Progetto di Basi di Dati
-Palestra-

Studente: Francesco Ferrara
Matricola: 0512109789

Indice

1. Raccolta delle specifiche della realtà d'interesse.

Descrizione

Si vuole progettare una base di dati per la gestione di una **palestra**.
L'accesso alla palestra avviene tramite riconoscimento della **card magnetica**,identificata univocamente da un codice: per ottenere la card è necessario sottoscrivere un abbonamento e si vuole conoscere inoltre il numero di accessi.La card è quindi legata sia all'abbonamento che all'atleta.
L'**abbonamento** è identificato da un nome univoco e dal tipo, e si vuole inoltre conoscere il costo.
Gli **atleti** che si iscrivono in palestra (di cui si vuole memorizzare nome, cognome,email e codice fiscale) possono frequentare un numero qualsiasi di corsi e allenarsi nella sala pesi.
Ogni **corso** ha un nome che lo identifica e una descrizione.
I **dipendenti** della palestra si dividono in Istruttori di corso e Personal Trainer.
Dei dipendenti si vuole memorizzare nome,cognome,codice fiscale,retribuzione annuale e email.
Degli **istruttori di corso** si vuole memorizzare il numero dei corsi a carico.Ciascun corso ha un unico istruttore.Un istruttore può insegnare ad un numero qualsiasi di corsi.
I **Personal Trainer** si occupano delle schede di allenamento degli atleti:un atleta può richiedere un numero qualsiasi di schede.Una **scheda** è identificata da un codice e ha come attributo una descrizione.
La palestra dispone di un **gestore** della parte amministrativa di cui si vuole memorizzare nome,cognome,indirizzo,numeri di telefono e codice fiscale.
Il gestore si occupa degli abbonamenti degli utenti e del pagamento dello stipendio ai dipendenti.

Specifiche della realtà d'interesse

La realtà d'interesse riguarda la gestione di una palestra, nella quale è possibile allenarsi in sala pesi o partecipare a dei corsi gestiti da istruttori.
Negli ultimi anni il settore del fitness e più in generale il mondo sportivo è cresciuto molto anche grazie alla spinta data dai social network.
I dati pre pandemia sono chiari: secondo l'European Health & Fitness Market Report 2019 pubblicato da Deloitte e EuropeActive il valore del mercato del fitness in Italia è pari a 2,3 miliardi di euro e le palestre e i circoli sportivi contano 5,5 milioni di iscritti.
Nonostante un calo dovuto al Covid-19 le palestre continuano ad essere il luogo di allenamento preferito dai più appassionati.
È possibile oggi trovarvi corsi di ogni genere, dalle arti marziali come il karate a corsi di ballo di vario tipo come danza classica o breakdance.
Per accedere è necessario sottoscrivere un abbonamento.La validità della card è associata all'abbonamento:quando quest'ultimo scade non è possibile accedere se non viene rinnovato.

L’associato ha quindi la possibilità di scegliere i corsi a cui è interessato fra quelli disponibili.
I personal trainer in sala vanno da 1 a 3 a seconda degli orari e dei giorni lavorativi del singolo dipendente,mentre gli istruttori gestiscono i propri corsi ad orari e giorni prestabiliti.

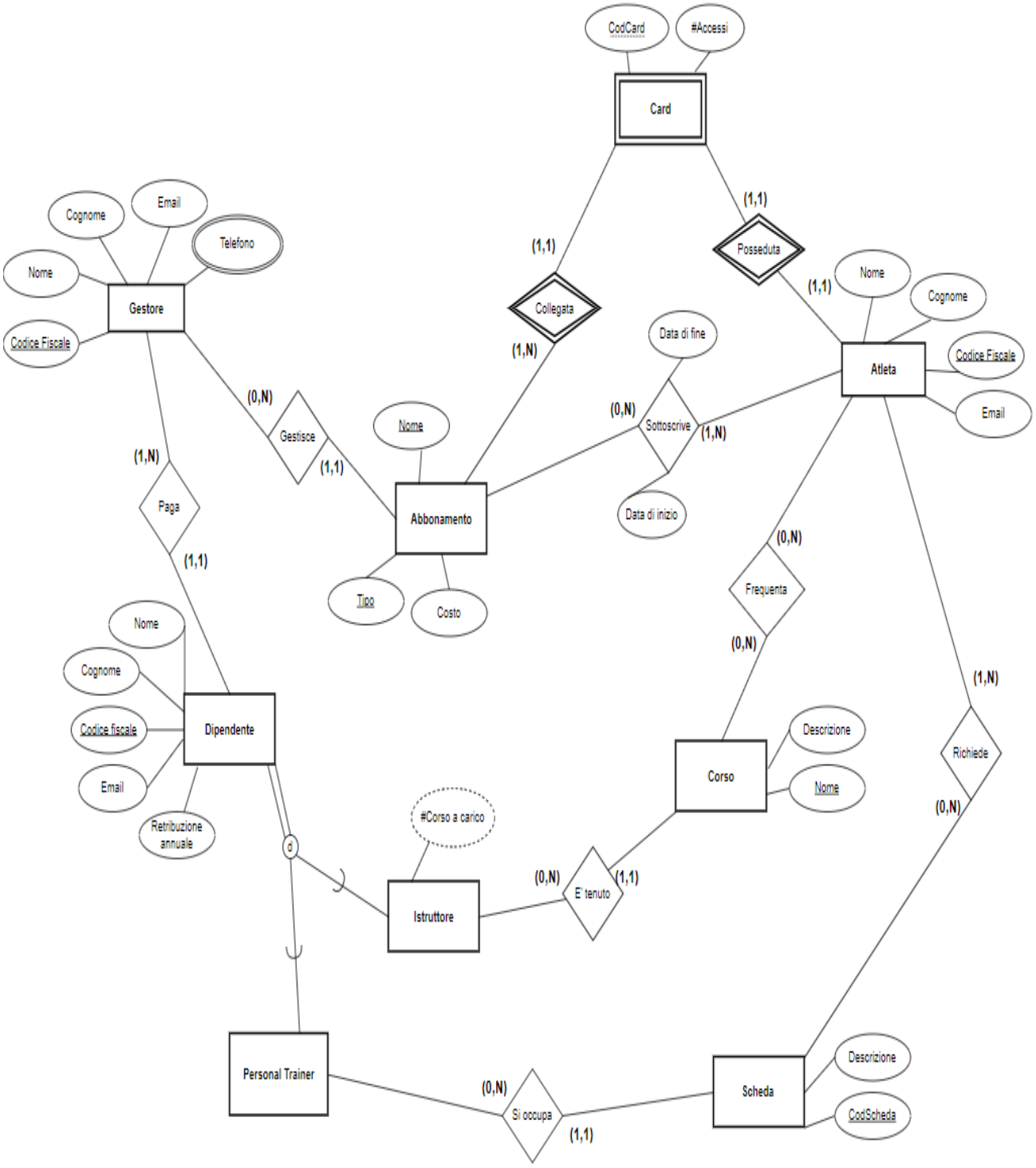
Glossario dei termini

Termine	Significato
Atleta	Persona che si allena in palestra
Abbonamento	Contratto per cui, pagando una determinata somma, si può usufruire di un servizio per un determinato periodo di tempo
Corso	Serie metodica di lezioni frontali finalizzate all’apprendimento di una disciplina e all’allenamento fisico
Dipendente	Figura che si occupa dell’assistenza dei clienti o della gestione della palestra
Personal Trainer	Figura che gestisce la sala pesi e gli atleti che si allenano in essa
Istruttore	Figura che si occupa della gestione di un corso
Gestore	Figura che si occupa di gestire la parte amministrativa della palestra
Card	Tessera che permette l’accesso alla palestra
Scheda	Lista di esercizi che un’atleta deve effettuare

