inizio concorso.
* DataFine: date, indica la data di fine concorso. Se il concorso dura un giorno allora coinciderà con DataInizio.
* Premio: string, descrizione del primo premio del concorso.

Sono definite le seguenti sottoclassi di Concorsi con vincolo di partizione: Interni ed Esterni.

Attributi di Esterni:

* Organizzatore: string, nome del centro ippico organizzatore.
* Indirizzo: string, indirizzo postale del centro ippico in cui si svolgerà il concorso.
* Telefono: string, recapito telefonico del centro ippico organizzatore.
* E-mail: string, indirizzo e-mail del centro ippico organizzatore.
* Richieste: modella una generica richiesta da parte di un utente ad un amministratore.

Attributi:

* CodRichiesta: contatore «PK», codice identificativo della richiesta e chiave primaria.
* TipoRichiesta: string, descrizione della tipologia della richiesta.
* Desc: string, descrizione della richiesta.
* Citta: modella una generica città

Attributi:

* CodCitta: contatore «PK», codice identificativo della città e chiave primaria.
* Nome: string, nome della città.
* Prov: string, provincia in cui risiede la città.
* Assicurazioni: modella una generica assicurazione

Attributi:

* CodAssicurazione: contatore «PK», codice che identifica l’assicurazione e chiave primaria.
* Desc: string, descrizione del tipo dell’assicurazione.
* Quota: double, premio assicurativo dell’assicurazione.
* Scadenza: date, data di scadenza dell'assicuraione.

DESCRIZIONE TESTUALE DELLE ASSOCIAZIONI

* Centri ippici - Cavalli: "possiede"  
  Molteplicità: 1:N, un centro ippico può possedere più di un cavallo, mentre un cavallo appartiene ad un solo centro ippico.  
  Totalità: totale verso Centri ippici e totale verso Cavalli, un centro ippico possiede almeno un cavallo mentre un cavallo deve appartenere ad un solo centro ippico.
* Dipendenti - Centri ippici: "lavora in"  
  Molteplicità: N:1, in un centro ippico possono lavorare più dipendenti mentre un dipendente lavora in un solo centro ippico.  
  Totalità: totale verso Dipendenti e totale verso Centri ippici, un dipendente lavora in un solo centro ippico mentre in un centro ippico lavorano uno o più dipendenti.
* Allievi - Centri ippici: "iscritto a"  
  Molteplicità: N:1, ad un centro ippico possono essere iscritti più allievi mentre un allievo si può iscrivere ad un solo centro ippico.  
  Totalità: totale verso Allievi e totale verso Centri ippici, un allievo è iscritto ad un solo centro ippico mentre ad un centro ippico sono iscritti uno o più allievi.
* Interni - Centri ippici: "si tiene in"  
  Molteplicità: 1:N, un concorso interno è tenuto da un solo centro ippico mentre un centro ippico può tenere più concorsi interni.  
  Totalità: parziale verso Interni e totale verso Centri ippici, un concorso iterno è tenuto da un solo centro ippico mentre un centro ippico può tenere zero o più concorsi interni.
* Centri ippici - Citta: "ha sede in"  
  Molteplicità: 1:N, un centro ippico ha sede in una città mentre una città può essere sede di più centri ippici  
  Totalità: totale verso Centri ippici e parziale verso Citta, un centro ippico ha sede in una sola città mentre una città può essere sede di zero o più centri ippici.
* Utenti registrati - Citta: "risiede a"  
  Molteplicità: N:1, un utente registrato risiede in una città mentre in una cità possono essere risiedere più utenti registrati.  
  Totalità: parziale verso Utenti registrati e totale verso Citta, un utente risiede in una città mentre in una città possono essere risiedere zero o più utenti registrati.
* Utenti registrati - Richieste: "compila"  
  Molteplicità: 1:N, un utente registrato può compilare più richieste mentre una richiesta può essere compilata da un utente.  
  Totalità: totale verso Utenti registrati e parziale verso Richieste, un utente può compilare zero o più richieste mentre una richiesta deve essere compilata da un solo utente.
* Assicurazioni - Allievi: "copre"  
  Molteplicità: 1:1, un assicurazione copre un solo utente e un utente è coperto da una sola assicurazione.  
  Totalità: totale verso Assicurazioni e totale verso Allievi, un assicurazione copre un allievo e un allievo è coperto da un assicurazione.
* Istruttori - Allievi: "allena"  
  Molteplictà: 1:N, un istruttore può allenare più llievi mentre un allievo è allenato da un solo istruttore  
  Totalità: totale verso Istruttori e parziale verso Allievi, un istruttore allena nessuno o più allievi mentre un allievo è allenato da un solo istruttore.
* Mezza fida - Cavalli: "ha in affidamento"  
  Molteplicità: N:1, un utente mezza fida ha in affidamento un solo cavallo mentre un cavallo è affidato ad al massimo due utenti mezza fida, per garantire questa proprietà verrà creato una apposita funzione.  
  Totalità: parziale verso Mezza fida e totale verso Cavalli, un mezza fida ha in affidamento un cavallo mentre un cavallo è in affidamento a zero, uno o due mezza fida.
* Cavalli - Agonistici: "è affidato a"  
  Molteplicità: 1:1, un cavallo è affidato a un solo agonistico e un agonistico ha in affidamento un solo cavallo.  
  Totalità: totale verso Cavalli e parziale verso Agonistici, un cavallo è affidato a zero o un allievo agonistico e un agonistico ha in affidamento un solo cavallo.
* Agonistici - Concorsi: "partecipa"  
  Molteplicità: N:M, un allievo agonistico può partecipare a più concorsi e ad un consorso possono prendere parte più agonistici.  
  Totalità: parziale verso Agonistici e parziale verso Concorsi, un allievo agonistico può partecipare a zero o più concorsi e ad un concorso possono partcipare zero o più agonistici.
* Amministratori - Richieste: "convalida"  
  Molteplicità: 1:N, un amministratore può convalidare più richieste mentre una richiesta è convalidata da un solo amministratore  
  Totalità: parziale verso Amministratori e parziale verso Richieste, un amministratore può convalidare zero o più richieste, una richiesta convalidata è associata ad un solo amministratore mentre, se in attesa di convalida non è associata a nessun amministratore.
* Citta - Esterni: "ospita"  
  Molteplicità: 1:N, una città può ospitare più concorsi esterni mentre un concorso esterno è ospitato da una sola città  
  Totalità: totale verso Citta e parziale verso Esterni, una città può ospitare zero o più concosrsi esterni mentre un concorso esterno deve essere ospitato da una sola città.

DESCRIZIONE TESTUALE DELLE GERARCHIE

Lo schema concettuale prevede quattro gerarchie.

* La classe Utenti registrati ha le sottoclassi Dipendenti e Allievi. Questa gerarchia ha vincolo di disgiuzione in quanto gli utenti registrati possono appartenere alla classe dei Dipendenti o a quella degli Allievi oppure possono essere utenti registrati che non hanno ancora compilato una richiesta di iscrizione.
* Dipendenti è superclasse di Amministratori, Stallieri ed Istruttori. Questa gerarchia ha vincolo di copertura, infatti la superclasse equivale all'insieme delle sottoclassi e queste ultime non sono disgiunte tra loro.
* Allievi è superclasse di SenzaFida, MezzaFida ed Agonistici. Questa gerarchia ha vincolo di partizione in quanto l'unione delle sottoclassi, tra loro disgiunte, equivale alla superclasse stessa.
* Infine, la classe Concorsi ha le sottoclssi Interni ed Esterni. Questa gerarchia ha vincolo di partizione in quanto, come per la gerarchia precedente, l'unione delle sottoclassi, tra loro disgiunte, equivale alla superclasse stessa.

