Università degli Studi di Trieste
Ingegneria Elettronica e Informatica

Progetto di Basi di Dati
IN0500619
"Trofeo Annuale Assoluto di Atletica Leggera"

Giberna Marco

6 giugno 2021

Indice

Capitolo 1

Presentazione Progetto

Un ente sportivo organizza ogni anno in uno stadio un trofeo di atletica leggera. Il torneo si svolge in un giorno. Gli stadi sono identificati dal nome, e si conosce il loro indirizzo (città, via, civico), e il numero di posti.
Al trofeo possono partecipare atleti maggiorenni, maschi e femmine, tesserati ad una società sportiva. L'ente vuole tenere traccia del nome, cognome, codice atleta, data e luogo di nascita, telefono e email di ogni atleta, e le informazioni delle corrispettive società, quali nome, sede, email, telefono, codice società e opzionalmente il sito web.
Le gare del torneo verranno effettuate ad un dato orario ed hanno un codice proprio, sono divise in gare maschili, per gli atleti di sesso maschile, e femminili, per le atlete di sesso femminile. Di ogni gara si possiede il risultato di ogni atleta e la posizione d'arrivo.
Gli atleti possono partecipare a più gare di discipline diverse nel torneo. Ogni gara ha un solo giudice designato, tesserato all'ordine dei giudici delle gare di atletica leggera. Dei giudici si sa nome, cognome, data di nascita, luogo di nascita, numero di telefono, email, codice tesserino giudice; questi dati sono stati forniti dalla federazione giudici atletica leggera.
Inoltre l'ente permette la visione del trofeo agli spettatori che hanno acquistato il biglietto. I biglietti, oltre ad avere un codice progressivo univoco, sono nominativi, e sono acquistabili tramite il loro sito web, il quale al momento dell'acquisto raccoglie i seguenti dati: nome, cognome, numero di telefono, email, codice fiscale e l'indirizzo. Ogni acquirente può acquistare più di un biglietto.

Azioni

- La FIDAL chiede i risultati tutti gli atleti che hanno partecipato alle gare nell'ultima edizione

  (1 volta l'anno)

- L'ente vuole verificare i biglietti venduti e i dati degli acquirenti

  (2 volte al mese)

- L'ente vuole ottenere il numero di biglietti venduti per l'ultima edizione

  (1 volta l'anno)

- L'ente vuole ottenere i dati degli atleti partecipanti per inviare mail promozionali

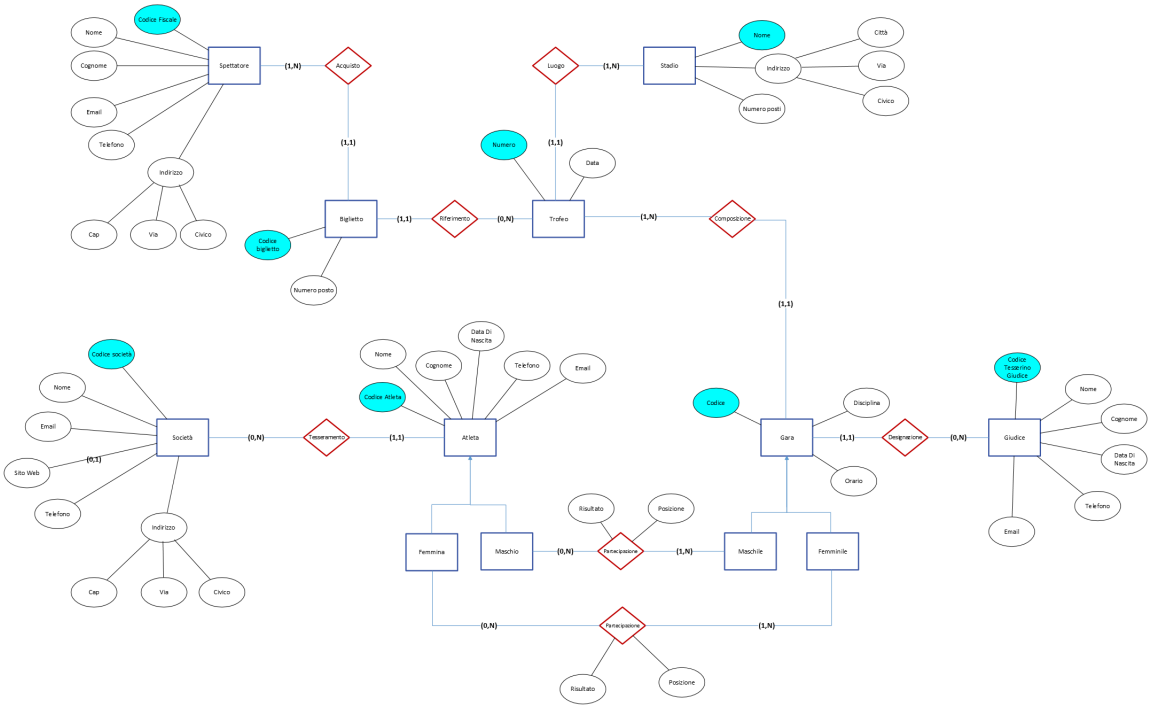
  (1 volta al mese)

