Relazione sul progetto Basi di Dati

Burlin Valerio, matr. 1029442

ABSTRACT

Il progetto modella la gestione di una squadra di ciclismo. In particolare vengono gestite le informazioni anagrafiche riguardo i ciclisti e lo staff della squadra, le iscrizioni dei corridori alle gare a cui la squadra intende partecipare, le eventuali vittorie parziali di tappa ed i piazzamenti finali dei corridori nelle gare terminate. Le operazioni tipiche sono la creazione e modifica di record anagrafici di ciclisti e staff, aggiunta di nuove corse ed iscrizione dei corridori in esse, inserimento dei risultati a competizione conclusa.

1 ANALISI DEI REQUISITI

Il progetto intende realizzare una base di dati e relativa interfaccia web per la gestione di una squadra di ciclismo. L'entità principale che interessa modellare sono le corse a cui la squadra partecipa. Di una corsa interessa il nome, la categoria, la data di inizio e fine, i corridori della squadra iscritti a tale corsa, il luogo di partenza e quello di arrivo della corsa e la lunghezza. Inoltre una corsa può essere formata da due o più tappe intermedie.

Di una tappa interessa il numero d'ordine all'interno della corsa, il luogo di partenza e di arrivo, la data di svolgimento e la lunghezza. Inoltre di una tappa interessa sapere se è stata vinta da un corridore della squadra.

Una corsa può essere già conclusa o ancora da correre. Di una corsa da correre interessa sapere se l'iscrizione della squadra a tale corsa è stata approvata.

Di un luogo interessa sapere il CAP, che lo identifica, il nome e le relative nazione, regione ed eventuale provincia di appartenenza.

Le persone fisiche della squadra si dividono in staff e ciclisti. Di una persona interessa nome e cognome, il codice identificativo all'interno della squadra, che la identifica, la nazionalità, l'anno di nascita e il recapito formato dall'indirizzo e dal numero di telefono. Dei membri dello staff interessa sapere il ruolo, se è allenatore o direttore sportivo(DS). Ai membri dello staff è associata inoltre una password per autenticarsi nel sistema.

Dei ciclisti interessa l'anno in cui sono diventati professionisti e l'anno di arrivo nella squadra. Inoltre interessa sapere se hanno ottenuto piazzamenti nei primi 10 alle corse a cui partecipano. Ai ciclisti è assegnato un passaporto biologico, obbligatorio per poter partecipare alle competizioni, contenente i valori ematici standard dell'atleta, utili per il confronto a seguito di controlli antidoping che un atleta può subire durante la stagione.

Di un passaporto biologico interessa sapere i valori dei globuli rossi(RBC), dell'emoglobina(Hb) e dell'ematocrito(Hct).

Le operazione usuali sono la creazione, eliminazione o modifica delle entità qui sopra descritte. Si possono individuare tre diverse tipologie di utenti: utenti non autenticati nel sito, che possono solo effettuare ricerche su informazioni, risultati e iscrizioni alle corse dei corridori; utenti autenticati non amministratori, ossia gli allenatori, che possono solo aggiungere, rimuovere o sostituire dalle liste di iscrizione alle corse non ancora approvate i corridori; utenti autenticati e amministratori, i DS, che possono eseguire qualunque operazione.

2 PROGETTAZIONE CONCETTUALE

2.1 LISTA DELLE CLASSI

- PersoneFisiche: modella una persona generica della squadra.
  - Id: *string* <<PK>>*;*
  - Nome: *string;*
  - Cognome: *string;*
  - Nazionalità: *string;*
  - AnnoNascita: *year;*
  - Recapito: seq. [Telefono: *string*, Via: *string,* Num: *int,* Località: *string*, Prov: *string*].

  Sono definite le seguenti sottoclassi disgiunte con vincolo di partizionamento:

  - Ciclisti: rappresenta un ciclista della squadra.
    - TurnProf: *year;*
    - Arrivo: *year.*
  - Staff: rappresenta un membro dello staff della squadra.
    - Ruolo: enum ['DS', 'Allenatore'];
    - pw: *string.*
- Corse: modella una corsa.
  - NomeCorsa: *string;*

- DataInizio: *date;*
- DataFine: *date;*
- Distanza: *float.*

Sono definite le seguenti sottoclassi disgiunte con vincolo di partizionamento:

- Terminate: rappresenta le corse terminate. Non ha nessun attributo, è una specializzazione di Corse.
- Da Correre: rappresenta le corse ancora da correre.
  - Approvata: *bool.*

- Luoghi: modella un luogo.
  - CAP: *string* <<PK>>*;*
  - Località: *string;*
  - Nazione: *string;*
  - Regione: *string;*
  - Prov: *string.*
- Tappe: modella una tappa di una corsa.
  - Numero: *int;*
  - Data: *date;*
  - Distanza: *float.*
- PassBio: modella il passaporto biologico di un corridore.
  - RBC: *float;*
  - Hb: *int;*
  - Hct: *int.*

2.2 LISTA ASSOCIAZIONI

- Ciclisti-PassBio: "AssociatoA".
  - Ogni ciclista può aver associato un passaporto biologico, e ogni passaporto biologico è associato ad un solo ciclista.
  - Molteplicità 1 : 1.
  - Parziale verso PassBio, totale verso Ciclisti.
- Ciclisti-Tappe: "HaVinto".
  - Un ciclista può vincere zero o più tappe, una tappa può essere vinta al più da un ciclista.
  - Molteplicità 1 : N.
  - Parziale verso Ciclisti, parziale verso Tappe.
- Ciclisti-Corse: "IscrittoA".
  - Un ciclista può essere iscritto a zero o più corse, una corsa può avere uno o più ciclisti iscritti.