2. Progettazione concettuale della base di dati

Schema EER

Procedendo con la progettazione concettuale della base di dati, si ottiene il seguente schema EER:



Dizionario delle entità

Legenda: sotto-entità, attributo multivalore, attributo ridondante, entità debole, chiave candidata

Entità	Descrizione	Attributi	Identificatore
Atleta	Persona che si allena in palestra	-Codice Fiscale -Nome -Cognome -Email	Codice Fiscale
Abbonamento	Contratto stipulato dall'atleta	-Nome univoco -Tipo -Costo	Nome Univoco Tipo
Card	Tessera necessaria per l'accesso in palestra	-CodCard -#Accessi	CodCard
Corso	Serie di lezioni frontali	-Nome -Descrizione	Nome
Dipendente	Lavoratore presso la palestra	-Codice Fiscale -Nome -Cognome -Retribuzione annuale -Email	Codice Fiscale
Personal Trainer	Figura che si occupa della sala pesi		/
Istruttore	Figura che si occupa della gestione di un corso	-#Corso a carico	/
Gestore	Figura che si occupa della parte amministrativa	-Codice Fiscale -Nome -Cognome -Telefono -Email	Codice Fiscale
Scheda	Programma di allenamento	-CodScheda -Descrizione	CodScheda

Dizionario delle relazioni

Relazione	Descrizione	Entità coinvolte	Attributi
Sottoscrive	Un atleta sottoscrive un abbonamento	Atleta(1,N) Abbonamento(0,N)	-Data inizio -Data fine
Collegata	Una card è collegata ad un abbonamento	Card(1,1) Abbonamento(1,N)	/
Posseduta	Un atleta possiede una card personale	Card(1,1) Atleta(1,1)	/
Gestisce	Il gestore gestisce gli abbonamenti	Gestore(0,N) Abbonamento(1,1)	/
Paga	Il gestore paga i dipendenti della palestra	Gestore(1,N) Dipendente(1,1)	/
Si occupa	Il personal trainer si occupa delle schede degli atleti	Personal Trainer(0,N) Scheda(1,1)	/
È tenuto	Un istruttore tiene dei corsi	Corso(1,1) Istruttore(0,N)	/
Frequenta	Un atleta frequenta un corso	Atleta(0,N) Corso(0,N)	/
Richiede	Un atleta richiede una scheda	Atleta(1,N) Scheda(0,N)	/

Vincoli non esprimibili nello schema

Oltre ciò che è deducibile dallo schema EER, si tenga conto dei seguenti **vincoli**:

- L'attributo "costo" dell'entità Abbonamento deve avere valore compreso
- fra 0 e 250;
- CodCard,CodScheda e i vari codici fiscali devono avere un valore alfanumerico ;
- Le date sono del tipo anno-mese-giorno(aaa-mm-gg);
- L'attributo "descrizione" dell'entità Corso e dell'entità Scheda deve avere valore minore o uguale a 3000.

3. Definizione delle procedure per la gestione della base di dati

Tavola dei volumi

Definiamo di seguito la tavola dei volumi della base di dati.

Concetto	Tipo	Carico Applicativo
Atleta	E	200
Abbonamento	E	10
Card	E	300
Corso	E	5
Dipendente	E	6
Personal Trainer	E	3
Istruttore	E	3
Gestore	E	1
Scheda	E	50
Posseduta	R	200
Collegata	R	150
Sottoscrive	R	600
Gestisce	R	5
Paga	R	6
Si occupa	R	60
È tenuto	R	6
Frequenta	R	150
Richiede	R	600

Tavola delle operazioni

Definiamo di seguito la tavola delle operazioni per la gestione dei dati memorizzati nella base di dati.

	Operazione	Tipo	Frequenza
1	Registrare un nuovo abbonamento	I	3/m
2	Inserire un nuovo atleta	I	10/m
3	Aggiungere un nuovo dipendente	I	1/a
4	Aggiungere un nuovo corso	I	1/a
5	Aggiungere una nuova scheda	I	3/m
6	Inserire una nuova richiesta di una scheda	I	3/m
7	Inserire un nuovo accesso di un atleta	I	12/m
8	Stampare nome,cognome e retribuzione annuale di tutti i dipendenti	B	1/a
9	Trovare il cognome degli istruttori che insegnano a più di un corso	B	3/a
10	Visualizzare il numero di atleti di un dato corso	B	2/m
11	Visualizzare il numero di atleti che hanno richiesto più di 2 schede	B	2/a
12	Leggi il numero di accessi di uno specifico atleta	I	2/m
13	Visualizzare il numero di corsi a carico di un istruttore	I	2/m
14	Ordinare gli abbonamenti di tutti gli iscritti per data di scadenza	B	5/m
15	Selezionare nome e cognome dei Personal Trainer che si sono occupati di almeno 3 schede	B	2/m
16	Selezionare nome e cognome degli iscritti che frequentano almeno un corso	B	4/m
17	Selezionare l'email di tutti gli atleti che non hanno sottoscritto un determinato tipo di abbonamento	B	2/m
18	Selezionare l'email di tutti gli atleti che hanno una determinata scheda	B	3/m

4. Progettazione logica

Analisi delle ridondanze

Il dato ridondante è l'attributo "corso a carico" dell'entità Istruttore.Infatti, sarebbe possibile ottenere il nome e quindi il numero di corsi a carico attraverso il conto delle partecipazioni di un determinato Istruttore nella relazione "Corso è tenuto da Istruttore". Supponendo che l'attributo abbia un peso di 2 byte e considerato che il volume dell'entità Istruttore è uguale a 3, il dato andrebbe ad occupare uno spazio totale di circa 6 byte. Per decidere se mantenere o meno il dato ridondante è necessario calcolare, per le operazioni che lo coinvolgono, la differenza nel numero di accessi con e senza quest'ultimo.