- L'ente vuole ottenere i dati delle società sportive con almeno 10 atleti iscritti alle gare per inviare mail promozionali

  (1 volta al mese)

Capitolo 2

Schema Entity-Relationship



Nota: sono stati evidenziati in azzurro degli identificatori.

Capitolo 3

Dizionario dei Dati

entita'	**descrizione**	**attributi**	**identificatore**
società	società sportiva di atletica leggera	codice società, nome, email, telefono, indirizzo, sito web	codice società
atleta	atleti iscritti al trofeo	codice atleta, nome, cognome, data di nascita, telefono, email	codice atleta
gara	gara reale, appartiene ad un edizione del trofeo	codice, orario, disciplina	codice
giudice	giudici tesserati	codice tesserino giudice, nome, cognome, data di nascita, telefono, email	codice tesserino giudice
trofeo	edizione del trofeo	numero, data	numero
stadio	luogo in cui un torneo può avvenire	nome, via, numero posti	nome
biglietto	biglietto	codice biglietto, numero posto	codice biglietto
spettatore	persona che ha acquistato almeno un biglietto	nome, cognome, CF, email, telefono, indirizzo	CF

Tabella 3.1: Dizionario dei Dati - Entità.

relazione	descrizione	composizione	attributi
tesseramento	tesseramento di un atleta ad una società sportiva	codice società, codice atleta	-
partecipazione	partecipazione di un atleta ad una gara	codice atleta, codice gara, risultato, posizione	risultato, posizione
designazione	designazione di un giudice ad una gara	codice gara, codice giudice	-
composizione	gara appartenente ad una specifica edizione del torneo	codice gara, edizione torneo	-
luogo	luogo dell'edizione trofeo	edizione trofeo, nome stadio	-
riferimento	edizione del torneo a cui un biglietto si riferisce	codice biglietto, edizione trofeo	-
acquisto	biglietto acquistato da uno specifico spettatore	codice biglietto, CF spettatore	-

Tabella 3.2: Dizionario dei Dati - Relazioni.

Capitolo 4

Vincoli Non Esprimibili

Dall'analisi del testo del problema emergono i seguenti vincoli non esprimibili:

- Gli atleti devono essere maggiorenni per poter partecipare alle competizioni.
- Non si possono vendere più biglietti del numero di posti complessivi dello stadio in cui si svolge l'edizione del trofeo.