PROGETTAZIONE LOGICA

Descrizione testuale dello schema relazionale

* Citta (CodCitta: string, Nome: string, Prov: string, CAP: string)

- PK(CodCitta)

- NOT NULL CodCitta, Nome, Provincia

* Centri ippici (Email: string,Nome: string, CodCitta\*: string, Indirizzo: string, Tel: string)

- PK(Email)

- CodCitta FK(Citta)

NOT NULL Email, Nome, CodCitta, Indirizzo

* Asicurazioni (CodAss: int, CodAllievo\*: string, Desc: string, Quota: double, Scadenza: date)

- PK(CodAss)

- CodAllievo FK(Allievi)

- NOT NULL CodAss, CodAllievo, Desc, Quota, Scadenza

* Cavalli (CodPassaporto: string, Nome: string, Razza: string, AnnoNascita: int, Tipo: string, CentroIppico\*: string)

- PK(CodPassaporto)

- CentroIppico FK(CentriIppici)

- NOT NULL CodPassaporto, Nome, Razza, AnnoNascita, Tipo, CentroIppico

* UtentiRegistrati (CodiceFiscale: string, Nome: string, Cognome: string, DataNascita: date, CAP: string, Riesede\*: string, CAP: string, Indirizzo: string, NumTel: string, Password: string)

- PK(CodiceFiscale)

- Risiede FK(Citta)

- NOT NULL CodiceFiscale, Nome, Cognome, Nato, DataNascita, CAP, Indirizzo, Password

* Dipendenti (CodDip\*: string, CentroIppico\*: string, Stipendio: double)

- PK(CodDip)

- CodDip FK(UtentiRegistrati)

- CentroIppico FK(CentriIppici)

- NOT NULL CodDip, CentroIppico, Stipendio

* Amministratori (CodAmm\*: string)

- PK(CodAmm)

- CodAmm FK(Dipendenti)

- NOT NULL CodAmm

* Istruttori (CodIstr\*: string)

- PK(CodIstr)

- CodIstr FK(Dipendenti)

- NOT NULL CodIstr

* Allievi (CodAllievo\*: string, CodIstr\*: string, Orario: string, CentroIppico\*: string)

- PK(CodAllievo)

- Codistr FK(Istruttori)

- CentroIppico FK(CentriIppici)

- NOT NULL CodAllievo, CodIstr, Orario, CentriIppico

* MezzaFida (CodMF\*: string, CodCavallo\*: string)

- PK(CodMF)

- CodMF FK(Allievi)

- CodCavallo FK(Cavalli)

- NOT NULL CodMF, CodCavallo

* Agonistici (CodAg\*: string, CodCavallo\*: string, Patente: string)

- PK(CodAg)

- CodAg FK(Allievi)

- CodCavallo FK(Cavalli)

- NOT NULL CodAg, CodCavallo, Patente

* Concorsi (CodConcorso: string, Desc: string, DataInizio: date, DataFine: string, Premio: string)

- PK(CodConcorso)

- NOT NULL CodConcorso, Desc, DataInizio, DataFine, Premio

* Interni (CodConcorso\*: string, CentroIppico\*: string)

- PK(CodConcorso)

- CodConcorso FK(Concorso)

- CentroIppico FK(CentriIppici)

- NOT NULL CodConcorso, CentroIppico

* Esterni (CodConcorso\*: string, Organizzatore: string, CodCitta\*: string, Indirizzo: string, Telefono: string, Email: string)

- PK(CodConcorso)

- CodConcorso FK(Concorso)

- CodCitta FK(Citta)

- NOT NULL CodConcorso, Organizzatore, CodCitta, Indirizzo, Email

* Partecipazioni (CodConcorso\*: string, CodAg\*: string, Vincitore: string)

- PK(CodConcorso)

- CodConcorso FK(Concorso)

- CodAg FK(Agonistici)

- NOT NULL CodConcorso, CodAg

* Richieste (CodRichiesta: string, TipoRichiesta: string, Descrizione: string, CodUtente\*: string, CodAmm\*: string)

- PK(CodRichiesta)

- CodUtente FK(UtentiRegistrati)

- CodAmm FK(Amministratori)

- NOT NULL CodRichiesta, TipoRichiesta, Descrizione, CodUtente

TRASFORMAZIONE DELLE GERARCHIE

Le gerarchie presenti nel modello concettuale della base di dati sono state sviluppate nel modello logico tramite il partizionamento verticale. Questa scelta è dettata dalle seguenti ragioni:

* le sottoclassi di ogni gerarchia intervengono in associazioni diverse da quelle delle superclasse, quindi sia il partizionamento orizzontale sia la relazione unica con discriminante avrebbero reso dificile la gestione di tali associazioni.
* inoltre, per quanto riguarda le gerarchie di UtentiRegistrati, Concorsi e Allievi, la relazione unica con il discriminante avrebbe creato una sola tabella con gli attributi di tutte le classi facenti parte della gerarchia. In questo modo sarebbe stato impossible richiedere il vincolo NOT NULL su questi attributi, vincolo che invece è importante per la consistenza della base di dati.

TRASFORMAZIONE DEL CAMPO Orario MULTIVALORE

Il campo in questione, nella definizione dello schema logico relazionale, non è stato modificato. Non abbiamo ritenuto necessaria la trasformazione del campo multivalore in una tabella che contiene i possibili valori dell'attributo con una chiave esterna verso la tabella originale, in quanto non indispensabile ai fini dello sviluppo della nostra base di dati. Infatti, la tabella Orario, se fosse stata creata, non sarebbe stata utilizzata. È onere dell'amministratore controllare che non ci siano conflitti di orari per quanto riguarda gli utenti MezzaFida che condividono lo stesso cavallo. In caso di richiesta di cambio orario conflittuale (richiesta improbabile) l'amministratore dovrà rifiutare la stessa.

DEFINIZIONE DELLO SCHEMA LOGICO TRAMITE DDL DI MySql

**SET** FOREIGN\_KEY\_CHECKS**=**0**;**

/\* Creazione tabella CentriIppici \*/

**DROP** **TABLE** **IF** **EXISTS** `CentriIppici`**;**

**CREATE** **TABLE** `CentriIppici` **(**

`Email` **VARCHAR(**40**)** **NOT** **NULL,**

`Nome` **VARCHAR(**20**)** **NOT** **NULL,**

`CodCitta` **INT** **NOT** **NULL,**

`Indirizzo` **VARCHAR(**20**)** **NOT** **NULL,**

`Tel` **VARCHAR(**11**)** **NOT** **NULL,**

**PRIMARY** **KEY(**Email**),**

**FOREIGN** **KEY(**CodCitta**)** **REFERENCES** Citta**(**CodCitta**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Cavalli \*/

**DROP** **TABLE** **IF** **EXISTS** `Cavalli`**;**

**CREATE** **TABLE** `Cavalli`**(**

`CodPassaporto` **CHAR(**15**)** **NOT** **NULL,**

`Nome` **VARCHAR(**20**)** **NOT** **NULL,**

`Razza` **VARCHAR(**20**)** **NOT** **NULL,**

`AnnoNascita` **INT(**4**)** **NOT** **NULL,**

`Tipo` **VARCHAR(**20**)** **NOT** **NULL,**

`CentroIppico` **VARCHAR(**40**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodPassaporto**),**

**FOREIGN** **KEY(**CentroIppico**)REFERENCES** CentriIppici**(**Email**)** **ON** **DELETE** **CASCADE**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella UtentiRegistrati \*/