Tavola degli accessi

Operazione 4

Calcolo con ridondanza				Calcolo senza ridondanza			
Concetto	Costrutto	Accesso	Tipo	Concetto	Costrutto	Accesso	Tipo
Corso	E	1	S	Corso	E	1	S
E' tenuto	R	2	S	E' tenuto	R	2	S
Istruttore	E	2	L	Frequenta	R	30	S
Istruttore	E	2	S	Atleta	E	30	S
Frequenta	R	30	S				
Atleta	E	30	S				
Totale: 65S * 1/a * 2 + 2L * 1/a= 132/a				Totale: 63S * 1/a * 2= 126/a			

Operazione 9

Calcolo con ridondanza				Calcolo senza ridondanza			
Concetto	Costrutto	Accesso	Tipo	Concetto	Costrutto	Accesso	Tipo
Istruttore	E	1	L	Istruttore	E	1	L
				E' tenuto	R	2	L
Totale: 1L * 3/a= 3/a				Totale: 3L * 3/a= 9/a			

Operazione 13

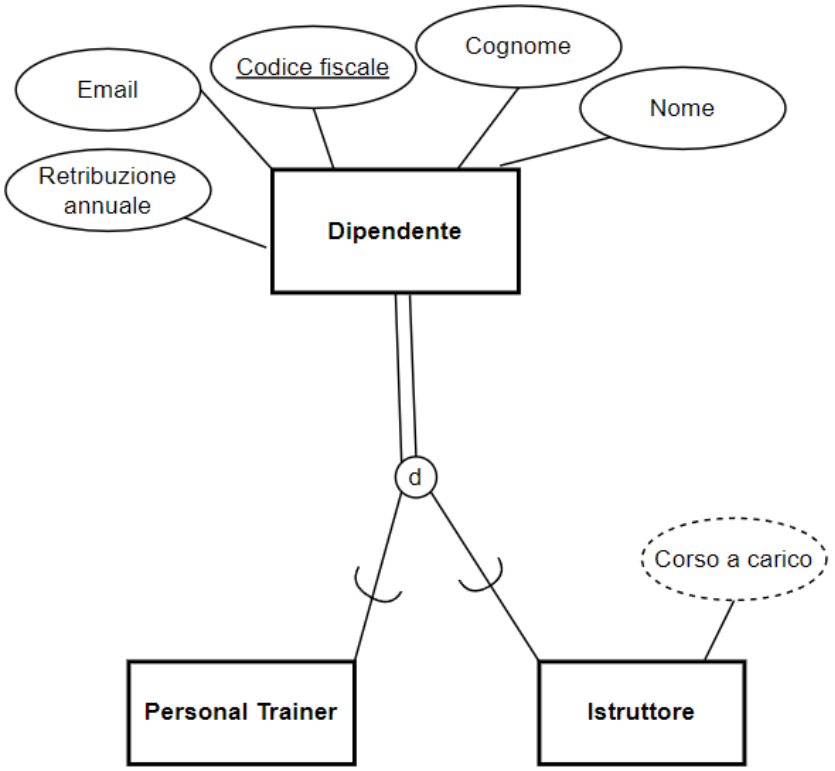
Calcolo con ridondanza				Calcolo senza ridondanza			
Concetto	Costrutto	Accesso	Tipo	Concetto	Costrutto	Accesso	Tipo
Istruttore	E	1	L	Istruttore	E	1	L
				E' tenuto	R	2	L
Totale: 1L * 2/m= 2/m * 12= 24/a				Totale: 3L * 2/m= 6/m * 12=72/a			

Totale con ridondanza = (132 + 3 +24)/a = 159/a
Totale senza ridondanza = (126 + 97 + 72)/a = 295/a

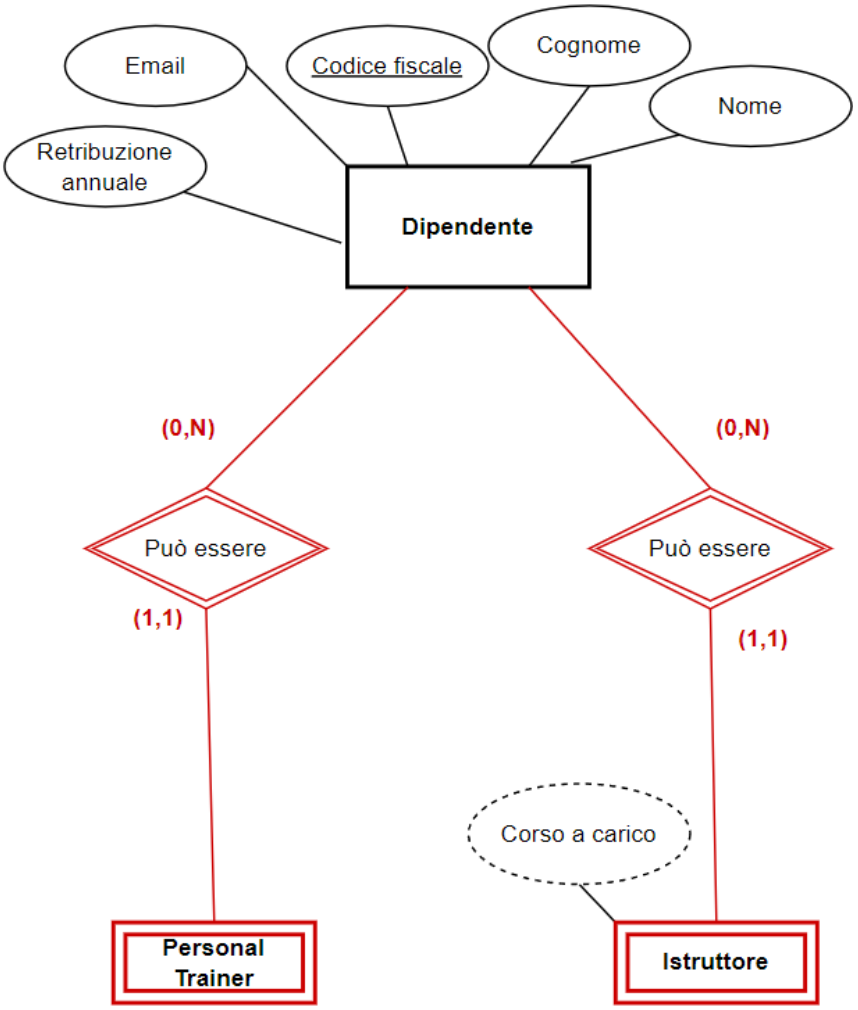
In definitiva conviene mantenere il dato ridondante

Eliminazione delle gerarchie

Nello schema inizialmente elaborato, è presente la seguente specializzazione dell'entità "Dipendente":

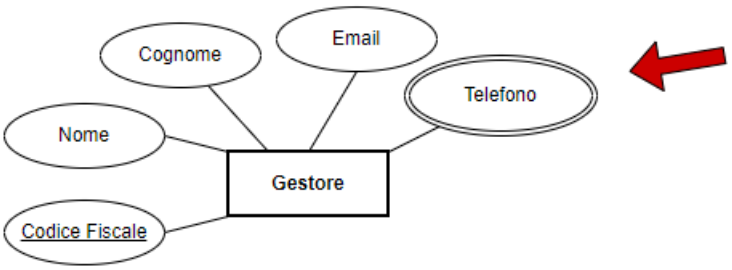


In questa fase di progettazione logica, è necessario individuare un metodo efficace di ristrutturazione che permetta l'eliminazione di questa gerarchia: la scelta effettuata è la creazione di due nuove relazioni, che rendono le entità figlie
due nuove entità deboli.