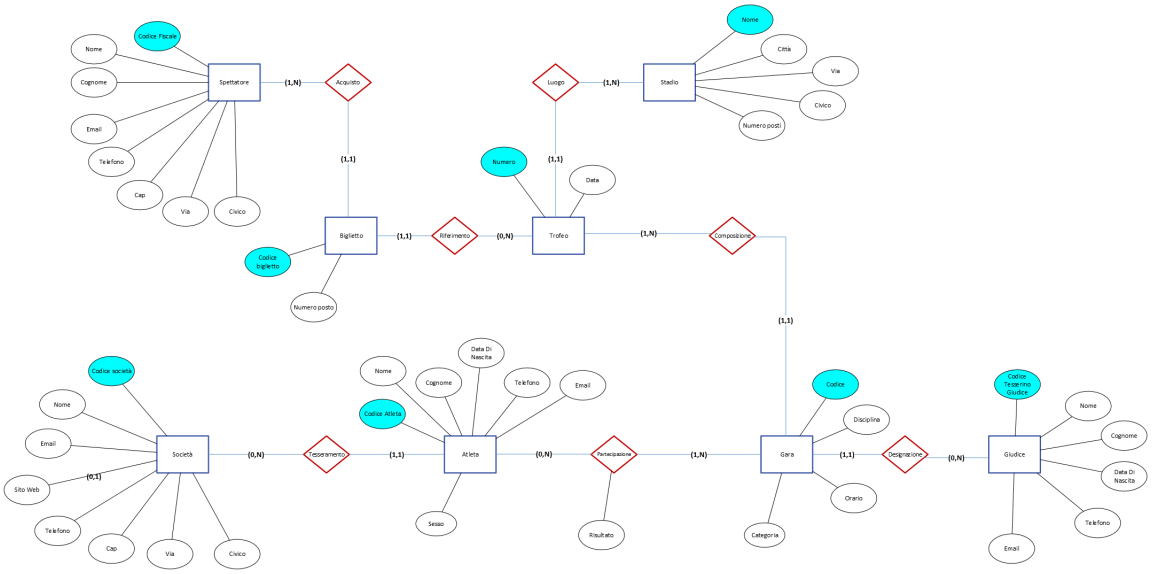
Capitolo 5

Tabella dei Volumi

CONCETTO	TIPO	VOLUME
società	E	20
tesseramento	R	1000
atleta	E	1000
partecipazione	R	2000
gara	E	200
designazione	R	200
giudice	E	100
composizione	R	200
trofeo	E	10
luogo	R	10
stadio	E	10
riferimento	R	20000
biglietto	E	20000
acquisto	R	20000
spettatore	E	10000

Capitolo 6

Schema Entity-Relationship Ristrutturato



Eliminazione generalizzazioni

Per quanto riguarda l'entità atleta e l'entità gara, entrambe generalizzazioni, si è scelto di accorpare i figli ai genitori, fondendo le due relazioni che univano le specializzazioni.
Ciò è stato fatto in quanto da un'analisi delle azione richieste sulla base di dati risulta che gli accessi a tale entità sono contestuali. In particolare non sono richiesti con frequenza i records di una sola specializzazione piuttosto che l'altra.

Le specializzazioni diventano così rispettivamente l'attributo "sesso" nell'entità atleta e l'attributo "categoria" nell'entità gara.
Questa scelta implica la necessità di andare ad inserire un vincolo (Trigger) che permetta le relazioni solo tra entità in cui l'attributo "sesso" corrisponde con "categoria".

Analisi ridondanze

La relazione "partecipazione" presenta l'attributo "posizione". Questo risulta essere un attributo calcolabile. In seguito all'analisi delle azioni da eseguire sulla base di dati, in particolare per il fatto che vengono periodicamente richiesti solo i "risultati" degli atleti, risulta dimostrarsi un appesantimento non necessario per il sistema.
Si sceglie quindi di eliminare l'attributo posizione, anche in ottica di mantenere la Terza Forma Normale.

Eliminazione attributi composti

Vengono inoltre scomposti gli attributi composti in attributi semplici, come "indirizzo".

Scelta Identificatori Primari

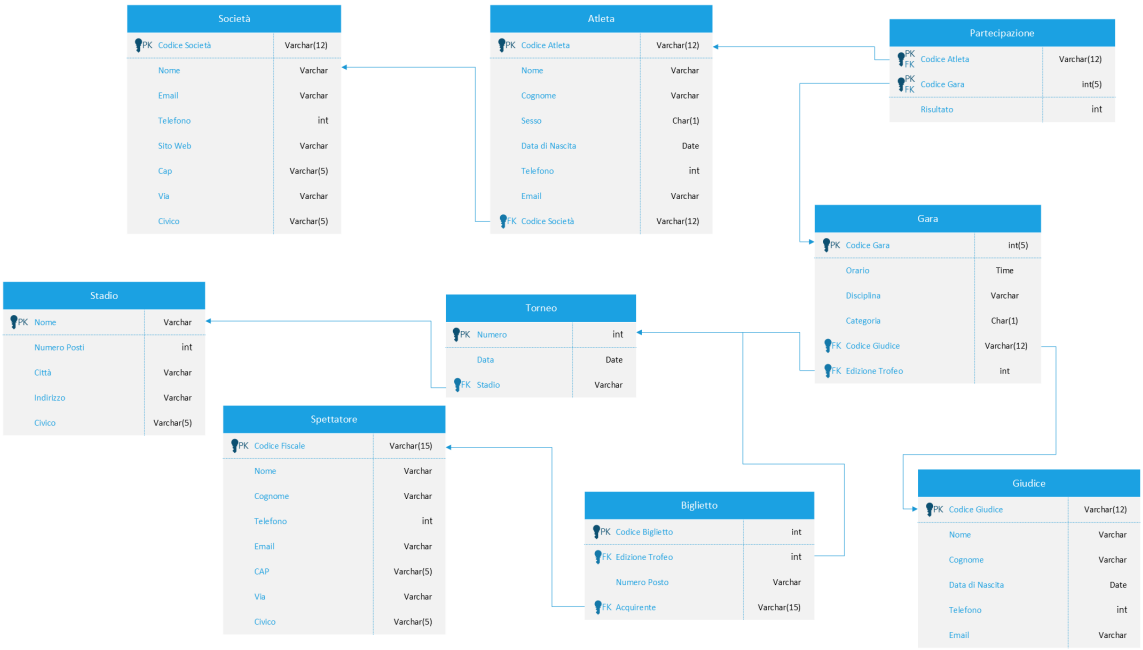
Per ogni entità è stato evidenziato in azzurro un identificatore primario. Rispettivamente, sono stati scelti:

- società: il codice società;
- atleta: il codice atleta;
- gara: il codice della gara;
- giudice: il codice tesserino giudice;
- trofeo: il numero dell'edizione;
- stadio: il nome, che dopo un'attenta ricerca è risultato essere un attributo univoco per l'entità stadio;

- biglietto: il codice biglietto (matrice);
- spettatore: il codice fiscale.

Capitolo 7

Schema Logico



- Società(<u>Codice Società</u>, Nome, Email, Telefono, Sito Web, Cap, Via, Civico)

- Atleta(<u>Codice Atleta</u>, Nome, Cognome, Sesso, Data di Nascita, Telefono, Email, Codice Società)

- Partecipazione(<u>Codice Atleta</u>, <u>Codice Gara</u>, Risultato)

- Gara(Codice Gara, Orario, Disciplina, Categoria, Codice Giudice, Edizione Trofeo)

- Giudice(Codice Giudice, Nome, Cognome, Data di Nascita, Telefono, Email)

- Torneo(Numero, Data, Stadio)

- Stadio(Nome, Numero Posti, Città, Via, Civico)

- Biglietto(Codice Biglietto, Edizione Trofeo, Numero Posto, Acquirente)

- Spettatore(Codice Fiscale, Nome, Cognome, Telefono, Email, Cap, Via, Civico)

Normalizzazione

La base di dati è già in **prima forma normale**, tutte le colonne sono atomiche.
La base di dati è già in **seconda forma normale**, ciascuna colonna dipende dalla primary key.
La base di dati è già in **terza forma normale**, ogni attributo dipende solo dalla primary key.