**DROP** **TABLE** **IF** **EXISTS** `UtentiRegistrati`**;**

**CREATE** **TABLE** `UtentiRegistrati`**(**

`CodiceFiscale` **CHAR(**16**)** **NOT** **NULL,**

`Nome` **VARCHAR(**20**)** **NOT** **NULL,**

`Cognome` **VARCHAR(**20**)** **NOT** **NULL,**

`DataNascita` **DATE** **NOT** **NULL,**

`Risiede` **INT** **NOT** **NULL,**

`CAP` **CHAR(**5**)** **NOT** **NULL,**

`Indirizzo` **VARCHAR(**20**)** **NOT** **NULL,**

`Email` **VARCHAR(**40**)** **NOT** **NULL** **UNIQUE,**

`NumTel` **VARCHAR(**10**),**

`Password` **VARCHAR(**50**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodiceFiscale**),**

**FOREIGN** **KEY(**Risiede**)** **REFERENCES** Citta**(**CodCitta**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Dipendenti \*/

**DROP** **TABLE** **IF** **EXISTS** `Dipendenti`**;**

**CREATE** **TABLE** `Dipendenti` **(**

`CodDip` **CHAR(**16**)** **NOT** **NULL,**

`CentroIppico` **VARCHAR(**40**)** **NOT** **NULL,**

`Stipendio` **DOUBLE** **NOT** **NULL,**

**PRIMARY** **KEY(**CodDip**),**

**FOREIGN** **KEY(**CodDip**)** **REFERENCES** UtentiRegistrati**(**CodiceFiscale**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CentroIppico**)** **REFERENCES** CentriIppici**(**Email**)** **ON** **DELETE** **CASCADE**

**)** ENGINE**=**InnoDB**;**

/\* Creazione tabella Amministratori \*/

**DROP** **TABLE** **IF** **EXISTS** `Amministratori`**;**

**CREATE** **TABLE** `Amministratori`**(**

`CodAmm` **CHAR(**16**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodAmm**),**

**FOREIGN** **KEY(**CodAmm**)** **REFERENCES** Dipendenti**(**CodDip**)** **ON** **DELETE** **CASCADE**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Istruttori \*/

**DROP** **TABLE** **IF** **EXISTS** `Istruttori`**;**

**CREATE** **TABLE** `Istruttori`**(**

`CodIstr` **CHAR(**16**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodIstr**),**

**FOREIGN** **KEY(**CodIstr**)** **REFERENCES** Dipendenti**(**CodDip**)** **ON** **DELETE** **CASCADE**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Allievi \*/

**DROP** **TABLE** **IF** **EXISTS** `Allievi`**;**

**CREATE** **TABLE** `Allievi` **(**

`CodAllievo` **CHAR(**16**)** **NOT** **NULL,**

`CodIstr` **CHAR(**16**)** **NOT** **NULL,**

`Orario` **VARCHAR(**40**)** **NOT** **NULL,**

`CentroIppico` **VARCHAR(**40**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodAllievo**),**

**FOREIGN** **KEY(**CodAllievo**)** **REFERENCES** UtentiRegistrati**(**CodiceFiscale**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CodIstr**)** **REFERENCES** Istruttori**(**CodIstr**),**

**FOREIGN** **KEY(**CentroIppico**)** **REFERENCES** CentriIppici**(**Email**)** **ON** **DELETE** **CASCADE**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella MezzaFida \*/

**DROP** **TABLE** **IF** **EXISTS** `MezzaFida`**;**

**CREATE** **TABLE** `MezzaFida` **(**

`CodMF` **CHAR(**16**)** **NOT** **NULL,**

`CodCavallo` **CHAR(**15**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodMF**),**

**FOREIGN** **KEY(**CodMF**)** **REFERENCES** Allievi**(**CodAllievo**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CodCavallo**)** **REFERENCES** Cavalli**(**CodPassaporto**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Agonistici \*/

**DROP** **TABLE** **IF** **EXISTS** `Agonistici`**;**

**CREATE** **TABLE** `Agonistici` **(**

`CodAg` **CHAR(**16**)** **NOT** **NULL,**

`CodCavallo` **CHAR(**15**)** **NOT** **NULL,**

`Patente` **VARCHAR(**10**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodAg**),**

**FOREIGN** **KEY(**CodAg**)** **REFERENCES** Allievi**(**CodAllievo**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CodCavallo**)** **REFERENCES** Cavalli**(**CodPassaporto**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Assicurazioni \*/

**DROP** **TABLE** **IF** **EXISTS** `Assicurazioni`**;**

**CREATE** **TABLE** `Assicurazioni` **(**

`CodAss` **INT** AUTO\_INCREMENT**,**

`CodAllievo` **CHAR(**16**),**

`**Desc**` **VARCHAR(**60**)** **NOT** **NULL,**

`Quota` **DOUBLE** **NOT** **NULL,**

`Scadenza` **DATE** **NOT** **NULL,**

**PRIMARY** **KEY(**CodAss**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Richieste \*/

**DROP** **TABLE** **IF** **EXISTS** `Richieste`**;**

**CREATE** **TABLE** `Richieste` **(**

`CodRichiesta` **INT** AUTO\_INCREMENT**,**

`TipoRichiesta` **VARCHAR(**20**)** **NOT** **NULL,**

`Descrizione` **VARCHAR(**40**)** **NOT** **NULL,**

`CodUtente` **CHAR(**16**)** **NOT** **NULL,**

`CodAmm` **CHAR(**16**),**

`DataConvalida` **DATE,**

**PRIMARY** **KEY(**CodRichiesta**),**

**FOREIGN** **KEY(**CodUtente**)** **REFERENCES** UtentiRegistrati**(**CodiceFiscale**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CodAmm**)** **REFERENCES** Amministratori**(**CodAmm**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Concorsi \*/

**DROP** **TABLE** **IF** **EXISTS** `Concorsi`**;**

**CREATE** **TABLE** `Concorsi` **(**

`CodConcorso` **CHAR(**4**)** **NOT** **NULL,**

`**Desc**` **VARCHAR(**30**)** **NOT** **NULL,**

`DataInizio` **DATE** **NOT** **NULL,**

`DataFine` **DATE** **NOT** **NULL,**

`Premio` **VARCHAR(**15**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodConcorso**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Interni \*/

**DROP** **TABLE** **IF** **EXISTS** `Interni`**;**

**CREATE** **TABLE** `Interni` **(**

`CodConcorso` **CHAR(**4**)** **NOT** **NULL,**

`CentroIppico` **VARCHAR(**40**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodConcorso**),**

**FOREIGN** **KEY(**CodConcorso**)** **REFERENCES** Concorsi**(**CodConcorso**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CentroIppico**)** **REFERENCES** CentriIppici**(**Email**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Esterni \*/

**DROP** **TABLE** **IF** **EXISTS** `Esterni`**;**

**CREATE** **TABLE** `Esterni` **(**

`CodConcorso` **CHAR(**4**)** **NOT** **NULL,**

`Organizzatore` **VARCHAR(**30**)** **NOT** **NULL,**

`CodCitta` **INT** **NOT** **NULL,**

`Indirizzo` **VARCHAR(**20**)** **NOT** **NULL,**

`Telefono` **CHAR(**10**),**

`Email` **VARCHAR(**40**)** **NOT** **NULL,**

**PRIMARY** **KEY(**CodConcorso**),**

**FOREIGN** **KEY(**CodConcorso**)** **REFERENCES** Concorsi**(**CodConcorso**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CodCitta**)** **REFERENCES** Citta**(**CodCitta**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Partecipazioni \*/

