UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Dipartimento di Matematica

Laurea Triennale in Informatica

Relazione al progetto per il corso Base di Dati

Stadium Security

Lago Leonardo	2034297
Cecchin Andrea	2034299

1 Abstract

Stadium Security è una società che, attraverso contratti di appalto, si impegna a fornire il personale necessario a gestire la sicurezza e la vigilanza durante eventi, siano essi di natura sportiva o meno, all'interno di stadi.
La società, per garantire il servizio, assume con dei contratti a chiamata quelle figure che comunemente vengono chiamate steward. Ad affiancarli, prestano servizio membri delle forze dell'ordine, come poliziotti, carabinieri e vigili del fuoco e personale sanitario istruito, ad esempio volontari della croce rossa.
Per garantire la corretta gestione della sicurezza anche a livello burocratico, Stadium Security opera a stretto contatto con le procure delle varie città dove sono ubicati gli stadi. Questo permette un tempestivo report relativo a comportamenti ritenuti pericolosi e/o illeciti da parte del pubblico spettatore dei vari eventi.
Ogni società cliente è chiamata a designare un responsabile, figura di riferimento che la rappresenta, per ogni suo stadio in custodia.

2 Analisi dei requisiti

2.1 Descrizione testuale

Nella base di dati sono presenti le informazioni relative agli **operatori** addetti alla sicurezza, e di ognuno sono noti:

- codice fiscale
- nome
- cognome
- data di nascita
- sesso

Un operatore può essere uno **steward**, di cui si conosce lo stipendio orario in euro, un **vigile del fuoco**, di cui si conosce l'abilitazione o meno a trattare oggetti pericolosi, un addetto al **pronto intervento**, di cui si tiene traccia dell'abilitazione blsd, oppure una **forza dell'ordine**, di cui si vuole conoscere il corpo di appartenenza.
Delle informazioni relative agli **eventi** si considera:

- codice
- nome
- data

Quando un operatore viene assegnato ad un evento, si avrà informazione anche su quale settore esso ha prestato servizio
Di uno **stadio** si tiene traccia:

- codice
- nome

- capienza
- numero totale di settori
- città

Per ognuno dei **responsabili** di un impianto si vuole conoscere:

- codice fiscale
- nome
- cognome
- data di nascita
- sesso
- società per cui lavora

Delle **società** con gestione degli stadi, si vuole tener traccia:

- codice della partita IVA
- nome
- città della sede principale
- capitale sociale in euro

Per ogni **persona segnalata** nel corso di un evento da un operatore steward, vigile del fuoco o forza dell'ordine, si vuole conoscere:

- codice fiscale
- nome
- cognome
- data di nascita
- sesso
- recapito telefonico
- motivo della segnalazione
- evento nel quale è stato segnalato
- operatore autore della segnalazione

Un segnalato può andare in contro a daspo, pena inflitta da un **procuratore**. Di questi ultimi, si terrà conto di:

- codice fiscale
- nome

- cognome
- data di nascita
- sesso
- procura di riferimento

Quando un segnalato riceve il daspo, di esso si terrà traccia della data di inizio e fine. La data di inizio dell'espulsione deve essere successiva a quella dell'evento che ne ha scaturito la pena.

2.2 Glossario dei termini

Termine	Descrizione
Operatore	Persona incaricata alla sicurezza durante un evento. Si dividono in steward, vigili del fuoco, forze dell'ordine e addetti al pronto intervento.
Steward	Addetto alla vigilanza.
Vigile del fuoco	Addetto alla vigilanza chiamato a intervenire in caso di lancio di oggetti pericolosi da parte del pubblico.
Pronto intervento	Persona incaricata di prestare assistenza sanitaria.
BLSD	Abilitazione all'utilizzo del defibrillatore in caso di necessità.
Forze dell'ordine	Pubblico ufficiale incaricato di sorvegliare ad un evento.
Evento	Manifestazione, di qualunque tipologia, con sede in uno stadio.
Stadio	Impianto sportivo sede degli eventi di interesse.
Società	Società che hanno in gestione gli stadi.
Settore	Porzione degli spalti di uno stadio.
Responsabile	Persona incaricata dalla società per cui lavora a gestire la messa in sicurezza degli eventi.
Segnalazione	Notifica di uno spettatore che ha compiuto un gesto illecito.
Segnalato	Spettatore di un evento che ha compiuto un gesto illecito.
Procuratore	Persona incaricata di valutare le segnalazioni, comminando o meno daspo allo spettatore segnalato.
Daspo	Espulsione di un soggetto per un determinato lasso di tempo da ogni evento all'interno di uno stadio stabilito.