Capitolo 8

Query SQL per la Realizzazione della Base di Dati

```
CREATE DATABASE IF NOT EXISTS‘trofeoatletica‘;

USE ‘trofeoatletica‘;

DROP TABLE IF EXISTS ‘atleta‘;

CREATE TABLE ‘atleta‘ (
  ‘codiceAtleta‘ varchar(12) NOT NULL,
  ‘nome‘ varchar(50) NOT NULL,
  ‘cognome‘ varchar(50) NOT NULL,
  ‘sesso‘ char(1) NOT NULL,
  ‘dataDiNascita‘ varchar(50) NOT NULL,
  ‘telefono‘ varchar(50) NOT NULL,
  ‘email‘ varchar(50) NOT NULL,
  ‘codiceSocietà‘ varchar(12),
  PRIMARY KEY (‘codiceAtleta‘),
  KEY ‘codiceSocietà‘ (‘codiceSocietà‘) NOT NULL,
  CONSTRAINT ‘atleta_ibfk_1‘ FOREIGN KEY (‘codiceSocietà‘) REFERENCES ‘società‘ (‘codiceSocietà‘)
                                                              ON UPDATE CASCADE,
--  CONSTRAINT ‘atleta_check_age‘ CHECK ((year(‘dataDiNascita‘) < year(now()) - 18)),
--  CONSTRAINT ‘atleta_check_sex‘ CHECK (‘sesso‘ = ’M’ OR ‘sesso‘ = ’F’)
);



DROP TABLE IF EXISTS ‘società‘;

CREATE TABLE ‘società‘ (
  ‘codiceSocietà‘ varchar(12) NOT NULL,
  ‘nome‘ varchar(50) NOT NULL,
  ‘email‘ varchar(50) NOT NULL,
  ‘telefono‘ varchar(50) NOT NULL,
```

```
  `sitoWeb` varchar(50) DEFAULT NULL,
  `cap` varchar(5) NOT NULL,
  `via` varchar(50) DEFAULT NULL,
  `civico` varchar(5) NOT NULL,
  PRIMARY KEY (`codiceSocietà`)
);



DROP TABLE IF EXISTS `gara`;

CREATE TABLE `gara` (
  `codiceGara` int(5) NOT NULL AUTO_INCREMENT,
  `orario` varchar(50) NOT NULL,
  `disciplina` varchar(50) NOT NULL,
  `catogeria` char(1) NOT NULL,
  `codiceGiudice` varchar(50) NOT NULL,
  `edizioneTrofeo` smallint(4) NOT NULL,
  PRIMARY KEY (`codiceGara`),
  KEY `codiceGiudice` (`codiceGiudice`),
  KEY `edizioneTrofeo` (`edizioneTrofeo`),
  CONSTRAINT `gara_ibfk_1` FOREIGN KEY (`codiceGiudice`) REFERENCES `giudice` (`codiceGiudice`)
                                                          ON UPDATE CASCADE,
  CONSTRAINT `gara_ibfk_2` FOREIGN KEY (`edizioneTrofeo`) REFERENCES `trofeo` (`numero`),
--  CONSTRAINT `gara_check_sex` CHECK (`categoria` = 'M' OR `categoria` = 'F')
) ;



DROP TABLE IF EXISTS `giudice`;

CREATE TABLE `giudice` (
  `codiceGiudice` varchar(12) NOT NULL,
  `nome` varchar(50) NOT NULL,
  `cognome` varchar(50) NOT NULL,
  `dataDiNascita` varchar(50) NOT NULL,
  `telefono` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  PRIMARY KEY (`codiceGiudice`)
);



DROP TABLE IF EXISTS `trofeo`;

CREATE TABLE `trofeo` (
  `numero` smallint(4) NOT NULL,
  `data` varchar(50) NOT NULL,
  `luogo` varchar(50) NOT NULL,
  PRIMARY KEY (`numero`),
  KEY `stadio` (`stadio`),
```

```
  CONSTRAINT ‘torneo_ibfk_1‘ FOREIGN KEY (‘stadio‘) REFERENCES ‘stadio‘ (‘nome‘)
);



DROP TABLE IF EXISTS ‘partecipazione‘;

CREATE TABLE ‘partecipazione‘ (
  ‘codiceAtleta‘ varchar(12) NOT NULL,
  ‘codiceGara‘ int(5) NOT NULL,
  ‘risultato‘ varchar(50) NOT NULL,
  PRIMARY KEY (‘codiceAtleta‘, ‘codiceGara‘),
  KEY ‘codiceAtleta‘ (‘codiceAtleta‘),
  KEY ‘codiceGara‘ (‘codiceGara‘),
  CONSTRAINT ‘partecipazione_ibfk_1‘ FOREIGN KEY (‘codiceAtleta‘) REFERENCES ‘atleta‘ (‘codiceAtleta‘)
                                                                    ON UPDATE CASCADE,
  CONSTRAINT ‘partecipazione_ibfk_1‘ FOREIGN KEY (‘codiceGara‘) REFERENCES ‘gara‘ (‘codiceGara‘)
);



DROP TABLE IF EXISTS ‘stadio‘;

CREATE TABLE ‘stadio‘ (
  ‘nome‘ varchar(50) NOT NULL,
  ‘numeroPosti‘ int(6) NOT NULL,
  ‘città‘ varchar(50) NOT NULL,
  ‘indirizzo‘ varchar(50) NOT NULL,
  ‘civico‘ varchar(5) NOT NULL,
  PRIMARY KEY (‘nome‘)
);



DROP TABLE IF EXISTS ‘biglietto‘;

CREATE TABLE ‘biglietto‘ (
  ‘codiceBiglietto‘ int(15) NOT NULL,
  ‘edizioneTrofeo‘ smallint(4) NOT NULL,
  ‘numeroPosto‘ varchar(50) NOT NULL,
  ‘acquirente‘ varchar(15),
  PRIMARY KEY (‘codiceBiglietto‘),
  KEY ‘edizioneTrofeo‘ (‘edizioneTrofeo‘),
  KEY ‘acquirente‘ (‘acquirente‘),
  CONSTRAINT ‘biglietto_ibfk_1‘ FOREIGN KEY (‘edizioneTrofeo‘) REFERENCES ‘trofeo‘ (‘numero‘),
  CONSTRAINT ‘biglietto_ibfk_1‘ FOREIGN KEY (‘acquirente‘) REFERENCES ‘spettatore‘ (‘codiceFiscale‘)
                                                                  ON DELETE SET NULL ON UPDATE CASCADE
);
```

```
DROP TABLE IF EXISTS ‘spettatore‘;

CREATE TABLE ‘spettatore‘ (
  ‘codiceFiscale‘ varchar(15) NOT NULL,
  ‘nome‘ varchar(50) NOT NULL,
  ‘cognome‘ varchar(50) NOT NULL,
  ‘telefono‘ varchar(50) NOT NULL,
  ‘email‘ varchar(50) NOT NULL,
  ‘cap‘ varchar(5) NOT NULL,
  ‘via‘ varchar(50) NOT NULL,
  ‘civico‘ varchar(5) NOT NULL,
  PRIMARY KEY (‘codiceFiscale‘)
 );
```