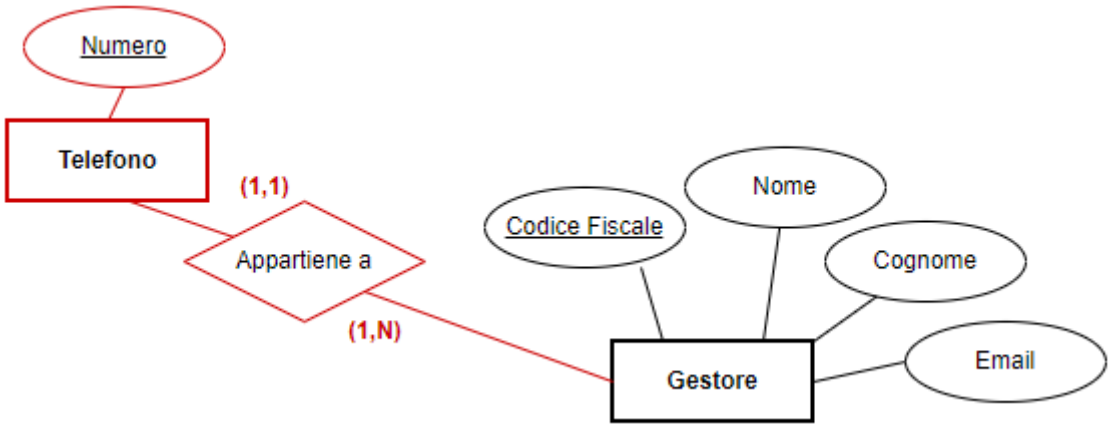


Eliminazione dell'attributo multivalore

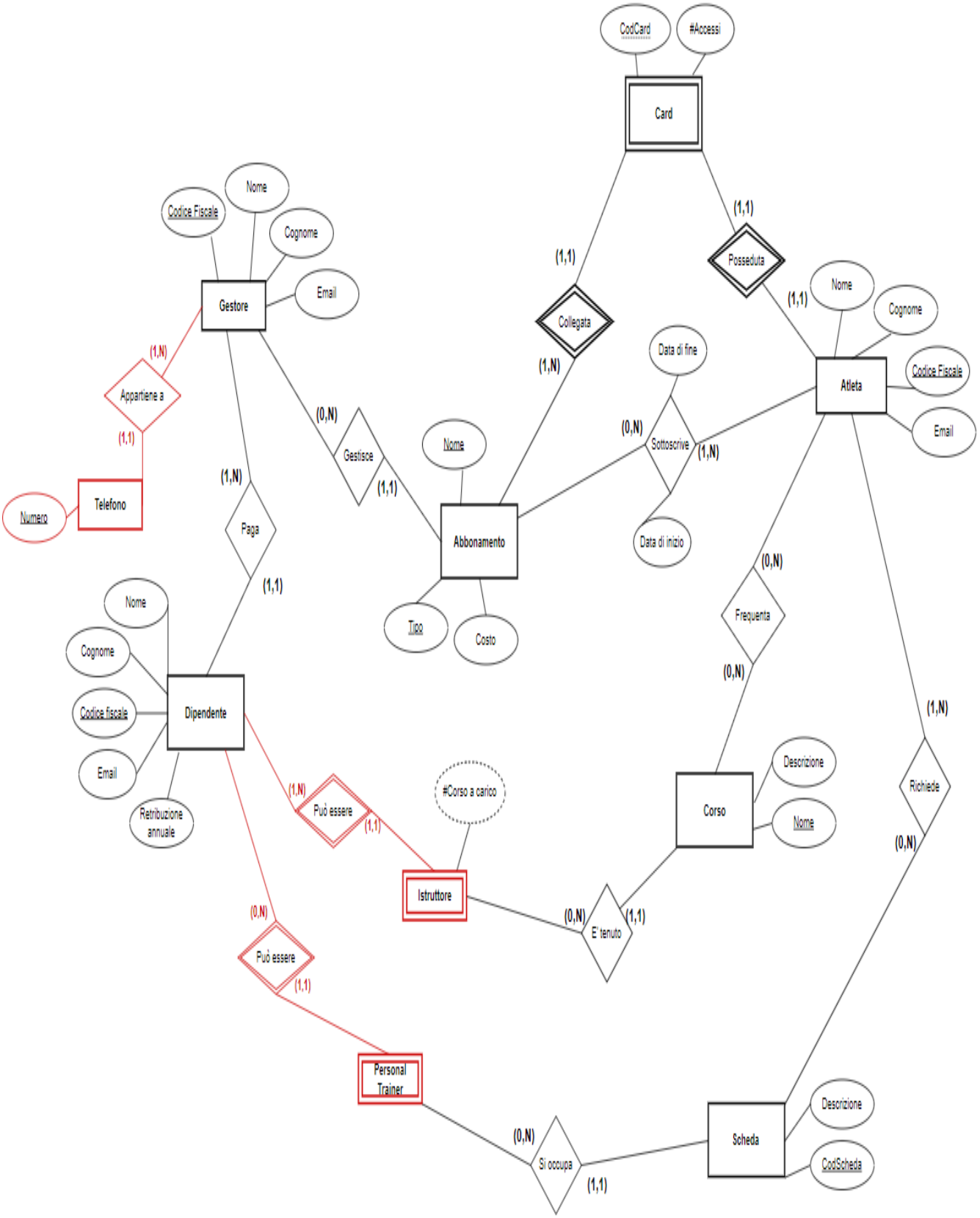
Nello schema inizialmente elaborato, compare un attributo multivalore:



Tale forma di attributo va risolto in maniera differente in fase di progettazione logica. Si sceglie quindi di definire una nuova entità forte "Telefono", in relazione con l'entità "Gestore":


Numero
Telefono
(1,1)
Appartiene a
(1,N)
Codice Fiscale
Nome
Cognome
Email
Gestore

Schema EER ristrutturato



Schema relazionale

Si procede al mapping della base di dati:

Atleta(Codice Fiscale, nome, cognome, e-mail)
Gestore(Codice Fiscale, nome, cognome, email)
Abbonamento(Nome, Tipo,gestore.codiceFiscale↑, costo)
Sottoscrive(atleta.codiceFiscale↑, abbonamento.nome↑,abbonamento.tipo↑, dataInizio, dataFine)
Card(CodCard, atleta.codiceFiscale↑, abbonamento.nome↑,abbonamento.tipo↑, #accessi)
Dipendente(Codice Fiscale,gestore.codiceFiscale↑, nome, cognome, e-mail,retribuzione annuale)
Istruttore(dipendente.codiceFiscale↑,corso a carico)
Corso(Nome, istruttore.codiceFiscale↑, descrizione)
Frequenta(corso.nome↑, atleta.codiceFiscale↑)
Personal Trainer(dipendente.codiceFiscale↑)
Scheda(codScheda, pt.codiceFiscale↑, descrizione)
Richiede(atleta.codiceFiscale↑, scheda.codScheda↑)
Telefono(Numero, gestore.codiceFiscale↑)

Normalizzazione

Il database si presenta già normalizzato.
È infatti in **prima forma normale** in quanto tutti gli attributi sono atomici dopo la ristrutturazione (è stato infatti eliminato l'attributo multivalore 'telefono' nell'entità Gestore). È in **seconda forma normale** perché, oltre ad essere già in 1NF, quando è presente una chiave primaria composta da più attributi tutte le dipendenze funzionali che la riguardano sono piene e non parziali. È in **terza forma normale** perché, oltre ad essere già in 2NF, in tutte le tabelle non sono presenti dipendenze transitive fra attributi non chiave e la chiave primaria.