- Molteplicità M : N.
- Totale verso Ciclisti, parziale verso Corse.
- Attributo:
  - ✓ Piazzamento: *int.*

- Corse-Tappe: "FormataDa".
  - Una corsa può essere formata da zero o più tappe, una tappa appartiene ad una corsa.
  - Molteplicità 1 : N.
  - Parziale verso Tappe, totale verso Corse.
- Tappe-Luoghi: "ParteDa".
  - Una tappa deve partire da un luogo, un luogo può essere sede di partenza di zero o più tappe.
  - Molteplicità N : 1.
  - Parziale verso Tappe, totale verso Luoghi.
- Corse-Luoghi: "ParteDa".
  - Una corsa deve partire da un luogo, un luogo può essere sede di partenza di zero o più corse.
  - Molteplicità N : 1.
  - Parziale verso Corse, totale verso Luoghi.
- Tappe-Luoghi: "ArrivaA".
  - Una tappa deve arrivare in un luogo, un luogo può essere sede di arrivo di zero o più tappe.
  - Molteplicità N : 1.
  - Parziale verso Tappe, totale verso Luoghi.
- Corse-Luoghi: "ArrivaA".
  - Una corsa deve arrivare in un luogo, un luogo può essere sede di arrivo di zero o più tappe.
  - Molteplicità N : 1.
  - Parziale verso Corse, totale verso Luoghi.

2.3 LISTA GERARCHIE

- Gerarchia PersoneFisiche: Staff / Ciclisti.
  - Classi disgiunte.
  - L'unione forma copertura.
- Gerarchia Corse: Terminare / DaCorrere.
  - Classi disgiunte.
  - L'unione forma copertura.

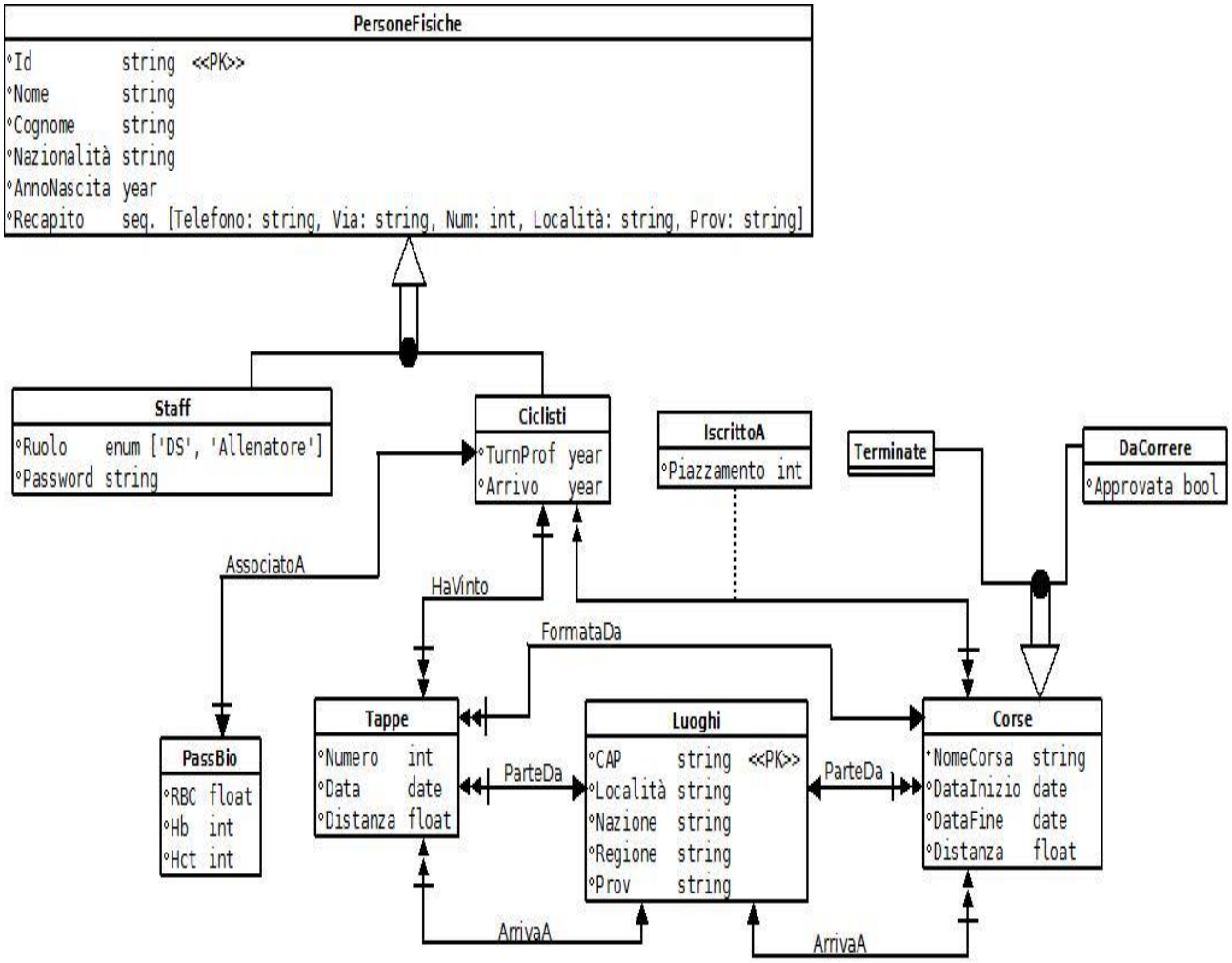

Figura 1: Schema concettuale della base di dati

3 PROGETTAZIONE LOGICA

3.1 GERARCHIE

Nella gerarchia PersoneFisiche è stato scelto il partizionamento verticale in quanto sia Staff che Ciclisti presentano alcuni attributi distinti. La gerarchia viene quindi completata con le seguenti classi:

- PersoneFisiche: *Id* è la chiave primaria per tutte le persone. Si è scelto di non usare il codice fiscale in quanto potrebbe essere sconosciuto e quindi non rappresenterebbe un valore assoluto per identificare univocamente una persona.
  - Id: *string* <<PK>>*;*
  - Nome: *string;*
  - Cognome: *string;*
  - Nazionalità: *string;*
  - AnnoNascita: *year;*

- Recapito: seq. [Telefono: *string*, Via: *string,* Num: *int,* Località: *string*, Prov: *string*].