Capitolo 9

Query SQL Aggiuntive

```
Trigger per controllare che non vengano venduti più biglietti
del numero di posti disponibili allo stadio

DELIMITER $$
CREATE TRIGGER trg_beforeUpdateBiglietti BEFORE UPDATE ON biglietto
FOR EACH ROW
BEGIN
DECLARE numeroBiglietti int(5);
SELECT count(*) INTO numeroBiglietti FROM biglietto b WHERE b.edizioneTrofeo = max(edizioneTrofeo);
IF (numeroBiglietti >= stadio.numeroPosti) THEN
SIGNAL sqlstate '45001' SET message_text = "Basta biglietti, stadio pieno";
END IF;
END$$
DELIMITER ;


Trigger per controllare che un atleta di un certo genere partecipi
alle gare della categoria corrispondente

DELIMITER $$
CREATE TRIGGER trg_beforeUpdatePartecipazione BEFORE UPDATE ON partecipazione
FOR EACH ROW
BEGIN
SET atletaSex = 0;
SET categoriaSex = 0;
SELECT sesso INTO atletaSex FROM atleta WHERE partecipazione.codiceAtleta = atleta.codiceAtleta;
SELECT categoria INTO categoriaSex FROM gara WHERE partecipazione.codiceGara = gara.codiceGara;
IF ( atletaSex <> categoriaSex ) THEN
SIGNAL sqlstate '45002' SET message_text = "Sesso dell'atleta non corrispondente
                                            alla categoria della gara";
END IF;
END$$
DELIMITER ;
```

```
Stored Procedure per ottenere i dati dei biglietti venduti e i
corrispondenti acquirenti in una data edizione del trofeo

DELIMITER $$
CREATE PROCEDURE sp_getBigliettiVendutiConDati(IN numeroEdizione smallint(4) )
BEGIN
SELECT * FROM biglietti
INNER JOIN spettatore ON biglietti.acquirente = spettatore.codiceFiscale
WHERE biglietti.edizioneTrofeo = numeroEdizione;
END$$
DELIMITER ;

SET @edizione = 6;
CALL sp_getBigliettiVendutiConDati( @edizione);


Stored Procedure che ritorna il numero di biglietti venduti in una data edizione del trofeo

DELIMITER $$
CREATE PROCEDURE sp_getNumeroBigliettiVenduti(IN numeroEdizione smallint(4),
                                              OUT numeroBiglietti int(6))
BEGIN
SELECT count(*) INTO numeroBiglietti FROM biglietti
WHERE biglietti.edizioneTrofeo = numeroEdizione;
END$$
DELIMITER ;

SET @edizione = 6;
SET @numeroBiglietti = 0;
CALL sp_getNumeroBigliettiVenduti(@edizione, @numeroBiglietti);


Stored Procedure che mostra i dati di tutti gli atleti ordinati per cognome

DELIMITER $$
CREATE PROCEDURE sp_getDatiAtleta()
BEGIN
SELECT * FROM atleta a ORDER BY a.cognome;
END$$
DELIMITER ;

CALL sp_getDatiAtleta();


Stored Procedure che torna i dati di tutte le società con almeno dieci atleti iscritti al trofeo

DELIMITER $$
CREATE PROCEDURE sp_getDatiSocietàConAlmenoDieciIscritti()
BEGIN
SELECT codiceSocietà, count(*) as numeroIscritti
FROM atleta
```

```
INNER JOIN società USING (codiceSocietà)
GROUP BY codiceSocietà
HAVING numeroIscritti >= 10;
END$$
DELIMITER ;

CALL sp_getDatiSocietàConAlmenoDieciIscritti();
```