5. Realizzazione della base di dati con MySQL

Di seguito la lista delle istruzioni MySQL per la creazione della base di dati:

```sql
DROP SCHEMA IF EXISTS Palestra;
CREATE SCHEMA IF NOT EXISTS Palestra;
USE Palestra;

/*Atleta(Codice Fiscale, nome, cognome, e-mail)*/
CREATE TABLE IF NOT EXISTS Atleta(
codice_fiscale_atleta char(16) NOT NULL,
nome varchar(20) NOT NULL,
cognome varchar(20) NOT NULL,
email varchar(30) NOT NULL,
PRIMARY KEY(codice_fiscale_atleta)
);

/*Gestore(Codice Fiscale, nome, cognome, email)*/
CREATE TABLE IF NOT EXISTS Gestore(
codice_fiscale_gestore char(16) NOT NULL,
nome varchar(20) NOT NULL,
cognome varchar(20) NOT NULL,
email varchar(30) NOT NULL,
PRIMARY KEY(codice_fiscale_gestore)
);

/*Abbonamento(nomeUnivoco, gestore.codiceFiscale↑, tipo, costo)*/
CREATE TABLE IF NOT EXISTS Abbonamento(
nome_univoco varchar(15) NOT NULL,
codice_fiscale_gestore char(16) NOT NULL,
tipo varchar(14) NOT NULL,
costo int NOT NULL,
PRIMARY KEY(nome_univoco),
FOREIGN KEY(codice_fiscale_gestore) REFERENCES Gestore(codice_fiscale_gestore) ON UPDATE CASCADE ON DELETE CASCADE
);
/*Sottoscrive(atleta.codiceFiscale↑, abbonamento.nome↑, abbonamento.tipo↑, dataInizio, dataFine)*/
CREATE TABLE IF NOT EXISTS Sottoscrive(
codice_fiscale_atleta char(16) NOT NULL,
nome varchar(15) NOT NULL,
tipo varchar(14) NOT NULL,
data_inizio date NOT NULL,
data_fine date NOT NULL,
PRIMARY KEY(codice_fiscale_atleta, nome, tipo),
FOREIGN KEY(codice_fiscale_atleta) REFERENCES Atleta(codice_fiscale_atleta) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(nome, tipo) REFERENCES Abbonamento(nome, tipo) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Card(CodCard, atleta.codiceFiscale↑, abbonamento.nome, abbonamento.tipo↑, #accessi)*/
CREATE TABLE IF NOT EXISTS Card(
codice_card varchar(15) NOT NULL,
codice_fiscale_atleta char(16) NOT NULL,
nome varchar(15) NOT NULL,
tipo varchar(14) NOT NULL,
accessi int DEFAULT 0,
PRIMARY KEY(codice_card, codice_fiscale_atleta, nome, tipo),
FOREIGN KEY(codice_fiscale_atleta) REFERENCES Atleta(codice_fiscale_atleta) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(nome, tipo) REFERENCES Abbonamento(nome, tipo) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
/*Dipendente(Codice Fiscale,gestore.codiceFiscale↑, nome, cognome, e-mail,retribuzione annuale)*/
CREATE TABLE IF NOT EXISTS Dipendente(
codice_fiscale_dipendente char(16) NOT NULL,
codice_fiscale_gestore char(16),
nome varchar(20) NOT NULL,
cognome varchar(20) NOT NULL,
email varchar(30) NOT NULL,
retribuzione_annuale bigint NOT NULL,
PRIMARY KEY(codice_fiscale_dipendente),
FOREIGN KEY(codice_fiscale_gestore) REFERENCES Gestore(codice_fiscale_gestore) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Istruttore(dipendente.codiceFiscale↑,corso a carico)*/
CREATE TABLE IF NOT EXISTS Istruttore(
codice_fiscale_istruttore char(16) NOT NULL,
corso_a_carico varchar(20) NOT NULL,
PRIMARY KEY(codice_fiscale_istruttore),
FOREIGN KEY(codice_fiscale_istruttore) REFERENCES Dipendente(codice_fiscale_dipendente) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Corso(Nome, istruttore.codiceFiscale↑, descrizione)*/
CREATE TABLE IF NOT EXISTS Corso(
nome varchar(20) NOT NULL,
codice_fiscale_istruttore char(16) NOT NULL,
descrizione varchar(3000),
PRIMARY KEY(nome),
FOREIGN KEY(codice_fiscale_istruttore) REFERENCES Istruttore(codice_fiscale_istruttore) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Frequenta(corso.nome↑, atleta.codiceFiscale↑)*/
CREATE TABLE IF NOT EXISTS Frequenta(
nome varchar(20) NOT NULL,
codice_fiscale_atleta char(16) NOT NULL,
PRIMARY KEY(nome,codice_fiscale_atleta),
FOREIGN KEY(nome) REFERENCES Corso(nome) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(codice_fiscale_atleta) REFERENCES Atleta(codice_fiscale_atleta) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Personal Trainer(dipendente.codiceFiscale↑)*/
CREATE TABLE IF NOT EXISTS Personal_trainer(
codice_fiscale_pt char(16) NOT NULL,
PRIMARY KEY(codice_fiscale_pt),
FOREIGN KEY(codice_fiscale_pt) REFERENCES Dipendente(codice_fiscale_dipendente) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Scheda(codScheda, PT.codiceFiscale↑, descrizione)*/
CREATE TABLE IF NOT EXISTS Scheda(
codice_scheda varchar(15) NOT NULL,
codice_fiscale_pt char(16) NOT NULL,
descrizione varchar(1000),
PRIMARY KEY(codice_scheda),
FOREIGN KEY(codice_fiscale_pt) REFERENCES Personal_trainer(codice_fiscale_pt) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
/*Scheda(codScheda, PT.codiceFiscale↑, descrizione)*/
CREATE TABLE IF NOT EXISTS Scheda(
codice_scheda varchar(15) NOT NULL,
codice_fiscale_pt char(16) NOT NULL,
descrizione varchar(1000),
PRIMARY KEY(codice_scheda),
FOREIGN KEY(codice_fiscale_pt) REFERENCES Personal_trainer(codice_fiscale_pt) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Richiede(atleta.codiceFiscale↑, scheda.codScheda↑)*/
CREATE TABLE IF NOT EXISTS Richiede(
codice_fiscale_atleta char(16) NOT NULL,
codice_scheda varchar(15) NOT NULL,
PRIMARY KEY(codice_fiscale_atleta,codice_scheda),
FOREIGN KEY(codice_fiscale_atleta) REFERENCES Atleta(codice_fiscale_atleta) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(codice_scheda) REFERENCES Scheda(codice_scheda) ON UPDATE CASCADE ON DELETE CASCADE
);

/*Telefono(Numero, gestore.codiceFiscale↑)*/
CREATE TABLE IF NOT EXISTS Telefono(
numero bigint NOT NULL,
codice_fiscale_gestore char(16) NOT NULL,
PRIMARY KEY(numero),
FOREIGN KEY(codice_fiscale_gestore) REFERENCES Gestore(codice_fiscale_gestore) ON UPDATE CASCADE ON DELETE CASCADE
);