**DROP** **TABLE** **IF** **EXISTS** `Partecipazioni`**;**

**CREATE** **TABLE** `Partecipazioni` **(**

`CodConcorso` **CHAR(**4**)** **NOT** **NULL,**

`CodAg` **CHAR(**16**),**

`Vincitore` **VARCHAR(**40**),**

**PRIMARY** **KEY(**CodConcorso**,**CodAg**),**

**FOREIGN** **KEY(**CodConcorso**)** **REFERENCES** Concorsi**(**CodConcorso**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY(**CodAg**)** **REFERENCES** Agonistici**(**CodAg**)** **ON** **DELETE** **CASCADE**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Citta \*/

**DROP** **TABLE** **IF** **EXISTS** `Citta`**;**

**CREATE** **TABLE** `Citta`**(**

`CodCitta` **INT** AUTO\_INCREMENT**,**

`Nome` **VARCHAR(**20**)** **NOT** **NULL,**

`Provincia` **CHAR(**2**),**

**PRIMARY** **KEY(**CodCitta**)**

**)**ENGINE**=**InnoDB**;**

/\* Creazione tabella Errori per la gestione dei trigger \*/

**DROP** **TABLE** **IF** **EXISTS** `Errori`**;**

**CREATE** **TABLE** `Errori`**(**

`CodErrore` **INT** AUTO\_INCREMENT**,**

`Descrizione` **VARCHAR(**50**),**

**PRIMARY** **KEY(**CodErrore**)**

**)**ENGINE**=**InnoDB**;**

**SET** FOREIGN\_KEY\_CHECKS**=**1**;**

QUERY

Di seguito sono riportate le query di maggior rilevanza utilizzate nell'interfaccia web.

1) /\* Visualizza le informazioni relative ai centri ippici \*/

**SELECT** c**.**Nome**,** ci**.**Nome **AS** Citta**,** c**.**Indirizzo**,** c**.**Tel**,** c**.**Email

**FROM** CentriIppici c **JOIN** Citta ci **ON** **(**c**.**CodCitta**=**ci**.**CodCitta**);**

Questa query permette di visualizzare le informazioni generali di tutti i centri ippici presenti nel database.

OUTPUT

2) \* Visualizza le informazioni relative ai cavalli di uno specifico centro ippico \*/

**SELECT** c**.**Nome**,** c**.**Razza**,** c**.**AnnoNascita**,** c**.**Tipo

**FROM** Cavalli c **JOIN** CentriIppici ci **ON** **(**c**.**CentroIppico**=**ci**.**Email**)**

**WHERE** ci**.**Nome**=**$select**;**

Il codice sopra riportato restituisce tutte le informazioni relative ai cavalli presenti in uno specifico centro ippico.

L'output seguente è il risultato della query con $select fissato a 'centroippico3province@email.it'

OUTPUT

3) /\* Visualizza lo storico dei concorsi \*/

**SELECT** c**.Desc,** c**.**DataInizio**,** c**.**DataFine**,** c**.**Premio**,** ci**.**Nome **AS** CentroIppico**,** ct**.**Nome **AS** Citta

**FROM** Concorsi c **JOIN** Interni i **ON** **(**c**.**CodConcorso**=**i**.**CodConcorso**)** **JOIN** CentriIppici ci **ON** **(**i**.**CentroIppico**=**ci**.**Email**)** **JOIN** Citta ct **ON** **(**ci**.**CodCitta**=**ct**.**CodCitta**)**

**WHERE** c**.**DataFine**<**CURDATE**()**

**UNION**

**SELECT** c**.Desc,** c**.**DataInizio**,** c**.**DataFine**,** c**.**Premio**,** e**.**Organizzatore **AS** CentroIppico**,** ct**.**Nome **AS** Citta

**FROM** Concorsi c **JOIN** Esterni e **ON** **(**c**.**CodConcorso**=**e**.**CodConcorso**)** **JOIN** Citta ct **ON** **(**e**.**CodCitta**=**ct**.**CodCitta**)**

**WHERE** c**.**DataFine**<**CURDATE**();**

La query sopra riportata seleziona i concorsi per i quali DataFine è minore della data corrente, in questo modo solo i concorsi che sono terminati potranno essere visualizzati.

OUTPUT

4) /\* Visualizza i concorsi a cui è possibile iscriversi \*/

**SELECT** c**.Desc,** c**.**DataInizio**,** c**.**DataFine**,** c**.**Premio**,** ci**.**Nome **AS** CentroIppico**,** ct**.**Nome **AS** Citta

**FROM** Concorsi c **JOIN** Interni i **ON** **(**c**.**CodConcorso**=**i**.**CodConcorso**)** **JOIN** CentriIppici ci **ON** **(**i**.**CentroIppico**=**ci**.**Email**)** **JOIN** Citta ct **ON** **(**ci**.**CodCitta**=**ct**.**CodCitta**)**

**WHERE** c**.**DataInizio**>**CURDATE**()**

**UNION**

**SELECT** c**.Desc,** c**.**DataInizio**,** c**.**DataFine**,** c**.**Premio**,** e**.**Organizzatore **AS** CentroIppico**,** ct**.**Nome **AS** Citta

**FROM** Concorsi c **JOIN** Esterni e **ON** **(**c**.**CodConcorso**=**e**.**CodConcorso**)** **JOIN** Citta ct **ON** **(**e**.**CodCitta**=**ct**.**CodCitta**)**

**WHERE** c**.**DataInizio**>**CURDATE**();**

Questa query visualizza solo i concorsi che non sono ancora iniziati ed ai quali gli agonisti possono iscriversi.

OUTPUT

5) /\* Visualizza gli allievi la cui assicurazione scade entro 14 giorni \*/

**SELECT** CentroIppico **FROM** Dipendenti **WHERE** CodDip**=**$codfiscale**;**

La query sopra riportata serve a visualizzare il centro ippico in cui lavora l'amministratore. Questo è necessario perchè ogni amministratore può visualizzare solo le assicurazioni in scadenza degli allievi del proprio centro ippico.

**SELECT** u**.**Nome**,** u**.**Cognome**,** u**.**Email**,** ass.Quota, ass**.**Scadenza

**FROM** UtentiRegistrati u **JOIN** Allievi al **ON** **(**u**.**CodiceFiscale**=**al**.**CodAllievo**)** **JOIN** Assicurazioni ass **ON** **(**al**.**CodAllievo**=**ass**.**CodAllievo**)**

**WHERE** **MONTH(**CURDATE**())+**1**=MONTH(**ass**.**Scadenza**) AND** al**.**CentroIppico**=**$centroi**;**

La query visualizza nome, cognome, indirizzo e-mail, quota assicurativa e giorno di scadenza di tutti gli allievi del centro ippico $centroi ai quali scade l'assicurazione nei prossimi 14 giorni.

L'output seguente è il risultato delle query sopra riportate con $codfiscale fissato a e di conseguenza $centroi uguale a .

OUTPUT

6) /\* Visualizza le richieste da convalidare per gli utenti di uno specifico centro ippico \*/

**SELECT** CentroIppico **FROM** Dipendenti **WHERE** CodDip**=**$codfiscale**;**

La query sopra riportata, come nel caso precendente, serve a selezionare il centro ippico in cui lavora l'amministratore. In questo modo l'amministratore di riferimento potrà accedere solo alle richieste a lui "inviate".