- **Staff:** Si utilizza *Id* come chiave esterna verso **PersoneFisiche**, che diviene anche chiave primaria della relazione.
  - Id: *int* <<PK>> <<FK(PersoneFisiche);
  - Ruolo: enum [‘DS’, ‘Allenatore’];
  - pw: *string.*
- **Ciclisti:** Si utilizza *Id* come chiave esterna verso **PersoneFisiche**, che diviene anche chiave primaria della relazione.
  - Id: *int* <<PK>> <<FK(PersoneFisiche);
  - TurnProf: *year;*
  - Arrivo: *year.*

Nella gerarchia **Corse** invece è stata scelta la tecnica della tabella unica essendo la tecnica più semplice da implementare ed in quanto **Terminate** non presentava nessun attributo proprio mentre **DaCorrere** ne presentava solo uno, *Approvata,* che viene aggiunto alla nuova classe **Corse.** La situazione diviene la seguente:

- **Corse:** si aggiunge un discriminatore, *Stato*, per distinguere le corse terminate da quelle ancora da correre.
  - NomeCorsa: *string;*
  - DataInizio: *date;*
  - DataFine: *date;*
  - Distanza: *float*;
  - Terminata: *bool*;
  - Approvata: *bool*;

3.2 CHIAVI PRIMARIE

Sono state aggiunte delle chiavi primarie nelle classi **Corse, Tappe** e **PassBio** in modo da poter identificare univocamente una istanza di tali relazioni.

3.3 ASSOCIAZIONI

- **Ciclisti-PassBio**: “AssociatoA”.
  - Ogni ciclista deve aver associato un passaporto biologico, e ogni passaporto biologico è associato ad un solo ciclista.
  - Molteplicità 1 : 1.
  - Parziale verso **PassBio**, totale verso **Ciclisti**.
  - Chiave esterna con vincolo *<<unique>>* su **PassBio** verso **Ciclisti**.
- **Ciclisti-Tappe**: “HaVinto”.

- Un ciclista può vincere zero o più tappe, una tappa può essere vinta al più da un ciclista.
- Molteplicità 1 : N.
- Parziale verso **Ciclisti**, parziale verso **Tappe**.
- Chiave esterna su **Tappe** verso **Ciclisti**.

- **Ciclisti-Corse**: "IscrittoA".
  - Un ciclista può essere iscritto a zero o più corse, una corsa può avere uno o più ciclisti iscritti.
  - Molteplicità M : N.
  - Totale verso **Ciclisti**, parziale verso **Corse**.
  - Attributo:
    - ✓ Piazzamento: *int.*
  - Nuova relazione **CiclistiCorse** con chiavi esterne verso **Ciclisti** e **Corse**:
    - **CiclistiCorse**:
      - Id: *string* <<PK>> <<FK(Ciclisti)>>;
      - IdCorsa: *int* <<PK>> <<FK(Corse)>>;
      - Piazzamento: *int*.
- **Corse-Tappe**: "FormataDa".
  - Una corsa può essere formata da zero o più tappe, una tappa appartiene ad una corsa.
  - Molteplicità 1 : N.
  - Parziale verso **Tappe**, totale verso **Corse**.
  - Chiave esterna con vincolo *<<not null>>* su **Tappe** verso **Corse.**
- **Tappe-Luoghi**: "ParteDa".
  - Una tappa deve partire da un luogo, un luogo può essere sede di partenza di zero o più tappe.
  - Molteplicità N : 1.
  - Parziale verso **Tappe**, totale verso **Luoghi**.
  - Chiave esterna con vincolo *<<not null>>* su **Tappe** verso **Luoghi.**
- **Corse-Luoghi**: "ParteDa".
  - Una corsa deve partire da un luogo, un luogo può essere sede di partenza di zero o più corse.
  - Molteplicità N : 1.
  - Parziale verso **Corse**, totale verso **Luoghi**.
  - Chiave esterna con vincolo *<<not null>>* su **Corse** verso **Luoghi.**
- **Tappe-Luoghi**: "ArrivaA".
  - Una tappa deve arrivare in un luogo, un luogo può essere sede di arrivo di zero o più tappe.
  - Molteplicità N : 1.

- Parziale verso **Tappe**, totale verso **Luoghi**.
- Chiave esterna con vincolo *<<not null>>* su **Tappe** verso **Luoghi.**

- **Corse-Luoghi**: "ArrivaA".
  - Una corsa deve arrivare in un luogo, un luogo può essere sede di arrivo di zero o più tappe.
  - Molteplicità N : 1.
  - Parziale verso **Corse**, totale verso **Luoghi**.
  - Chiave esterna con vincolo *<<not null>>* su **Corse** verso **Luoghi.**

3.4 PROPRIETÀ MULTIVALORE

Per rappresentare la proprietà multivalore *Recapito* nella relazione **PersoneFisiche** si è creata una nuova relazione, chiamata **Recapiti**, con una chiave esterna che fa riferimento alla chiave primaria di **PersoneFisiche** e chiave primaria la concatenazione della chiave esterna e dell'attributo *Recapito.*

- **Recapiti:**
  - Id: *string* <<PK>> <<FK(PersoneFisiche)>>;
  - Recapito: [Telefono: *string*, Via: *string,* Num: *int,* Località: *string*, Prov: *string*] <<PK>>.