```

6. Implementazione query SQL

```sql
/*Operazione 1:Registrare un nuovo abbonamento*/
INSERT INTO Gestore(codice_fiscale_gestore,nome,cognome,email)
VALUES ('CLCGRD94D20A509D','Gerardo','Colace','gcolace@gmail.com');

INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento annuale misto*/
VALUES ('annuale','misto','CLCGRD94D20A509D',250);

INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento semestrale sala pesi*/
VALUES ('semestrale','sala_pesi','CLCGRD94D20A509D',100);

INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento semestrale corso*/
VALUES ('semestrale','corso','CLCGRD94D20A509D',75);

INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento annuale sala pesi*/
VALUES ('annuale','sala_pesi','CLCGRD94D20A509D',200);

INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento mensile corso*/
VALUES ('mensile','corso','CLCGRD94D20A509D',15);

/*Operazione 2:Inserire un nuovo atleta*/
INSERT INTO Atleta(codice_fiscale_atleta,nome,cognome,email)
VALUES ('FRRFNC01P20I805J','Francesco','Ferrara','ferraraf682@gmail.com');/*Francesco*/

INSERT INTO Card(codice_card, codice_fiscale_atleta, nome, tipo, accessi)
VALUES ('DESA9854','FRRFNC01P20I805J','annuale','sala_pesi',1);

INSERT INTO Sottoscrive(codice_fiscale_atleta, nome, tipo, data_inizio, data_fine)
VALUES ('FRRFNC01P20I805J','annuale','sala_pesi','2022-06-06','2023-06-06');

INSERT INTO Atleta(codice_fiscale_atleta,nome,cognome,email)/*Alberto*/
VALUES ('RSSLRT03A01H703I','Alberto','Rossi','siciliano@gmail.com');

INSERT INTO Card(codice_card,codice_fiscale_atleta,nome, tipo, accessi)
VALUES ('MISA9854','RSSLRT03A01H703I','annuale','sala_pesi',1);

INSERT INTO Sottoscrive(codice_fiscale_atleta, nome, tipo, data_inizio, data_fine)
VALUES ('RSSLRT03A01H703I','annuale','sala_pesi','2022-07-12','2023-07-12');