**SELECT** r**.**CodRichiesta**,** r**.**TipoRichiesta**,** r**.**Descrizione**,** u**.**CodiceFiscale

**FROM** Richieste r **JOIN** UtentiRegistrati u **ON** **(**r**.**CodUtente**=**u**.**CodiceFiscale**)** **JOIN** Allievi a **ON** **(**u**.**CodiceFiscale**=**a**.**CodAllievo**)**

**WHERE** a**.**CentroIppico=$centroi **AND** r**.**CodAmm **is** **NULL**

**UNION**

**SELECT** r**.**CodRichiesta**,** r**.**TipoRichiesta**,** r**.**Descrizione**,** u**.**CodiceFiscale

**FROM** Richieste r **JOIN** UtentiRegistrati u **ON** **(**r**.**CodUtente**=**u**.**CodiceFiscale**)** **JOIN** Dipendenti d **ON** **(**u**.**CodiceFiscale**=**d**.**CodDip**)**

**WHERE** d**.**CentroIppico=$centroi **AND** r**.**CodAmm **is** **NULL**

**UNION**

**SELECT** r**.**CodRichiesta**,** r**.**TipoRichiesta**,** r**.**Descrizione**,** u**.**CodiceFiscale

**FROM** Richieste r **JOIN** UtentiRegistrati u **ON** **(**r**.**CodUtente**=**u**.**CodiceFiscale**)**

**WHERE** r**.**Descrizione=$centroi **AND** r**.**CodAmm **is** **NULL;**

Questa query permette all'amministratore di visualizzare le richieste degli allievi, dei dipendenti e degli utenti registrati che vogliono iscriversi al centro ippico in cui lavora. Solo le richieste dove CodAmm IS NULL sono richieste ancora da confermare o da rifiutare in quanto l'amministratore non ha ancora messo la sua "firma".

L'output sotto riportato è il risultato delle query appena descritte. $codfiscale è stato fissato a e di conseguenza $centroi è uguale a .

OUTPUT

7)/\* Inserimento e rimozione di un cavallo \*/

**SELECT** CentroIppico **FROM** Dipendenti **WHERE** CodDip=$codfiscale**;**

Query che seleziona il centro ippico dell'amministratore che vuole aggiungere un cavallo al suo centro ippico per, ad esempio, un acquisto.

/\* Inserimento \*/

**INSERT** **INTO** `Cavalli`**(**`CodPassaporto`**,** `Nome`**,** `Razza`**,** `AnnoNascita`**,** `Tipo`**,** `CentroIppico`**)** **VALUES** **(**$cod**,** $nome**,** $razza**,** $annon**,** $tipo**,** $centroi**);**

Questa query inserisce un cavallo appartenente al centro ippico $centroi.

/\* Rimozione \*/

**DELETE** **FROM** `Cavalli` **WHERE** CodPassaporto**=**$cod**;**

Questa query rimuove il cavallo con il codice $cod. Essendo che l'amministratore può visualizzare il codice dei soli cavalli appartenenti al centro ippico in cui lavora, non c'è pericolo che elimini dal database il cavallo di un altro centro ippico.

8) /\* Visualizza i dati relativi ad un utente \*/

**SELECT** u**.**Cognome**,** u**.**Nome**,** u**.**DataNascita**,** c**.**Nome **AS** Citta**,** u**.**Indirizzo**,** u**.**Email**,** u**.**NumTel

**FROM** UtentiRegistrati u **JOIN** Citta

**WHERE** CodiceFiscale**=**$codfiscale**;**

Questa query seleziona i dati relativi ad un utente per permettergli di visualizzare il proprio profilo.

L'output sotto riportato è il risultato della query con $codfiscale uguale a .

OUTPUT

9) /\* Visualizza i dati relativi alle richieste di un utente \*/

**SELECT** Descrizione**,** CodAmm**,** DataConvalida

**FROM** Richieste

**WHERE** CodUtente**=**$codfiscale**;**

Questa query seleziona i dati relativi alle richieste di un utente. L'utente nell'interfaccia web visualizzerà la descrizione e la data di convalida se la richiesta è stata confermata, la descrizione e "In attesa" se CodAmm IS NULL (ciò significa che l'amministratore non ha ancora confermato la richiesta) oppure la descrizione e "Riufiutata" se CodAmm IS NOT NULL ma la data di conferma è NULL (ciò significa che l'amministratore ha valutato la richiesta ma l'ha rifiutata, quindi la data di conferma è NULL).

L'output sotto riportato è il risultato della query con $codfiscale uguale a .

FUNZIONI PROCEDURE E TRIGGERS

FUNZIONI

1) /\* Ritorna true se il cavallo non è in affidamento altrimenti ritorna false \*/

**DROP** **FUNCTION** **IF** **EXISTS** `CavalloSenzaFida`**;**

DELIMITER $

**CREATE** **FUNCTION** CavalloSenzaFida**(**CodPassaporto **INT(**15**))**

**RETURNS** **BOOLEAN**

**BEGIN**

**DECLARE** Allievi **INT;**

**SELECT** **COUNT(**CodAg**)** **INTO** Allievi

**FROM** Agonistici

**WHERE** CodCavallo**=**CodPassaporto**;**

**IF** **(NOT** Allievi**)** **THEN**

**SELECT** **COUNT(**CodMF**)** **INTO** Allievi

**FROM** MezzaFida

**WHERE** CodCavallo**=**CodPassaporto**;**

**END** **IF;**

**IF** **(NOT** Allievi**)** **THEN**

**RETURN** **TRUE;**

**ELSE**

**RETURN** **FALSE;**

**END** **IF;**

**END** $

La funzione CavalloSenzaFida ritorna TRUE se il codice del cavallo passato come parametro si riferisce ad un cavallo senza alcun allievo in fida, mentre ritorna false in caso contrario.

2) /\* Ritorna true se il cavallo è in affidamento ad un solo mezzafida altrimenti ritorna false \*/

**DROP** **FUNCTION** **IF** **EXISTS** `CavalloMezzaFida`**;**

DELIMITER $

**CREATE** **FUNCTION** CavalloMezzaFida**(**CodPassaporto **CHAR(**15**))**

**RETURNS** **BOOLEAN**

**BEGIN**

**DECLARE** Allievi **INT;**

**SELECT** **COUNT(**CodMF**)** **INTO** Allievi

**FROM** MezzaFida

**WHERE** CodCavallo**=**CodPassaporto**;**

**IF** **(**Allievi**=**1**)** **THEN**

**RETURN** **TRUE;**

**ELSE**

**RETURN** **FALSE;**

**END** **IF;**

**END** $

Questa funzione ritorna true solo se il codice del cavallo passato come parametro si riferisce ad un cavallo che è in mezza fida con un solo allievo. In questo caso infatti può essere scelto per un abbonamento mezza fida di un altro allievo.

3) /\* Cerca l'istruttore con il minor numero di allievi \*/

**DROP** **FUNCTION** **IF** **EXISTS** `MinAllievi`**;**

DELIMITER $**;**

**CREATE** **FUNCTION** MinAllievi**(**Centro **VARCHAR(**40**))** **RETURNS** **CHAR(**16**)**

**BEGIN**

**DECLARE** Istr **CHAR(**16**);**

**SELECT** CodIstr **INTO** Istr

**FROM** **(SELECT** CodIstr**,** **MIN(count)**

**FROM** **(SELECT** CodIstr**,COUNT(**CodIstr**)** **AS** **count**

**FROM** Allievi **WHERE** CentroIppico**=**Centro

**GROUP** **BY** CodIstr**)** **AS** Query1‏**)** **AS** Query2**;**

**RETURN** Istr**;**

**END** $

Questa funzione permette di sapere quale istruttore, di uno specifico centro ippico, ha il minor numero di allievi. Questa funzione è necessaria in quanto permette di associare un istruttore ad un nuovo allievo appena iscritto.