3.5 ATTRIBUTI COMPOSTI

Si è successivamente provveduto all'appiattimento dell'attributo composto *Recapito*, ottenendo quindi la seguente relazione, in cui si è assunto che bastino *Località, Via*, *Num*(numero civico) e *Telefono* per determinare univocamente un recapito:

- **Recapiti:**
  - Id: *string* <<PK>> <<FK(PersoneFisiche)>>;
  - Telefono: *string* <<PK>>;
  - Via: *string* <<PK>>;
  - Num: *int* <<PK>>;
  - Località: *string* <<PK>>;
  - Prov: *string.*

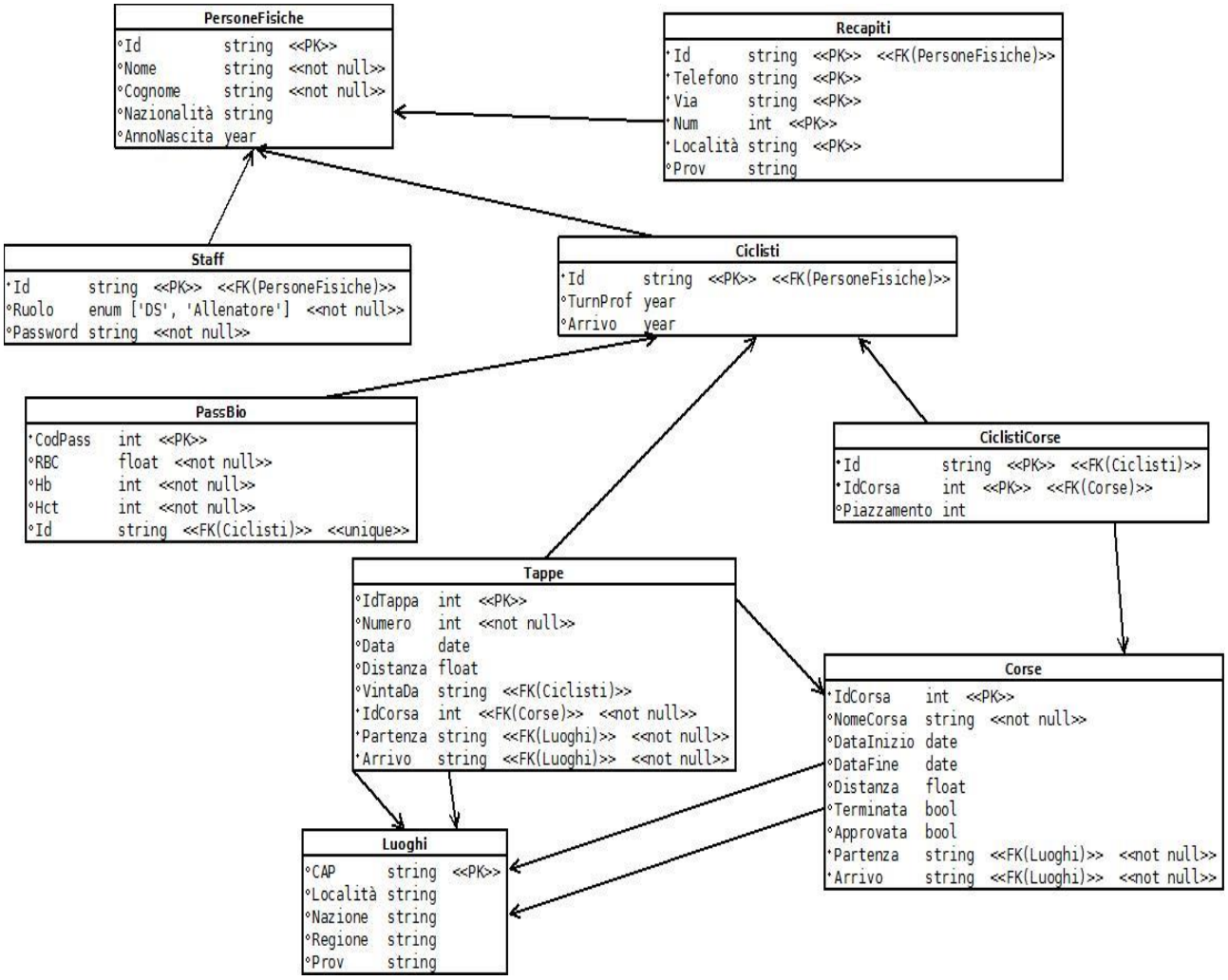

Figura 2: Schema relazionale della base di dati

4 IMPLEMENTAZIONE BASE DI DATI

È stata implementata una tabella aggiuntiva **Errori** contenente le descrizioni degli errori rilevati dai *trigger*.