3 Progettazione concettuale

3.1 Lista entità

1. *Operatore*:

   CF char(16) primary key

   Nome varchar(25) not null

Cognome varchar(25) not null
Sesso enum('M','F')
DdN date not null

L'entità *Operatore* si specializza in quattro sottocategorie con una generalizzazione totale:

(a) *Steward*:
Stipendio float(5)
(b) *VVFF*:
Pericolo bool
(c) *Pronto Intervento*:
Blsd bool
(d) *Forze dell'ordine*:
Corpo varchar(25)

2. *Stadio*:

Codice : char(3) primary key
Nome : varchar(25) not null
Capienza : integer not null
Città : varchar(25) not null
NSettori : integer not null

3. *Evento*:

Codice serial primary key
Nome varchar(50) not null
Data date not null

4. *Società*:

PIVA char(16) primary key
Nome varchar(25) not null
Sede varchar(25) not null
Capitale integer not null

5. *Responsabile*:

CF char(16) primary key
Nome varchar(25) not null
Cognome varchar(25) not null
Sesso enum('M','F')
DdN date not null

6. *Procuratore*:

CF char(16) primary key

Nome varchar(25) not null

Cognome varchar(25) not null

Sesso enum('M','F')

DdN date not null

Procura varchar(25) not null

7. *Segnalato*:

   CF char(16) primary key

   Nome varchar(25) not null

   Cognome varchar(25) not null

   Sesso enum('M','F')

   DdN date not null

   Telefono char(10) not null

8. *Segnalazione*:

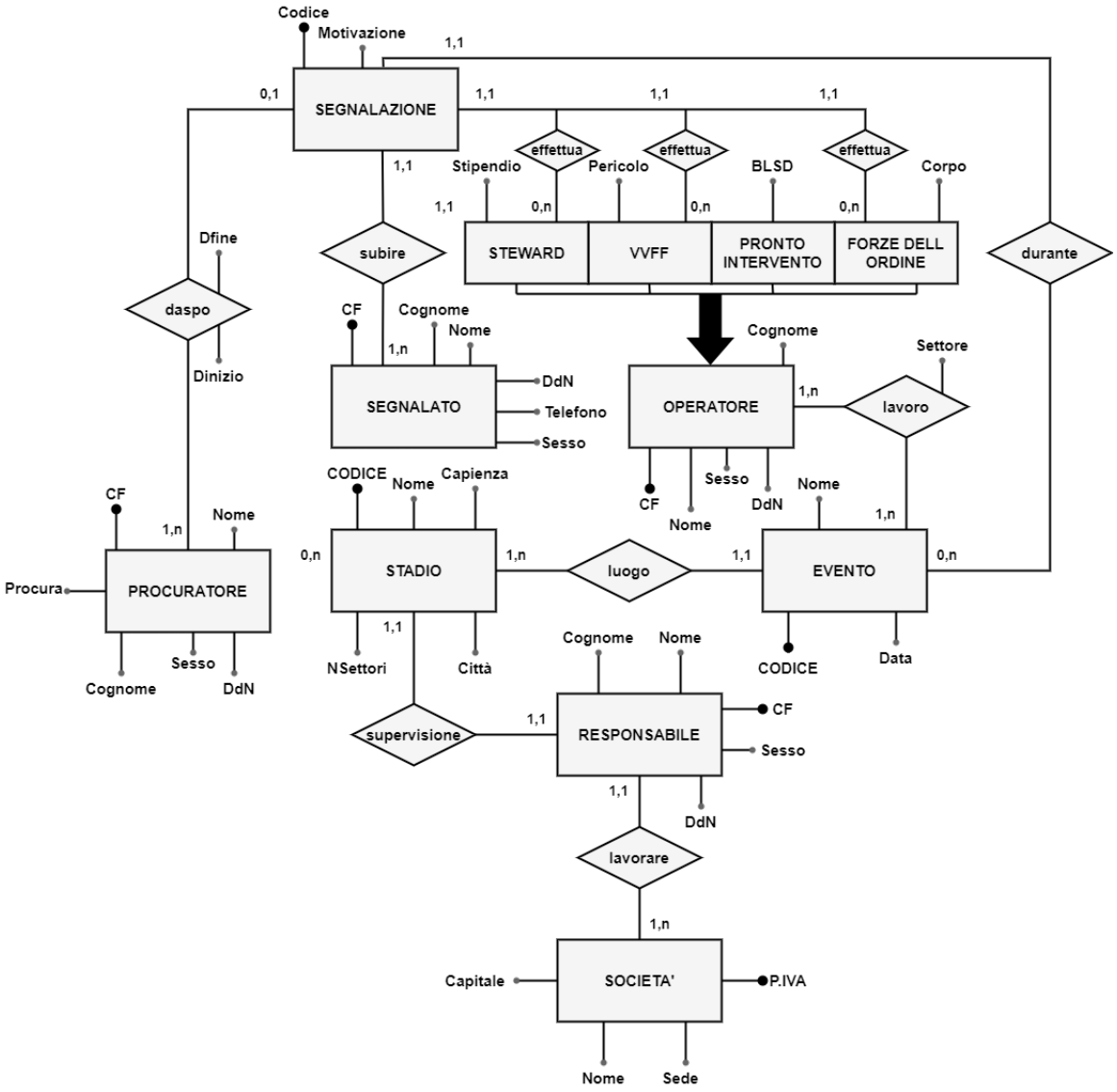
   Codice serial primary key

   Motivazione varchar(256) not null

3.2 Tabella delle relazioni

Relazione	Entità	Descrizione	Attributi
Lavoro	Operatore (1,n)	Un operatore può aver lavorato in uno o più eventi.	Settore
	Evento (1,n)	In un evento operano da uno a più operatori.	
Luogo	Stadio (1,n)	In uno stadio possono aver luogo uno o più eventi.	
	Evento (1,1)	Un evento è svolto in un determinato stadio.	
Supervisione	Stadio (1,1)	Uno stadio può avere un unico responsabile.	
	Responsabile (1,1)	Un responsabile può supervisionare un solo stadio.	
Lavorare	Società (1,n)	Una società può designare uno o più responsabili, qualora abbia in gestione più di uno stadio.	
	Responsabile (1,1)	Un responsabile lavora per un'unica società.	
Durante	Evento (0,n)	Durante un evento possono avvenire zero o molteplici segnalazioni	
	Segnalazione (1,1)	Ad una segnalazione è legato un unico evento	
Effettua	Segnalazione (1,1)	Una segnalazione è effettuata da un unico operatore.	
	VVFF (0,n) Forzedellordine (0,n) Steward (0,n)	Ogni operatore può effettuare zero o molteplici segnalazioni.	
Subire	Segnalazione (1,1)	Una segnalazione si riferisce ad un unico segnalato	
	Segnalato (1,n)	Ogni segnalato può aver ricevuto una o molteplici segnalazioni	
Daspo	Segnalazione (0,1)	Una segnalazione può portare o meno ad una sentenza di daspo.	Datainizio
	Procuratore (1,n)	Un procuratore può comminare da uno a più daspi.	Datafine

3.3 Schema concettuale



4 Progettazione logica

4.1 Ristrutturazione ed eliminazione della gerarchia

La generalizzazione relativa all'entità operatore viene eliminata tramite una soluzione ibrida:

- Le entità steward, vigile del fuoco e forza dell'ordine vengono accorpate nell'entità operatore, aggiungendo così a quest'ultima l'attributo "ruolo" e rendendo opzionali gli attributi relativi

alle entità che prima erano figlie, ovvero "stipendio", "pericolo" e "corpo".