PROCEDURE

1) /\* Cambio orario \*/

**DROP** **PROCEDURE** **IF** **EXISTS** `CambioO`**;**

DELIMITER $

**CREATE** **PROCEDURE** CambioO**(**OrarioN **VARCHAR(**36**),** CodUtente **CHAR(**16**))**

**BEGIN**

**UPDATE** `Allievi`

**SET** Orario**=**OrarioN

**WHERE** CodAllievo**=**CodUtente**;**

**END** $

Questa procedura permette di cambiare l'orario di un allievo dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura vengono passati i parametri OrarioN, il nuovo orario, e CodUtente, codice dell'allievo che vuole cambiare orario.

2) /\* Aggiorna patente\*/

**DROP** **PROCEDURE** **IF** **EXISTS** `AggPat`**;**

DELIMITER $

**CREATE** **PROCEDURE** AggPat**(**PatenteN **VARCHAR(**36**),** CodUtente **CHAR(**16**))**

**BEGIN**

**UPDATE** `Allievi`

**SET** Patente**=**PatenteN

**WHERE** CodAg**=**CodUtente**;**

**END** $

Questa procedura permette di aggiornare la patente di un allievo agonista dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura vengono passati i parametri PatenteN, il nuovo codice della patente, e CodUtente, codice dell'agonista che vuole aggiornare la propria patente.

3) /\* Iscrizione ad allievi senza fida \*/

**DROP** **PROCEDURE** **IF** **EXISTS** `Iscrizione`**;**

DELIMITER $

**CREATE** **PROCEDURE** Iscrizione**(**CentroN **VARCHAR(**36**),** CodUtente **CHAR(**16**))**

**BEGIN**

**INSERT** **INTO** `Allievi` **(**`CodAllievo`**,** `CodIstr`**,** `Orario`**,** `CentroIppico`**)**

**VALUES** **(**CodUtente**,**MinAllievi**(**CentroN**),**'Da definire'**,**CentroN**);**

**INSERT** **INTO** `Assicurazioni`**(**`CodAllievo`**,** `**Desc**`**,** `Quota`**,** `Scadenza`**)**

**VALUES** **(**CodUtente**,**'Ass. ann infortuni'**,** 20**,**CURDATE**());**

**END**‏

Questa procedura permette di iscrivere un nuovo utente dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura vengono passati i parametri CentroN, il codice del centro ippico al quale l'utente vuole iscriversi, e CodUtente, codice dell'utente che ha fatto la richiesta.

4) /\* Cambio abbonamento da Agonistici o MezzaFida ad Allievi senza fida \*/

**DROP** **PROCEDURE** **IF** **EXISTS** `CaabAllievi`**;**

DELIMITER $

**CREATE** **PROCEDURE** CaabAllievi**(**CodUtente **CHAR(**16**),** Abb **VARCHAR(**10**))**

**BEGIN**

**IF** **(**Abb**=**'agonistici'**)** **THEN**

**DELETE** **FROM** `Agonistici`

**WHERE** CodAg**=**CodUtente**;**

**ELSE**

**DELETE** **FROM** `MezzaFida`

**WHERE** CodMF**=**CodUtente**;**

**END** **IF;**

**END** $

Questa procedura permette di cambiare abbonamento, da MezzaFida o Agonistici ad allievo senza fida, dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura vengono passati i parametri Abb, tipologia del vecchio abbonamento, e CodUtente, codice dell'allievo che vuole cambiare abbonamento.

5) /\* Cambio abbonamento da Allievi senza fida ad Agonistici o MezzaFida \*/

**DROP** **PROCEDURE** **IF** **EXISTS** `CaabMForAG`**;**

DELIMITER $

**CREATE** **PROCEDURE** CaabMForAG**(**CodUtente **CHAR(**16**),** Abb **VARCHAR(**10**),**CentroI **VARCHAR(**40**))**

**BEGIN**

**DECLARE** Cavallo **CHAR(**15**);**

**IF** **(**Abb**=**'agonistici'**)** **THEN**

**SELECT** CodPassaporto **INTO** Cavallo

**FROM** Cavalli

**WHERE** CentroIppico**=**CentroI **AND** CavalloSenzaFida**(**CodPassaporto**)>**0**;**

**IF** **(**Cavallo**)** **THEN**

**INSERT** **INTO** `Agonistici`**(**`CodAg`**,** `CodCavallo`**,** `Patente`**)**

**VALUES** **(**CodUtente**,**Cavallo**,**'Dadefinire'**);**

**END** **IF;**

**ELSE**

**IF** **(**Abb**=**'mezzafida'**)** **THEN**

**SELECT** CodPassaporto **INTO** Cavallo

**FROM** Cavalli

**WHERE** CentroIppico**=**CentroI **AND** CavalloMezzaFida**(**CodPassaporto**)>**0**;**

**IF** **(**Cavallo**)** **THEN**

**INSERT** **INTO** `Mezzafida`**(**`CodMF`**,** `CodCavallo`**)**

**VALUES** **(**CodUtente**,**Cavallo**);**

**ELSE**

**SELECT** CodPassaporto **INTO** Cavallo

**FROM** Cavalli

**WHERE** CentroIppico**=**CentroI **AND** CavalloSenzaFida**(**CodPassaporto**)>**0**;**

**IF** **(**Cavallo**)** **THEN**

**INSERT** **INTO** `Mezzafida`**(**`CodMF`**,** `CodCavallo`**)**

**VALUES** **(**CodUtente**,**Cavallo**);**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** $

Questa procedura permette di cambiare abbonamento, da allievo senza fida a MezzaFida o Agonistici, dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura vengono passati i parametri CodUtente, codice dell'utente che vuole cambiare abbonamento, Abb, la tipologia del nuovo abbonamento, e CentroI, centro ippico in cui fa equitazione l'allievo.

6) /\* Cambio abbonamento da MezzaFida ad agonistici \*/

**DROP** **PROCEDURE** **IF** **EXISTS** `CaabMFtoAG`**;**

DELIMITER $

**CREATE** **PROCEDURE** CaabMFtoAG**(**CodUtente **CHAR(**16**),** CentroI **VARCHAR(**40**))**

**BEGIN**

**DECLARE** Cavallo **CHAR(**15**);**

**DECLARE** Cod **CHAR(**16**);**

**SELECT** CodCavallo **INTO** Cavallo

**FROM** MezzaFida

**WHERE** CodMF**=**CodUtente**;**

**DELETE** **FROM** `MezzaFida`

**WHERE** CodMF**=**CodUtente**;**

**SELECT** CodUtente **INTO** Cod

**FROM** MezzaFida

**WHERE** CodCavallo**=**Cavallo**;**

**IF** **(NOT** Cod**)** **THEN**

**INSERT** **INTO** `Agonistici`**(**`CodAg`**,** `CodCavallo`**,** `Patente`**)**

**VALUES** **(**CodUtente**,**Cavallo**,**'Dadefinire'**);**

**ELSE**

**SELECT** CodPassaporto **INTO** Cavallo

**FROM** Cavalli

**WHERE** CentroIppico**=**CentroI **AND** CavalloSenzaFida**(**CodPassaporto**)>**0**;**

**IF** **(**Cavallo**)** **THEN**

**INSERT** **INTO** `Agonistici`**(**`CodAg`**,** `CodCavallo`**,** `Patente`**)**

**VALUES** **(**CodUtente**,**Cavallo**,**'Dadefinire'**);**

**END** **IF;**

**END** **IF;**

**END** $

Questa procedura permette di cambiare abbonamento, da allievo in mezza fida ad allievo agonista, dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura vengono passati i parametri CodUtente, codice dell'utente che vuole cambiare abbonamento, e CentroI, centro ippico in cui fa equitazione l'allievo.