Il listato seguente implementa quanto illustrato e spiegato precedentemente.

```sql
DROP TABLE IF EXISTS PersoneFisiche;

DROP TABLE IF EXISTS Recapiti;
DROP TABLE IF EXISTS Staff;
DROP TABLE IF EXISTS Ciclisti;
DROP TABLE IF EXISTS PassBio;
DROP TABLE IF EXISTS CiclistiCorse;
DROP TABLE IF EXISTS Corse;
DROP TABLE IF EXISTS Tappe;
DROP TABLE IF EXISTS Luoghi;
DROP TABLE IF EXISTS Errori;

CREATE TABLE PersoneFisiche (
    Id            CHAR(8) PRIMARY KEY,
    Nome          VARCHAR(30) NOT NULL,
    Cognome       VARCHAR(30) NOT NULL,
    Nazionalita   CHAR(3),
    AnnoNascita   YEAR(4)
) ENGINE=InnoDB;

CREATE TABLE Recapiti (
    Id         CHAR(8),
    Telefono   VARCHAR(10),
    Via        VARCHAR(30),
    Num        INT,
    Localita   VARCHAR(30),
    Prov       CHAR(3),
    PRIMARY KEY (Id,Telefono,Via,Num,Localita),
    FOREIGN KEY (Id) REFERENCES PersoneFisiche(Id) ON DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Staff (
    Id        CHAR(8) PRIMARY KEY,
    Ruolo     ENUM('DS','Allenatore') NOT NULL,
    Password  VARCHAR(40) NOT NULL,
    FOREIGN KEY (Id) REFERENCES PersoneFisiche(Id) ON DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Ciclisti (
    Id        CHAR(8) PRIMARY KEY,
    TurnProf  YEAR(4),
    Arrivo    YEAR(4),
    FOREIGN KEY (Id) REFERENCES PersoneFisiche(Id) ON DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE PassBio (
    CodPass  INT AUTO_INCREMENT PRIMARY KEY,
    RBC      FLOAT(1) NOT NULL,
    Hb       INT NOT NULL,
    Hct      INT NOT NULL,
    Id       CHAR(8) UNIQUE,
    FOREIGN KEY (Id) REFERENCES Ciclisti(Id) ON DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Luoghi (

    CAP       CHAR(5) PRIMARY KEY,
    Localita  VARCHAR(30),
    Nazione   CHAR(3),
    Regione   VARCHAR(30),
    Prov      CHAR(2)
) ENGINE=InnoDB;
```

```sql
CREATE TABLE Luoghi (

    CAP       CHAR(5) PRIMARY KEY,
    Localita  VARCHAR(30),
    Nazione   CHAR(3),
    Regione   VARCHAR(30),
    Prov      CHAR(2)
) ENGINE=InnoDB;

CREATE TABLE Corse (
    IdCorsa     INT AUTO_INCREMENT PRIMARY KEY,
    NomeCorsa   VARCHAR(40) NOT NULL,
    DataInizio  DATE,
    DataFine    DATE,
    Distanza    FLOAT(1),
    Terminata   TINYINT(1),
    Approvata   TINYINT(1),
    Partenza    CHAR(5) NOT NULL,
    Arrivo      CHAR(5) NOT NULL,
    FOREIGN KEY (Partenza) REFERENCES Luoghi(CAP) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (Arrivo) REFERENCES Luoghi(CAP) ON UPDATE CASCADE ON DELETE
CASCADE
) ENGINE=InnoDB;

CREATE TABLE Tappe (
    IdTappa   INT AUTO_INCREMENT PRIMARY KEY,
    Numero    INT NOT NULL,
    Data      DATE,
    Distanza  FLOAT(1),
    VintaDa   CHAR(8),
    IdCorsa   INT NOT NULL,
    Partenza  CHAR(5),
    Arrivo    CHAR(5),
    FOREIGN KEY (VintaDa) REFERENCES Ciclisti(Id) ON DELETE SET NULL,
    FOREIGN KEY (IdCorsa) REFERENCES Corse(IdCorsa) ON DELETE CASCADE,
    FOREIGN KEY (Partenza) REFERENCES Luoghi(CAP) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (Arrivo) REFERENCES Luoghi(CAP) ON UPDATE CASCADE ON DELETE
CASCADE
) ENGINE=InnoDB;

CREATE TABLE CiclistiCorse (
    Id           CHAR(8),
    IdCorsa      INT,
    Piazzamento  INT,
    PRIMARY KEY (Id,IdCorsa),
    FOREIGN KEY (Id) REFERENCES Ciclisti(Id) ON DELETE CASCADE,
    FOREIGN KEY (IdCorsa) REFERENCES Corse(IdCorsa) ON DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Errori (
    CodiceErrore  INT PRIMARY KEY,
    Descrizione   VARCHAR(255)
) ENGINE=InnoDB;
INSERT INTO Errori(CodiceErrore,Descrizione) VALUES
       (1,'Il piazzamento deve riguardare uno dei primi 10 posti.'),
       (2,'Corsa non ancora disputata');
```

4.1 TRIGGER

Sono stati implementati due *trigger*:

- **SetTerminata:** Trigger attivo che dopo aver aggiornato il piazzamento in una corsa, imposta tale corsa come *Terminata*.
- **ControlloPiazzamentiData:** Trigger passivo che se si cerca di inserire un piazzamento di un ciclista in una corsa ancora da correre, si attiva prima di tale inserimento e genera un errore inserendo una chiave duplicata nella tabella **Errori**.

```
DELIMITER $

DROP TRIGGER IF EXISTS SetTerminata $
CREATE TRIGGER SetTerminata
AFTER UPDATE ON CiclistiCorse
FOR EACH ROW
BEGIN
    IF NEW.Piazzamento IS NOT NULL
      THEN UPDATE Corse SET Terminata = 1 WHERE IdCorsa = NEW.IdCorsa;
    END IF;
END $


DROP TRIGGER IF EXISTS ControlloPiazzamentiData $
CREATE TRIGGER ControlloPiazzamentiData
BEFORE UPDATE ON CiclistiCorse
FOR EACH ROW
BEGIN
    IF NEW.Piazzamento IS NOT NULL AND CURDATE() < (SELECT c.DataFine
                                                    FROM Corse c
                                                WHERE c.IdCorsa = NEW.IdCorsa )
      THEN INSERT INTO Errori(CodiceErrore) VALUE(2);
    END IF;
END $
```