Stored Procedure che ritorna i risultati dell'ultima edizione del trofeo

```
DELIMITER $$
CREATE PROCEDURE sp_getRisultatiAtletiUltimoAnno()
BEGIN
SELECT * FROM partecipazione
INNER JOIN atleta ON (codiceAtleta)
INNER JOIN gara ON (codiceGara)
WHERE gara.edizioneTrofeo = max(gara.edizioneTrofeo); -- = (select max(numero) FROM trofeo);
END$$
DELIMITER ;

CALL sp_getRisultatiAtletiUltimoAnno();
```

A seguire Stored Procedure per ottenere i risultati di una gara con la posizione d'arrivo

```
DELIMITER $$
CREATE PROCEDURE sp_getRisultatiGareConPosizioniCrescenti(IN codGara int(5))
BEGIN
SELECT a.nome, a.cognome, a.codiceAtleta, p.risultato, g.disciplina,
ROW_NUMBER() OVER (ORDER BY p.risultato) AS posizione
FROM partecipazione p
INNER JOIN atleta a ON (codiceAtleta)
INNER JOIN gara g ON (codiceGara)
WHERE gara.codiceGara = codGara
ORDER BY risultato;
END$$
DELIMITER ;

SET @codGara = 01234;
CALL sp_getRisultatiGareConPosizioniCrescenti(@codGara);



DELIMITER $$
CREATE PROCEDURE sp_getRisultatiGareConPosizioniDerescenti(IN codGara int(5))
BEGIN
SELECT a.nome, a.cognome, a.codiceAtleta, p.risultato, g.disciplina,
ROW_NUMBER() OVER (ORDER BY p.risultato) AS posizione
FROM partecipazione p
INNER JOIN atleta a ON (codiceAtleta)
```

```
INNER JOIN gara g ON (codiceGara)
WHERE gara.codiceGara = codGara
ORDER BY risultato DESC;
END$$
DELIMITER ;

SET @codGara = 01234;
CALL sp_getRisultatiGareConPosizioniDecrescenti(@codGara);



DELIMITER $$
CREATE PROCEDURE sp_getRisultatiGareConPosizioni(IN codGara int(5))
BEGIN
DECLARE disc VARCHAR(50);
SELECT disciplina INTO disc FROM gara WHERE codiceGara = codGara;
IF(disc LIKE '%lancio%' OR disc LIKE '%salto%') THEN
CALL sp_getRisultatiGareConPosizioniDecrescenti(@codGara);
ELSE
    CALL sp_getRisultatiGareConPosizioniCrescenti(@codGara);
END IF;
END$$
DELIMITER ;

SET @codGara = 01234;
CALL sp_getRisultatiGareConPosizioni(@codGara);
```

Ora una Stored Procedure che fa uso di un cursore per raccogliere in un unica stringa tutte le email degli spettatori, per poterle utilizzare a scopi promozionali o venderle a terzi.

```
DELIMITER $$
CREATE PROCEDURE sp_listaEmail (INOUT listaemail varchar(100000))
BEGIN
    DECLARE finito INTEGER DEFAULT 0;
    DECLARE var_email varchar(100) DEFAULT "";
    DECLARE email_cursor CURSOR FOR SELECT email FROM spettatore;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finito = 1;
    OPEN email_cursor;
    WHILE (finito = 0) DO
        FETCH email_cursor INTO var_email;
        IF finito = 0 THEN
            SET listaemail = CONCAT(var_email, "; ", listaemail);
        END IF;
    END WHILE;
    CLOSE email_cursor;
END$$
DELIMITER ;
```