```

```sql
INSERT INTO Atleta(codice_fiscale_atleta,nome,cognome,email)/*Samuel*/
VALUES ('TEOSML96H15H703J','Samuel','Etoo','samueletoo@gmail.com');

INSERT INTO Card(codice_card,codice_fiscale_atleta,nome,tipo,accessi)
VALUES ('MISA7890','TEOSML96H15H703J','semestrale','corso',1);

INSERT INTO Sottoscrive(codice_fiscale_atleta, nome, tipo, data_inizio, data_fine)
VALUES ('TEOSML96H15H703J','semestrale','corso','2020-01-12','2020-07-12');

/*Operazione 3:Aggiungere un nuovo dipendente*/
INSERT INTO Dipendente(codice_fiscale_dipendente,codice_fiscale_gestore,nome,cognome,email,retribuzione_annuale)
VALUES ('FRNSRA01R60A509X','CLCGRD94D20A509D','Sara','Fornace','sarafornace@gmail.com',20000);

INSERT INTO Dipendente(codice_fiscale_dipendente,codice_fiscale_gestore,nome,cognome,email,retribuzione_annuale)
VALUES ('FNZLNS00A01A509E','CLCGRD94D20A509D','Alfonso','Fanzi','alfanzi@gmail.com',20000);

INSERT INTO Dipendente(codice_fiscale_dipendente,codice_fiscale_gestore,nome,cognome,email,retribuzione_annuale)
VALUES ('DNILSS95A41F839D','CLCGRD94D20A509D','Alessia','Diana','odiana@gmail.com',20000);

/*Operazione 4:Aggiungere un nuovo corso*/
INSERT INTO Istruttore(codice_fiscale_istruttore,corso_a_carico)
VALUES ('FRNSRA01R60A509X',1);

INSERT INTO Corso(nome,codice_fiscale_istruttore,descrizione)
VALUES ('Breakdance','FRNSRA01R60A509X','Ballo acrobatico, nato
negli anni 80 del Novecento in connessione con la musica rap
come espressione dei gruppi giovanili dei quartieri neri di New York,
caratterizzato da rapidi movimenti a scatti e da evoluzioni spettacolari.');

INSERT INTO Corso(nome,codice_fiscale_istruttore,descrizione)
VALUES ('Pugilato','FRNSRA01R60A509X','Sport agonistico nel quale due contendenti
        si affrontano battendosi, secondo precisi regolamenti, con i pugni ricoperti da guantoni.');

UPDATE Istruttore AS I
                  SET I.corso_a_carico= (
                  SELECT COUNT(*) AS corsi_a_carico
                  FROM Corso AS C
                  WHERE I.codice_fiscale_istruttore = C.codice_fiscale_istruttore)
                  WHERE I.codice_fiscale_istruttore = 'FRNSRA01R60A509X';

INSERT INTO Frequenta(nome,codice_fiscale_atleta)
VALUES ('Pugilato','FRRFNC01P20I805J');

/*Operazione 5:Aggiungere una nuova scheda*/
INSERT INTO Personal_trainer(codice_fiscale_pt)
VALUES ('FNZLNS00A01A509E');

INSERT INTO Personal_trainer(codice_fiscale_pt)
VALUES ('DNILSS95A41F839D');

INSERT INTO Scheda(codice_scheda,codice_fiscale_pt,descrizione)
VALUES ('IT490','FNZLNS00A01A509E','Panca piana 5x5, Croci ai cavi 4x8, Pulley 4x8, Leg extension 3x10, French Press 3x10');

INSERT INTO Scheda(codice_scheda,codice_fiscale_pt,descrizione)
VALUES ('IT500','FNZLNS00A01A509E','Squat 6x4, Leg press 3x10/12, Panca 30 manubri 4x8, Rematore bil. 3x12, Curl ai cavi 3x10');

INSERT INTO Scheda(codice_scheda,codice_fiscale_pt,descrizione)
VALUES ('IT510','DNILSS95A41F839D','Stacchi da terra 8x3, leg curl 3x12, Shoulder press 4x8, Dip 3xmax, Push down 3x10');

/*Operazione 6:Inserire una nuova richiesta di una scheda*/
INSERT INTO Richiede(codice_fiscale_atleta,codice_scheda)
VALUES ('FRRFNC01P20I805J','IT510');

INSERT INTO Richiede(codice_fiscale_atleta,codice_scheda)
VALUES ('FRRFNC01P20I805J','IT500');
```

```sql
INSERT INTO Richiede(codice_fiscale_atleta,codice_scheda)
VALUES ('FRRFNC01P20I805J','IT490');

/*Operazione 7 Inserire un nuovo accesso di un atleta*/
UPDATE Card AS C
                    SET C.accessi = C.accessi + 1
                    WHERE C.codice_fiscale_atleta='RSSLRT03A01H703I';

/*Operazione 8:Stampare nome,cognome e retribuzione annuale di tutti i dipendenti*/
SELECT nome,cognome,retribuzione_annuale
FROM Dipendente;

/*Operazione 9:Trovare il cognome degli istruttori che insegnano a più di un corso*/
SELECT D.cognome,I.corso_a_carico
FROM Istruttore AS I,Dipendente AS D
WHERE D.codice_fiscale_dipendente=I.codice_fiscale_istruttore
HAVING I.corso_a_carico > 1;

/*Operazione 10:Visualizzare il numero di atleti di un dato corso*/
SELECT C.nome,COUNT(A.nome) AS numero_atleti
FROM Frequenta AS F,Atleta AS A,Corso AS C
WHERE A.codice_fiscale_atleta=F.codice_fiscale_atleta AND C.nome='Breakdance';

/*Operazione 11:Visualizzare nome e cognome degli atleti che hanno richiesto più di 2 schede*/
SELECT DISTINCT A.nome,A.cognome
FROM Atleta AS A,Richiede AS R,Scheda AS S
WHERE A.codice_fiscale_atleta=R.codice_fiscale_atleta AND S.codice_scheda=R.codice_scheda
HAVING COUNT(R.codice_scheda) > 2;

/*Operazione 12:Leggi il numero di accessi di uno specifico atleta*/
SELECT accessi
FROM Card
```

```sql
/*Operazione 12:Leggi il numero di accessi di uno specifico atleta*/
SELECT accessi
FROM Card
WHERE codice_fiscale_atleta ='RSSLRT03A01H703I';

/*Operazione 13:Visualizzare il numero di corsi a carico di un istruttore*/
SELECT corso_a_carico
FROM Istruttore AS I,Dipendente AS D
WHERE I.codice_fiscale_istruttore= 'FRNSRA01R60A509X' AND D.codice_fiscale_dipendente='FRNSRA01R60A509X';

/*Operazione 14:Ordinare gli abbonamenti di tutti gli iscritti per data di scadenza*/
SELECT A.nome,A.cognome,S.data_fine,A.email
FROM Sottoscrive AS S,Atleta AS A
WHERE S.codice_fiscale_atleta=A.codice_fiscale_atleta
ORDER BY data_fine ASC;

/*Operazione 15:Selezionare nome e cognome dei Personal Trainer che si sono occupati di almeno 3 schede*/
SELECT D.nome,D.cognome
FROM Dipendente AS D,Personal_trainer AS PT,Scheda AS S
WHERE PT.codice_fiscale_pt=S.codice_fiscale_pt AND D.codice_fiscale_dipendente=PT.codice_fiscale_pt
HAVING COUNT(S.codice_scheda) >= 3;

/*Operazione 16:Selezionare nome e cognome degli iscritti che frequentano almeno un corso*/
SELECT A.nome,A.cognome
FROM Atleta AS A ,Frequenta AS F
WHERE A.codice_fiscale_atleta=F.codice_fiscale_atleta;
```

```sql
/*Operazione 15:Selezionare nome e cognome dei Personal Trainer che si sono occupati di almeno 3 schede*/
SELECT D.nome,D.cognome
FROM Dipendente AS D,Personal_trainer AS PT,Scheda AS S
WHERE PT.codice_fiscale_pt=S.codice_fiscale_pt AND D.codice_fiscale_dipendente=PT.codice_fiscale_pt
HAVING COUNT(S.codice_scheda) >= 3;

/*Operazione 16:Selezionare nome e cognome degli iscritti che frequentano almeno un corso*/
SELECT A.nome,A.cognome
FROM Atleta AS A ,Frequenta AS F
WHERE A.codice_fiscale_atleta=F.codice_fiscale_atleta;

/*Operazione 17:Selezionare l'email di tutti gli atleti che non hanno sottoscritto un determinato tipo di abbonamento*/
SELECT A.email
FROM Atleta AS A
WHERE NOT EXISTS(
    SELECT Ab.nome
    FROM Abbonamento AS Ab,Sottoscrive AS S
    WHERE Ab.nome=S.nome AND S.codice_fiscale_atleta=A.codice_fiscale_atleta AND Ab.tipo='misto'
);

/*Operazione 18:Selezionare l'email di tutti gli atleti che hanno una determinata scheda*/
SELECT A.email
FROM Atleta AS A
WHERE NOT EXISTS(
    SELECT S.codice_scheda
    FROM Scheda AS S,Richiede AS R
    WHERE S.codice_scheda='IT490' AND R.codice_fiscale_atleta=A.codice_fiscale_atleta
);
```

7.Test dell’applicazione Java

All’avvio dell’applicazione si può scegliere mediante l’inserimento di un valore(da 1 a 18,0 per terminare l’esecuzione), la query da eseguire nel sistema,tra quelle esposte sopra.Effettuiamo procedure che eseguono diverse operazioni realizzate senza l’intervento dell’utente per l’inserimento dei dati.

Viene usato una DBConnectionPool poichè le operazioni di connessione,ad ogni query,sarebbero state troppo onerose per il sistema dunque con questa pratica la connessione al database e il rilascio di quest’ultima vengono eseguite rispettivamente una sola volta.

Definiamo dei test della base di dati e stampiamo dei risultati delle interrogazioni su tali dati.Ecco gli screenshot che visualizzano il risultato dell’esecuzione di alcune delle query implementate.

```
Inserisci numero query da eseguire (da 1 a 18, 0 se EXIT):
1
Registrare un nuovo abbonamento:


Inserimento effettuato.

Inserisci numero query da eseguire (da 1 a 18, 0 se EXIT):
2
Inserire un nuovo atleta:


Inserimento effettuato.

Inserisci numero query da eseguire (da 1 a 18, 0 se EXIT):
12
Leggi il numero di accessi di uno specifico atleta:


1

Inserisci numero query da eseguire (da 1 a 18, 0 se EXIT):
7
Operazione 7 Inserire un nuovo accesso di un atleta
Modifica effettuata.

Inserisci numero query da eseguire (da 1 a 18, 0 se EXIT):
12
Leggi il numero di accessi di uno specifico atleta:


2

Inserisci numero query da eseguire (da 1 a 18, 0 se EXIT):
```

Inizialmente vengono eseguite le query 1 e 2 per l'inserimento nel database di alcuni dati,dopodichè viene effettuata la query 12 che legge il numero di accessi di uno specifico atleta(nella query 2 era stato inserito un accesso all'atleta Alberto Rossi).