7) /\* Cambio abbonamento da Agonistici a MezzaFida \*/

**DROP** **PROCEDURE** **IF** **EXISTS** `CaabAGtoMF`**;**

DELIMITER $

**CREATE** **PROCEDURE** CaabAGtoMF**(**CodUtente **CHAR(**16**))**

**BEGIN**

**DECLARE** Cavallo **CHAR(**15**);**

**SELECT** CodCavallo **INTO** Cavallo

**FROM** Agonistici

**WHERE** CodAg**=**CodUtente**;**

**DELETE** **FROM** `Agonistici`

**WHERE** CodAg**=**CodUtente**;**

**INSERT** **INTO** `Mezzafida`**(**`CodMF`**,** `CodCavallo`**)**

**VALUES** **(**CodUtente**,**Cavallo**);**

**END** $

Questa procedura permette di cambiare abbonamento, da allievo agonista ad allievo in mezza fida, dopo che è stata accettata la relativa richiesta da parte dell'amministratore autorizzato. Alla procedura viene passato il parametro CodUtente, codice dell'utente che vuole cambiare abbonamento.

TRIGGERS

/\* Trigger che gestisce le richieste \*/

**DROP** **TRIGGER** **IF** **EXISTS** ConfermaRichiesta**;**

DELIMITER $

**CREATE** **TRIGGER** ConfermaRichiesta

**BEFORE** **UPDATE** **ON** Richieste

**FOR** **EACH** **ROW**

**BEGIN**

**DECLARE** Codice **CHAR(**16**);**

**DECLARE** CentroI **VARCHAR(**40**);**

**SELECT** CentroIppico **INTO** CentroI

**FROM** Allievi

**WHERE** CodAllievo**=OLD.**CodUtente**;**

**IF (OLD.**DataConvalida **IS NOT NULL)** **THEN**

**IF** **(OLD.**TipoRichiesta**=**'Cambio orario'**)** **THEN**

**CALL** CambioO**(OLD.**Descrizione**,OLD.**CodUtente**);**

**ELSE**

**IF** **(OLD.**TipoRichiesta**=**'Iscrizione'**)** **THEN**

**CALL** Iscrizione**(OLD.**Descrizione**,OLD.**CodUtente**);**

**ELSE**

**IF** **(OLD.**TipoRichiesta**=**'Aggiorna patente'**)** **THEN**

**CALL** AggPat**(OLD.**Descrizione**,OLD.**CodUtente**);**

**ELSE**

**IF** **(OLD.**TipoRichiesta**=**'Cambio abbonamento'**)** **THEN**

**SELECT** CodAg **INTO** Codice **FROM** Agonistici **WHERE** CodAg**=OLD.**CodUtente**;**

**IF** **(**Codice **IS** **NULL)** **THEN**

**SELECT** CodMF **INTO** Codice **FROM** MezzaFida **WHERE** CodMF**=OLD.**CodUtente**;**

**IF** **(**Codice **IS** **NULL)** **THEN**

**IF** **(OLD.**Descrizione**=**'mezzafida'**)** **THEN**

**CALL** CaabMForAG**(OLD.**CodUtente**,**'mezzafida'**,**CentroI**);**

**ELSE**

**IF** **(OLD.**Descrizione**=**'agonistici'**)** **THEN**

**CALL** CaabMForAG**(OLD.**CodUtente**,**'agonistici'**,**CentroI**);**

**END** **IF;**

**END** **IF;**

**ELSE**

**IF** **(OLD.**Descrizione**=**'allievi'**)** **THEN**

**CALL** CaabAllievi**(OLD.**CodUtente**,**'mezzafida'**);**

**ELSE**

**IF** **(OLD.**Descrizione**=**'agonistici'**)** **THEN**

**CALL** CaabMFtoAG**(OLD.**CodUtente**,**CentroI**);**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**ELSE**

**IF** **(OLD.**Descrizione**=**'allievi'**)** **THEN**

**CALL** CaabAllievi**(OLD.**CodUtente**,**'agonistici'**);**

**ELSE**

**IF** **(OLD.**Descrizione**=**'mezzafida'**)** **THEN**

**CALL** CaabAGtoMF**(OLD.**CodUtente**);**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END IF;**

**END** $

Questo trigger gestisce le richieste. Quando un amministratore conferma una richiesta, a seconda della tipologia, viene chiamata una procedura che esegue il compito desiderato. Nel caso in cui la richiesta venga rifiutata la condizione OLD.DataConvalida IS NOT NULL sarà falsa e quindi l'esecuzione del tigger terminerà senza apportare alcuna modifica alle tabelle coinvolte nella richiesta.

/\* Trigger che controlla l'età di un utente che si vuole registrare \*/

**DROP** **TRIGGER** **IF** **EXISTS** ControlloEta**;**

DELIMITER $

**CREATE** **TRIGGER** ControlloEta

**BEFORE** **INSERT** **ON** UtentiRegistrati

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **(YEAR(**CURDATE**())-YEAR(NEW.**DataNascita**)<**8**)** **THEN**

**INSERT** **INTO** `Errori` **(**`CodErrore`**,**`Descrizione`**)**

**VALUE** **(NULL,NULL);**

**ELSE**

**IF** **(MONTH(**CURDATE**())>MONTH(NEW.**DataNascita**)&&YEAR(**CURDATE**())-YEAR(NEW.**DataNascita**)=**7**)** **THEN**

**INSERT** **INTO** `Errori` **(**`CodErrore`**,**`Descrizione`**)**

**VALUE** **(NULL,NULL);**

**ELSE**

**IF(DAY(**CURDATE**())>DAY(NEW.**DataNascita**)&&**

**MONTH(**CURDATE**())=MONTH(NEW.**DataNascita**)&&**

**YEAR(**CURDATE**())-YEAR(NEW.**DataNascita**)=**7**)** **THEN**

**INSERT** **INTO** `Errori` **(**`CodErrore`**,**`Descrizione`**)**

**VALUE** **(NULL,NULL);**

**END** **IF;**

**END** **IF;**

**END** **IF;**

**END** $

Questo trigger ha lo scopo di non permettere agli utenti con età minore dei sette anni di registrarsi. Solo gli utenti che hanno un età superiore possono, infatti. diventare allievi di un centro ippico. Se l'utente ha meno di sette anni il trigger crea un errore fittizio con CodErrore (chiave primaria) NULL. In questo modo l'inserimento non va a buon fine.

**INTERFACCIA WEB**

L'applicazione web è strutturata come segue:

- in alto è posizionato l'header con il titolo del progetto;

- a sinistra è presente un menu verticale;

- di fianco al menu si trova il frame in cui vengono visualizzate le pagine a cui l'utente ha accesso.

Il menu cambia a seconda dell'utente che ha effettuato il login. Il menu di base, quello visibile a tutti, prevede delle pagine di visualizzazione di informazioni relative ai centri ippici, ai cavalli ed ai concorsi. A seconda che l'utente sia un semplice utente registrato, un allievo, un agonista, un istruttore, uno stalliere o un amministratore il menu permetterà di accedere a diverse funzionalità. L'unico utente ad avere accesso in lettura e scrittura alla maggior parte delle tabelle è l'amministratore che però gestisce solo i dati relativi al centro ippico di appartenenza e non ha accesso a nessuna informazione riguardante gli altri centri (tranne che per le informazioni raggiungibili tramite i link presenti nel menu di base).