4.2 FUNZIONI

Sono state implementate due *funzioni*:

- **HaTappe:** Riceve come input l'Id di una Corsa e restituisce 1(Vero) se tale corsa è formata da Tappe, altrimenti 0(False).
- **ContaVittorieParziali:** Riceve come input l'Id di un ciclista e conta, restituendo tale risultato, il numero di tappe totali vinte dal ciclista.

```
DROP FUNCTION IF EXISTS HaTappe $

CREATE FUNCTION HaTappe (corsa INT) RETURNS TINYINT
BEGIN
    DECLARE conta INT DEFAULT 0;
    DECLARE flag TINYINT(1) DEFAULT 0;

    SELECT COUNT(*) INTO conta
    FROM Tappe
    WHERE IdCorsa = corsa;

    IF conta > 0
      THEN SET flag = 1;
    END IF;

    RETURN flag;
END $


DROP FUNCTION IF EXISTS ContaVittorieParziali $
CREATE FUNCTION ContaVittorieParziali (ciclista CHAR(8)) RETURNS INT
BEGIN
    DECLARE vit INT DEFAULT 0;

    SELECT COUNT(*) INTO vit
    FROM Tappe t JOIN Ciclisti c ON (t.VintaDa = c.Id)
    WHERE c.Id = ciclista;

    RETURN vit;
END $
```

4.3 PROCEDURE

Sono state definite 3 *Procedure*:

- InserisciStaff: fa sì che l'inserimento congiunto dei dati nelle tabelle PersoneFisiche-Staff avvenga come una singola transazione.
- InserisciCiclisti: fa sì che l'inserimento congiunto dei dati nelle tabelle PersoneFisiche-Ciclisti avvenga come una singola transazione.
- InserisciTappa: Inserisce correttamente una tappa col proprio IdCorsa a partire dal nome della Corsa.

```
DELIMITER $


DROP PROCEDURE IF EXISTS InserisciStaff $
CREATE PROCEDURE InserisciStaff (IN Id CHAR(8), Nome VARCHAR(30), Cognome
VARCHAR(30), Nazionalita CHAR(3), AnnoNascita YEAR(4), Ruolo
ENUM('DS','Allenatore'), Password VARCHAR(40) )
BEGIN
    START TRANSACTION;
    INSERT INTO PersoneFisiche(Id,Nome,Cognome,Nazionalita,AnnoNascita)
    VALUES (Id,Nome,Cognome,Nazionalita,AnnoNascita);
    INSERT INTO Staff(Id,Ruolo,Password)
    VALUES (Id,Ruolo,Password);
    COMMIT;
END $


DROP PROCEDURE IF EXISTS InserisciCiclisti $
CREATE PROCEDURE InserisciCiclisti (IN Id CHAR(8), Nome VARCHAR(30), Cognome
VARCHAR(30), Nazionalita CHAR(3), AnnoNascita YEAR(4), TurnProf YEAR(4),
Arrivo YEAR(4) )
BEGIN
    START TRANSACTION;
    INSERT INTO PersoneFisiche(Id,Nome,Cognome,Nazionalita,AnnoNascita)
    VALUES (Id,Nome,Cognome,Nazionalita,AnnoNascita);
    INSERT INTO Ciclisti(Id,TurnProf,Arrivo)
    VALUES (Id,TurnProf,Arrivo);
    COMMIT;
END $


DROP PROCEDURE IF EXISTS InserisciTappa $
CREATE PROCEDURE InserisciTappa (IN Numero INT, Data DATE, Distanza FLOAT(1),
NomeCorsa VARCHAR(40), Partenza CHAR(5),
                                    Arrivo CHAR(5) )
BEGIN
    DECLARE IdCorsa INT;
    SELECT c.IdCorsa INTO IdCorsa
      FROM Corse c
     WHERE c.NomeCorsa=NomeCorsa;
    INSERT INTO Tappe(Numero,Data,Distanza,IdCorsa,Partenza,Arrivo)
    VALUES (Numero,Data,Distanza,IdCorsa,Partenza,Arrivo);
END $

DELIMITER ;
```

5 QUERY

Seguono alcune query significative.

1. Crea una vista, *view*, in cui vengono visualizzati i Nomi e Cognomi dei corridori che hanno vinto almeno una Tappa o una Corsa, il numero di vittorie di Tappa totali ed il numero di vittorie complessivo(vittorie di Tappa più vittorie Finali).

```
DROP VIEW IF EXISTS Vittorie;
CREATE VIEW Vittorie(Id,Nome,Cognome,VittorieParziali,VittorieTotali) AS
SELECT p.Id,p.Nome,p.Cognome,ContaVittorieParziali(p.Id), COUNT(*) +
ContaVittorieParziali(p.Id)
FROM PersoneFisiche p NATURAL JOIN CiclistiCorse cc
WHERE cc.Piazzamento = 1
GROUP BY p.Nome,p.Cognome
UNION
SELECT p1.Id,p1.Nome,p1.Cognome,ContaVittorieParziali(p1.Id), COUNT(*)
FROM PersoneFisiche p1 JOIN Tappe t ON (p1.Id=t.VintaDa);
WHERE p1.Id NOT IN (
                     SELECT p2.Id
                     FROM PersoneFisiche p2 NATURAL JOIN CiclistiCorse cc1
                     WHERE cc1.Piazzamento = 1
                    )
GROUP BY p1.Nome,p1.Cognome;
```

OUTPUT:

Id	Nome	Cognome	VittorieParziali	VittorieTotali
dvllella	Davide	Villella	0	1
eviviani	Elia	Viviani	3	4
ptrsagan	Peter	Sagan	0	2
admarchi	Alessandro	De Marchi	1	1

2. Restituisce il nome di una corsa a Tappe vinta da un ciclista ed il nome e cognome e l'anno di passaggio al professionismo del vincitore.

```
SELECT p.Nome,p.Cognome,c.TurnProf AS ProfessionistaDal,co.NomeCorsa
FROM PersoneFisiche p, Ciclisti c, CiclistiCorse cc, Corse co
WHERE p.Id=c.Id AND c.Id=cc.Id AND cc.IdCorsa=co.IdCorsa
      AND co.DataInizio <> co.DataFine AND cc.Piazzamento=1;
```

OUTPUT:

```
+--------+-------------+----------------------+-----------------------------------------------------+
| Nome | Cognome | ProfessionistaDal | NomeCorsa                                                |
+--------+-------------+----------------------+-----------------------------------------------------+
| Elia    | Viviani     |          2010      | Settimana Internazionale Coppi e Bartali  |
+--------+-------------+----------------------+-----------------------------------------------------+
```

3. Seleziona il nome e cognome dei corridori che si sono piazzati in una corsa che ha sede di partenza o arrivo uguale alla provincia del recapito del corridore, e ritorna anche informazioni riguardo la corsa e il piazzamento ottenuto in tale corsa.

```
SELECT p.Nome,p.Cognome,r.Prov AS ProvDiProvenienza, cc.Piazzamento,
       co.NomeCorsa, l1.Localita AS Partenza, l1.Prov AS ProvPartenza,
       l2.Localita AS Arrivo, l2.Prov AS ProvArrivo
FROM PersoneFisiche p, Recapiti r, CiclistiCorse cc, Corse co, Luoghi l1,
      Luoghi l2
WHERE p.Id = r.Id AND p.Id = cc.Id AND cc.IdCorsa = co.IdCorsa AND
      cc.Piazzamento <> 0 AND co.Partenza = l1.CAP AND co.Arrivo = l2.CAP
      AND (r.Prov = l1.Prov OR r.Prov = l2.Prov);
```

OUTPUT:

```
+---------+-------------+-----------------------+----------------+---------------------+-----------+-----------------+-----------+--------------+
| Nome  | Cognome  | ProvDiProvenienza | Piazzamento | NomeCorsa       | Partenza | ProvPartenza | Arrivo    | ProvArrivo |
+---------+-------------+-----------------------+----------------+---------------------+-----------+-----------------+-----------+--------------+
| Davide | Villella     | MI                    |              5 | Milano Sanremo | Milano    | MI              | Sanremo | IM          |
+---------+-------------+-----------------------+----------------+---------------------+-----------+-----------------+-----------+--------------+
```

4. Seleziona gli Id, Nomi e Cognomi dei ciclisti che hanno partecipato a più di una corsa nel mese di Febbraio nell'anno 2014.

```
SELECT Id,Nome,Cognome

FROM PersoneFisiche
WHERE Id IN (
            SELECT c.Id
            FROM Ciclisti c JOIN CiclistiCorse cc ON (c.Id=cc.Id)
                            JOIN Corse co ON (co.IdCorsa=cc.IdCorsa)
            WHERE EXTRACT(YEAR FROM co.DataFine)=2014 AND
                  EXTRACT(MONTH FROM co.DataFine)=02
            GROUP BY c.Id
            HAVING COUNT(*) > 1
          );
```

OUTPUT:

```
+------------+-----------+-------------+
| Id         | Nome      | Cognome     |
+------------+-----------+-------------+
| dformolo   | Davide    | Formolo     |
| dvllella   | Davide    | Villella    |
| oscgatto   | Oscar     | Gatto       |
| ptrsagan   | Peter     | Sagan       |
+------------+-----------+-------------+
```

5. Selezione il nome e il cognome dei ciclisti che non hanno ottenuto nessun piazzamento nelle corse che hanno disputato.

```
SELECT p.Nome,p.Cognome
FROM PersoneFisiche p NATURAL JOIN Ciclisti c
WHERE NOT EXISTS(
          SELECT *
          FROM Ciclisti c1 NATURAL JOIN CiclistiCorse cc
          WHERE p.Id=c1.Id AND cc.Piazzamento <> 0
                );
```

OUTPUT:

```
+---------------+--------------+
| Nome          | Cognome      |
+---------------+--------------+
| Alessandro    | De Marchi    |
| Damiano       | Caruso       |
| Oscar         | Gatto        |
+---------------+--------------+
```

6. Seleziona nome e cognome dei corridori vincitori di tappe che non hanno ottenuto piazzamenti in nessuna corsa.

```
SELECT p.Nome,p.Cognome
FROM PersoneFisiche p JOIN Ciclisti c ON (p.Id=c.Id)
                      JOIN Tappe t ON (c.Id=t.VintaDa)
WHERE p.Id NOT IN (
           SELECT p1.Id
           FROM PersoneFisiche p1 NATURAL JOIN Ciclisti c1
           WHERE EXISTS(
                        SELECT *
                        FROM Ciclisti c2 NATURAL JOIN CiclistiCorse cc
                        WHERE p1.Id=c2.Id AND cc.Piazzamento > 1
                        )
           );
```

OUTPUT:

```
+----------------+---------------+
| Nome         | Cognome   |
+----------------+---------------+
| Alessandro | De Marchi |
+----------------+---------------+
```

6 INTERFACCIA WEB

L'interfaccia web utilizza sia pagine di servizio che effettuano operazioni sulla BD sia pagine di interfaccia vere e proprie per l'interazione ed interrogazione della BD.