Viene poi eseguita la query 7 che incrementa il numero di accessi sempre dello stesso atleta e alla fine viene ri-eseguita la query 12 per testare il corretto funzionamento della nuova Card di Alberto Rossi nella nostra base dati.

Di seguito il codice java della query01:

```java
public void query01() {
    try {
        System.out.println("Registrare un nuovo abbonamento:\n\n");

        String query = "INSERT INTO Gestore(codice_fiscale_gestore,nome,cognome,email)\r\n"
                + "VALUES ('CLCGRD94D20A509D','Gerardo','Colace','gcolace@gmail.com');";
        Statement st = con.createStatement();
        st.executeUpdate(query);

        query = "INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento annuale misto*/\r\n"
                + "VALUES ('annuale','misto','CLCGRD94D20A509D',250);";
        st = con.createStatement();
        st.executeUpdate(query);

        query = "INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento semestrale sala pesi*/\r\n"
                + "VALUES ('semestrale','sala_pesi','CLCGRD94D20A509D',100);";
        st = con.createStatement();
        st.executeUpdate(query);

        query= "INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento semestrale corso*/\r\n"
                + "VALUES ('semestrale','corso','CLCGRD94D20A509D',75);";
        st = con.createStatement();
        st.executeUpdate(query);

        query = "INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento annuale sala pesi*/\r\n"
                + "VALUES ('annuale','sala_pesi','CLCGRD94D20A509D',200);";
        st = con.createStatement();
        st.executeUpdate(query);

        query= "INSERT INTO Abbonamento(nome, tipo, codice_fiscale_gestore,costo)/*Abbonamento mensile corso*/\r\n"
                + "VALUES ('mensile','corso','CLCGRD94D20A509D',15);";
        st = con.createStatement();
        st.executeUpdate(query);

        System.out.println("Inserimento effettuato.");
    }
    catch(Exception e) {
        System.err.print(e.getMessage());
    }
}
```

la query02:

```
62    public void query02() {
63        try {
64            System.out.println("Inserire un nuovo atleta:\n\n");
65
66            String query = "INSERT INTO Atleta(codice_fiscale_atleta,nome,cognome,email)\r\n"
67                    + "VALUES ('FRRFNC01P20I805J','Francesco','Ferrara','ferraraf682@gmail.com');\r\n";
68            Statement st = con.createStatement();
69            st.executeUpdate(query);
70
71            query = "INSERT INTO Card(codice_card, codice_fiscale_atleta, nome, tipo, accessi)\r\n"
72                    + "VALUES ('DESA9854','FRRFNC01P20I805J','annuale','sala_pesi',1);\r\n";
73            st = con.createStatement();
74            st.executeUpdate(query);
75
76            query = "INSERT INTO Sottoscrive(codice_fiscale_atleta, nome, tipo, data_inizio, data_fine)\r\n"
77                    + "VALUES ('FRRFNC01P20I805J','annuale','sala_pesi','2022-06-06','2023-06-06');\r\n";
78            st = con.createStatement();
79            st.executeUpdate(query);
80
81            query = "INSERT INTO Atleta(codice_fiscale_atleta,nome,cognome,email)\r\n"
82                    + "VALUES ('RSSLRT03A01H703I','Alberto','Rossi','siciliano@gmail.com');";
83            st = con.createStatement();
84            st.executeUpdate(query);
85
86            query = "INSERT INTO Card(codice_card,codice_fiscale_atleta,nome, tipo, accessi)\r\n"
87                    + "VALUES ('MISA9854','RSSLRT03A01H703I','annuale','sala_pesi',1);\r\n";
88            st = con.createStatement();
89            st.executeUpdate(query);
90
91            query = "INSERT INTO Sottoscrive(codice_fiscale_atleta, nome, tipo, data_inizio, data_fine)\r\n"
92                    + "VALUES ('RSSLRT03A01H703I','annuale','sala_pesi','2022-07-12','2023-07-12');\r\n";
93            st = con.createStatement();
94            st.executeUpdate(query);
95
96            query = "INSERT INTO Atleta(codice_fiscale_atleta,nome,cognome,email)\r\n"
97                    + "VALUES ('TEOSML96H15H703J','Samuel','Etoo','samueletoo@gmail.com');";
98            st = con.createStatement();
99            st.executeUpdate(query);
100
101           query = "INSERT INTO Card(codice_card,codice_fiscale_atleta,nome,tipo,accessi)\r\n"
102                   + "VALUES ('MISA7890','TEOSML96H15H703J','semestrale','corso',1);\r\n";
103           st = con.createStatement();
104           st.executeUpdate(query);
105
106           query = "INSERT INTO Sottoscrive(codice_fiscale_atleta, nome, tipo, data_inizio, data_fine)\r\n"
107                   + "VALUES ('TEOSML96H15H703J','semestrale','corso','2020-01-12','2020-07-12');\r\n";
108           st = con.createStatement();
109           st.executeUpdate(query);
110
111           System.out.println("Inserimento effettuato.");
112       }
113       catch(Exception e) {
114           System.err.print(e.getMessage());
```

la query07:

```
248        public void query07() {
249            try {
250                System.out.println("Operazione 7 Inserire un nuovo accesso di un atleta");
251
252                String query = "UPDATE Card AS C\r\n"
253                        + "                SET C.accessi = C.accessi + 1\r\n"
254                        + "                WHERE C.codice_fiscale_atleta='RSSLRT03A01H703I';";
255                Statement st = con.createStatement();
256                st.executeUpdate(query);
257
258                System.out.println("Modifica effettuata.");
259
260                }
261            catch(Exception e) {
262                System.err.print(e.getMessage());
263            }
264        }
```

e della query 12:

```
357        public void query12() {
358            try {
359                System.out.println("Leggi il numero di accessi di uno specifico atleta:\n\n");
360
361                String query = "SELECT accessi\r\n"
362                        + "FROM Card\r\n"
363                        + "WHERE codice_fiscale_atleta ='RSSLRT03A01H703I';";
364                PreparedStatement cmd = con.prepareStatement(query);
365                ResultSet res = cmd.executeQuery();
366                boolean tro = res.next();
367                while(tro) {
368                    int accessi = res.getInt("accessi");
369                    System.out.println(accessi);
370                    tro = res.next();
371                }
372            }
373            catch(Exception e) {
374                System.err.print(e.getMessage());
375            }
376        }
```