INFORMAZIONI SULL'AUTENTICAZIONE

L'autenticazione di ogni utente viene effettuata tramite l'utilizzo di sessioni. Ad ogni login, dopo aver inserito e-mail di registrazione e password corrette, vengono inizializzate le variabili di sessione utente e permessi che contengono rispettivamente il codice fiscale dell'utente e la tipologia di utente che ha effettuato il login (utenti registrati, allievi, mezzafida, agonistici, stallieri, istruttori, amministratori). Queste variabili di sessione permettono di abilitare o meno determinate funzioni in tutte le pagine in cui vengono utilizzate.

GERARCHIA DEI FILE DI CODICE DELL'INTERFACCIA WEB

La gerarchia principale dei file utilizzati per creare l'interfaccia web è la seguente:

* index.html
* chisiamo.html
* vis\_Cavalli.php
* vis\_CentriIppici.php
* iconcorsi.php
* login.php
* registration.php
* utentiregistrati.php
* vis\_Cavalli.php
* vis\_CentriIppici.php
* iconcorsi.php
* richieste.php
* visualizzaprofilo.php
* modificaprofilo.php
* logout.php
* mezzafida.php
* vis\_Cavalli.php
* vis\_CentriIppici.php
* iconcorsi.php
* richieste.php
* orarimiei.php
* visualizzaprofilo.php
* modificaprofilo.php
* logout.php
* agonistici.php
* vis\_Cavalli.php
* vis\_CentriIppici.php
* iconcorsi.php
* richieste.php
* imieiconcorsi.php
* orarimiei.php
* visualizzaprofilo.php
* modificaprofilo.php
* logout.php
* istruttori.php
* vis\_Cavalli.php
* vis\_CentriIppici.php
* iconcorsi.php
* richieste.php
* orari.php
* visualizzaprofilo.php
* modificaprofilo.php
* logout.php
* amministratori.php
* vis\_Cavalli.php
* vis\_CentriIppici.php
* concamm.php
* amministra.php
* visualizza.php
* richiesteamm.php
* agg\_rim dip.php
* agg\_rim cavallo.php
* agg\_rim concorso.php
* vittoriaconcorso.php
* assicurazioni.php
* visualizzaprofilo.php
* modificaprofilo.php
* logout.php

DESCRIZIONE DELLE PAGINE DELL'INTERFACCIA WEB

* index.html

È la pagina iniziale visibile a qualsiasi utente (registrato e non). Permette di accedere alla pagine di login e di regstrazione. Inoltre, con il menu di navigazione verticale posizionato sulla sinistra, permette agli utenti di visualizzare le informazioni di base relative ai centri ippici, ai cavalli ed ai concorsi.

Le pagine amministratori.php, istruttori.php, agonistici.php, mezzafida.php e utentiregistrati.php sono pagine simili a index.html con la differenza che hanno un menu di navigazione verticale differente a seconda dei permessi dell'utente.

* chisiamo.php

Questa pagina descrive la società proprietaria dei centri ippici.

* vis\_Cavalli.php

Questa pagina permette a tutti gli utenti (registrati e non) di visualizzare le informazioni relative ai cavalli presenti nel database. L'utente può decidere di visualizzare tutti i cavalli oppure può visualizzare i cavalli di uno specifico centro ippico.

* vis\_CentriIppici.php

Questa pagina permette a qualsiasi utente di visualizzare le informazioni relative ai centri ippici presenti nel database.

* iconcorsi.php

La pagina iconcorsi.php permette di visualizzare i concorsi ai quali i centri ippici hanno aderito. L'utente può decidere se visualizzarli tutti o se visualizzarli in base al periodo di svolgimento.

* login.php

Questa pagina permette all'utente registrato di accedere alla propria area riservata. In base al tipo di utente vengono inizializzate le variabili di sessione che permetteranno all'utente di navigare nelle sole aree a lui dedicate.

* registration.php

Questa pagina permette all'utente non registrato di registrarsi.

* richieste.php

Questa pagina permette a tutti gli utenti registrati di fare richieste all'amministratore del proprio centro ippico (o del centro ippico in cui voglioni iscriversi).

* visualizzaprofilo.php

Questa pagina permette ad un utente di visualizzare il proprio profilo.

* modificaprofilo.php

modificaprofilo.php permette all'utente registrato di modificare i dati del proprio profilo. Affinchè l'operazione vada a buon fine l'utente deve inserire la propria password per motivi di sicurezza.

* logout.php

Questa pagina serve a disconnettere l'utente e di tornare all'homepage. In questa pagine vengono rilasciate le variabili di sessione.

* orarimiei.php

Questa pagina permette ad ogni allievo di visualizzare il proprio orario di lezione.

* imieiconcorsi.php

imieiconcorsi.php è una pagina riservata degli utenti agonistici e permette loro di visualizzare i concorsi ai quali hanno partecipato e di iscriversi ad un altro concorso. L'operazione andrà a buon fine solo se l'agonista ha inserito il codice della propria patente nel database.

* orari.php

Questa pagina è riservata agli istruttori e permette loro di visualizzare il proprio orario di lavoro.

* amministra.php

amministra.php è una pagina riservata all'amministratore. È il pannello di controllo da cui può accedere alle pagine di visualizzazione, modifica, inserimento e cancellazione dei dati relativi al centro ippico di cui è dipendente.

* visualizza.php

Permette all'amministratore di visualizzare codici e altre informazioni riguardo tutti gli utenti e i cavalli associati al proprio database.

* concamm.php

Pagina simile a oconcorsi.php, ma riservata agli amministratori. In questa pagina infatti vengono visualizzati anche i codici relativi ai concorsi.

* richiesteamm.php

richiesteamm.php è la pagina relativa alle richieste degli utenti che l'amministratore può accettare o rifiutare.

* agg\_rim dip.php

Pagina che permette all'amministratore di aggiungere o rimuovere un dipendente.

* agg\_rim cavallo.php

Pagina che permette all'amministratore di aggiungere o rimuovere un cavallo.

* agg\_rim concorso.php

Pagina che permette all'amministratore di aggiungere o rimuovere un concorso. L'amministratore può scegliere se aggiunere un concorso interno o un concorso esterno.

* vittoriaconcorso.php

vittoriaconcorso.php ha lo scopo di aggiornare la tabella Partecipazioni inserendo il vincitore di un concorso terminato.

* assicurazioni.php

La pagina assicurazioni.php visualizza tutte le assicurazioni che scadono entro 14 giorni. In questo modo l'amministratore potrà avvisare in tempo tutti gli allievi ai quali l'assicurazione sta per scadere. Inoltre permette all'amministratore di aggiornare la scadenza di una assicurazione della quale sa essere stata pagata la corrispondente quota.

Altre pagine a cui può accedere un utente sono:

* aggiornapatente.php, caab.php, cambiaoraio.php, colloquio.php e iscrizione.php sono le pagine che permettono ad un utente registrato di aggiungere una richiesta.

Piu' precisamente aggiornapatente.php permette ad un agonista di richiedere l'aggiornamento della propria patente. caab.php permette ad un allievo di fare richiesta di cambiare abbonamento. cambioorario.php permette ad un allievo di fare richiesta di cambiare l'orario delle lezioni. colloquio.php permette ad un dipendente o un utente registrato di fare richiesta di colloquio con l'amministratore del centro ippico desiderato. Infine, iscrizione.php permette ad un utente non ancora iscritto a nessuna lezione di fare richiesta di iscrizione ad un centro ippico.

* creacitta.php

È' una pagina che permette l'inserimento di una citta non presente nel database qualora fosse necessario per una nuova registrazione o per la modifica di un profilo utente.

* dataora.php

È' una pagina che permette di visualizzare la data e l'ora corrente nel menu a sinistra delle pagine.