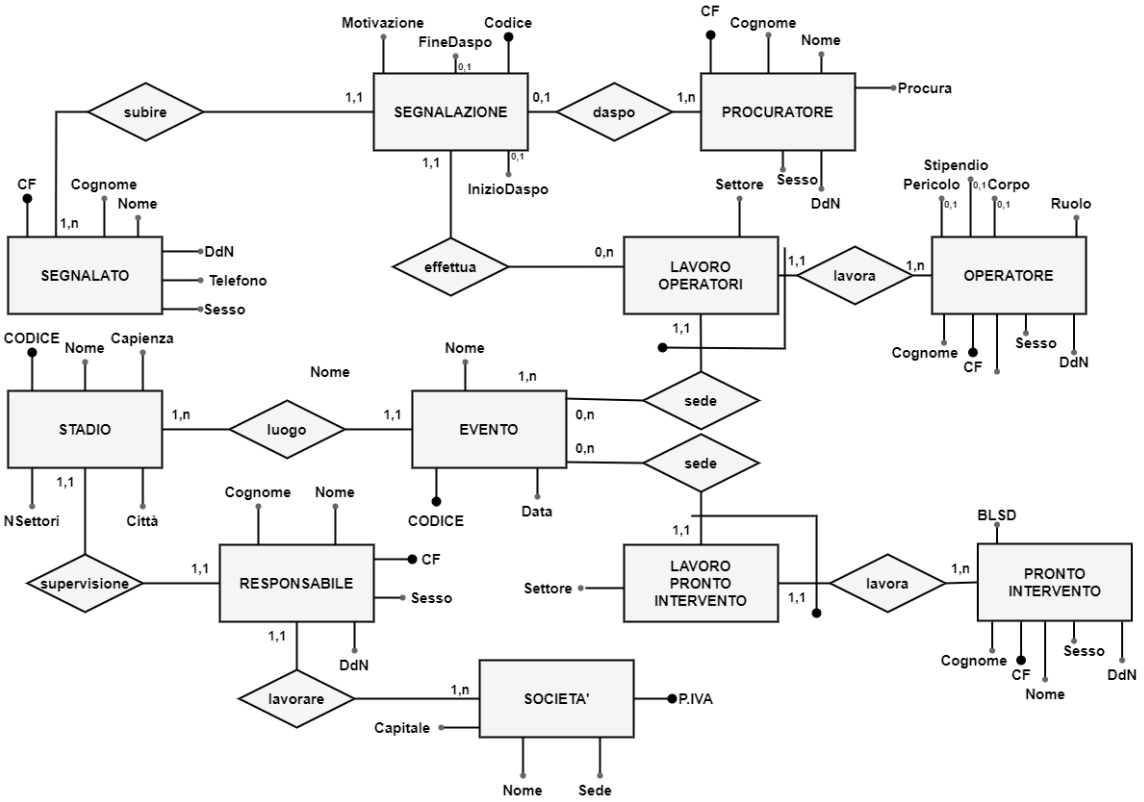
- La rimanente entità pronto intervento, la quale si differenzia dalle altre entità figlie in quanto non prendono parte alle segnalazioni, diventà entità a sé stante, inglobando al suo interno gli attributi di operatore.