6.1 PAGINE DI INTERFACCIA

Ci sono 5 pagine principali per la navigazione nel sito web, la pagina **home.php** che presenta semplicemente una foto della squadra e il menù di navigazione a sinistra, le pagine **squadra.php** e **corse.php** che presentano in forma tabellare i dati relativi a Staff, Ciclisti e Corse della squadra, la pagina **ricerca.php** che permette di effettuare delle interrogazioni sui risultati dei ciclisti nelle varie corse e la pagina **ins_login.php** che presenta il form per l'autenticazione nell'area protetta.

Le tabelle presenti in *squadra.php* e *corse.php* hanno, per ogni riga, un hyperlink "info" che passa l'Id di un membro dello staff, un ciclista o una corsa, come querystring che viene recuperato attraverso il metodo GET nelle pagine **infociclista.php**, **infocorsa.php** e **infostaff.php**. Tali pagine mostrano dati specifici cu ciclisti, corse e staff.

All'interno di *infocorsa.php* si trova una tabella identica per visualizzare, se presenti, le tappe di una corsa, il cui funzionamento è identico alle precedenti e l'hyperlink invoca la pagina **infotappa.php** con l'Id della tappa desiderata.

6.2 AREA RISERVATA

I DS e Allenatori che accedono all'area riservata, oltre alla normale navigazione, hanno altre funzionalità a disposizione. Nelle tabelle sono presenti altri due hyperlink, "modifica" ed "elimina", i quali rispettivamente chiamano le pagine per effettuare modifiche sulla BD(**mod_corsa.php**, **mod_ciclista.php**, **mod_staff.php**, **mod_tappa.php**) ed eliminare entità sulla BD(**elimina_ciclista.php**, **elimina_corsa.php**, **elimina_staff.php**, **elimina_tappa.php**). Le pagine di modificha utilizzano sempre il passaggio dell'Id specifico tramite querystring e presentano un form con aree di testo contenenti i dati dell'entità che possono essere modificati ed aggiornati premendo l'apposito pulsante. Le pagine di cancellazione semplicemente cancellano il record

dalla BD e ritornano nella stessa pagina da cui si sta eseguendo la cancellazione tramite la funzione php *header()*. Oltre agli hyperlink si hanno a disposizione alcuni pulsanti, alcuni disponibili solo per i DS altri per entrambe le categorie di persone che possono accede al sito. Tali pulsanti sono realizzati tramite piccole form con passaggio di parametri con metodo GET e quelli indicati con "_hid" utilizzano anche degli hidden input per passare valori significativi alla pagina che viene invocata dal pulsante.

I pulsanti comuni alle 2 categorie sono quelli per chiamare le pagine per inserire o modificare le iscrizioni dei ciclisti alle corse e per inserire e/o modificare piazzamenti finali e vittorie di tappa. Tali funzionalità sono rese disponibili tramite le pagine ins_iscritti.php, mod_iscritti.php, ins_piazzamento.php, ins_vittoria.php, che presentano form per l'inserimento/modifica di tali dati.

I DS hanno a disposizione altri pulsanti per inserire nuovi ciclisti corse e tappe tramite le pagine add_corsa.php, add_ciclista.php, add_tappa.php, add_staff.php e un pulsante per confermare una corsa in attesa di approvazione, compito svolto invocando la pagina di servizio conferma_corsa.php che aggiorna correttamente la tabella Corse della BD e ritorna nella pagina *corse.php*.

6.3 AUTENTICAZIONE

La pagina ins_login.php presenta un form per inserire username e password. Una volta inserite invoca una funzione che controlla che il login corrisponda ad un membro dello staff, recupera la password attraverso il metodo di decodifica SHA1 e infine controlla che login e password siano quelli corretti. Se l'autenticazione va a buon fine si viene reindirizzati nella Home e nell'header del sito comparirà la data dell'ultimo accesso autenticato effettuato da un membro dello staff tramite un Cookie. Se invece si presenta qualche errore, esso viene segnalato nella form di inserimento. Oltre a verificare username e password, tale pagina, se l'autenticazione va a buon fine setta anche le corrette variabili di sessione che aiutano nella navigazione e rendono disponibili funzionalità e pulsanti corretti al membro dello staff che sta effettuando l'accesso.

[Le password con relativo login per l'accesso sono:

-rbamadio: sge4nt2y

-szanatta: sutnb48s

-mrscirea: tbsc6kpb]

6.4 PAGINE DI SERVIZIO

Oltre alle già citate pagine per eliminare record sulla BD e la conferma delle corse, sono presenti altre pagine che servono solo a svolgere determinate funzioni ma non vengono visualizzate. **logout.php** si occupa dell'uscita dall'area riservata eliminando le variabili di sessioni precedentemente impostate, **connessione.php** si occupa di effettuare la connessione con la BD e viene inclusa in tutte le pagine che ne richiedano l'interazione, **errori.php** contiene due funzioni per la gestione degli errori di connessione alla BD. La pagina **funzioni.php** contiene un vasto insieme di funzioni utilizzate dalle altre pagine, e per questo è inclusa in tutte le altre pagine. Sono presenti funzioni per la stampa di specifiche parti di codice HTML, funzioni per stampare a video le tabelle nel formato e coi i link desiderati e dati specifici di ciclisti,corse,staff e tappe, funzioni di aiuto nell'autenticazione(*get_pwd($login,$conn)*) e che controllano per ogni pagina se l'utente è autenticato o meno(*autenticato()*), funzioni che controllano la consistenza dei dati che si vogliono inserire nella BD e infine funzioni che vanno a effettuare modifiche nella BD.