Questa soluzione permette di distinguere gli operatori che possono o meno segnalare uno spettatore per comportamento illecito, dando vita così a due differenti nuove entità, chiamate "lavoro operatori" e "lavoro pronto intervento", prima rappresentate dalla relazione che univa operatore ed evento. In queste entità si tiene conto della coppia codice operatore e codice evento, e il relativo settore di cui l'operatore era chiamato a presiedere.
Questa modifica ci permette dunque di accertarci che alcun pronto intervento possa risultare come l'operatore autore di una segnalazione.
Analogamente, l'entità segnalazione sarà direttamente legata a lavoro operatore tramite la relazione effettua, così che una segnalazione venga effettuata da un operatore in un evento nel quale esso era chiamato ad operare.

4.2 Scelta degli identificatori primari

Per le nuove entità "lavoro operatori" e "lavoro pronto intervento", l'identificatore primario è rappresentato dalla coppia codice fiscale dell'operatore e codice numerico univoco dell'evento, i quali sono foreign key delle primary key di operatore, o pronto intervento, ed evento.
Ogni persona fisica nella base di dati è identificata dal proprio codice fiscale, con le altre entità quali società e segnalazioni identificate rispettivamente dal codice della partita IVA e da un codice numerico seriale. Rimane invece il codice univoco a tre caratteri per identificare gli stadi.

4.3 Schema Entità Relazioni ristrutturato



4.4 Schema logico

Operatore (<u>CF</u>, Nome, Cognome, Sesso, Datadinascita, Ruolo, Stipendio$_{[0,1]}$, Pericolo$_{[0,1]}$, Corpo$_{[0,1]}$)

Prontointervento (<u>CF</u>, Nome, Cognome, Sesso, Datadinascita, BLSD)

Stadio (<u>Codice</u>, Nome, Capienza, Citta, NSettori)

Evento (<u>Codice</u>, Nome, Data, Stadio$_{[FK]}$)

Stadio → Stadio.Codice

LavoroOperatore (<u>CodiceOperatore$_{[FK]}$</u>, Settore, <u>CodiceEvento$_{[FK]}$</u>)

CodiceOperatore → Operatore.CF
CodiceEvento → Evento.Codice

LavoroPI (<u>CodiceOperatore$_{[FK]}$</u>, Settore, <u>CodiceEvento$_{[FK]}$</u>)

CodiceOperatore → Prontointervento.CF
CodiceEvento → Evento.Codice

Società (<u>PIVA</u>, Nome, Sede, Capitale)

Responsabile (CF, Nome, Cognome, Sesso, Datadinascita, Stadio$_{[FK]}$, Societa$_{[FK]}$)

Stadio → Stadio.Codice
Societa → Societa.PIVA

Procuratore (CF, Nome, Cognome, Sesso, Datadinascita, Procura)

Segnalato (CF, Nome, Cognome, Sesso, Datadinascita, Telefono)

Segnalazione (Codice, CodiceSegnalato$_{[FK]}$, CodiceOperatore$_{[FK]}$, CodiceEvento$_{[FK]}$, Motivazione, CodiceProcuratore$_{[FK]}$, InizioDaspo$_{[FK]}$, FineDaspo$_{[FK]}$)

CodiceSegnalato → Segnalato.CF
CodiceOperatore → Operatore.CF
CodiceEvento → Evento.Codice
CodiceProcuratore → Procuratore.CF

5 Query e indici

5.1 Query

5.1.1 Query non parametriche

1. Mostra la lista di tutti gli stadi, riportando il loro nome, nome e cognome del responsabile che le gestisce e il nome della società per cui lavora.

```
SELECT r.nome, r.cognome, s.nome AS Stadio, sc.nome AS Societa
FROM responsabile r JOIN stadio s ON (r.stadio=s.codice)
     JOIN societa sc ON (r.societa=sc.piva)
```

2. Mostra la lista di tutte le segnalazione, riportando nome e cognome del segnalato, nome e cognome dell'operatore autore della segnalazione, oltre al nome dell'evento in questione.

```
SELECT segnalato.nome AS NomeSegnalato,
       segnalato.cognome AS CognomeSegnalato,
       operatore.nome AS NomeSteward,
       operatore.cognome AS CognomeSteward,
       evento.nome AS Evento
FROM segnalazione
     JOIN segnalato ON (segnalazione.codicesegnalato = segnalato.cf)
     JOIN evento ON (segnalazione.codiceevento = evento.codice)
     JOIN operatore ON (operatore.cf = segnalazione.codiceoperatore)
WHERE operatore.stipendio IS NOT NULL
```

3. Mostra la lista di tutti i daspi comminati, riportando nome e cognome del condannato, data di inizio e fine della pena, oltre al nome dello stadio dal quale è allontanato.

```sql
SELECT segnalato.nome, segnalato.cognome,
       stadio.nome AS stadio, s.iniziodaspo, s.finedaspo
FROM segnalazione s
     JOIN segnalato ON (segnalato.cf = s.codicesegnalato)
     JOIN evento e ON(s.codiceevento = e.codice)
     JOIN stadio ON (stadio.codice = e.stadio)
WHERE s.iniziodaspo IS NOT NULL AND s.finedaspo IS NOT NULL
```

4. Mosta gli eventi più costosi dal punto di vista organizzativo e il relativo costo totale degli stipendi ad ora.

```sql
SELECT e.nome AS evento, SUM(stipendio) AS costooratotale
FROM lavorooperatori l
    JOIN operatore o ON (l.codiceoperatore = o.cf)
    JOIN evento e ON (l.codiceevento = e.codice)
WHERE o.ruolo = 'Steward'
GROUP BY evento
HAVING SUM(stipendio) > 1000
ORDER BY evento ASC;
```

5.1.2 Vincoli non catturate da vincoli di integrità referenziale

1. Verifica che la data di inizio di ogni daspo sia successiva alla data dell'evento origine della condanna (verifica tabella vuota).

```sql
SELECT s.codice
FROM segnalazione s
     JOIN evento e ON (s.codiceevento = e.codice)
WHERE  s.iniziodaspo < e.data
```

2. Verifica che nessun operatore sia stato assegnato a presiedere un settore non esistente (verifica tabella vuota) guardando il numero di settori di ogni stadio.

```
SELECT lavorooperatori.codiceoperatore, evento.codice
FROM lavorooperatori
     JOIN evento ON (lavorooperatori.codiceevento = evento.codice)
     JOIN stadio ON (evento.stadio = stadio.codice)
WHERE lavorooperatori.settore>stadio.nsettori
UNION
SELECT lavoropi.codiceoperatore, evento.codice
FROM lavoropi
     JOIN evento ON (lavoropi.codiceevento = evento.codice)
     JOIN stadio ON (evento.stadio = stadio.codice)
WHERE lavoropi.settore>stadio.nsettori
```

5.1.3 Query parametriche

1. Mostra il numero di operatori che hanno lavorato in un determinato evento, riportando il ruolo e relativo conteggio, chiedendo in input il valore numerico del codice seriale dell'evento.

```
SELECT o.ruolo, COUNT(l.codiceoperatore) AS num
FROM lavorooperatori l
    JOIN operatore o ON (l.codiceoperatore = o.cf)
    JOIN evento e ON (l.codiceevento = e.codice)
WHERE e.codice = 1
GROUP BY o.ruolo
UNION
SELECT 'Pronto Intervento' as ruolo, COUNT(l.codiceoperatore) AS num
FROM lavoropi l
    JOIN prontointervento o ON (l.codiceoperatore = o.cf)
    JOIN evento e ON (l.codiceevento = e.codice)
WHERE e.codice = 1
GROUP BY ruolo
ORDER BY num ASC;
```

2. Mostra la lista di tutti gli eventi organizzati in un determinato stadio, riportando nome e data della manifestazione, chiedendo in input il codice di tre caratteri identificativo dello stadio.

```
SELECT e.nome AS evento, e.data
FROM evento e JOIN stadio s ON (e.stadio = s.codice)
WHERE s.codice = 'SGM';
```

3. Chiedendo in input il codice di uno stadio e una data, mostra la lista di tutte le persone daspate da quello stadio con pena valida nella data in questione, riportando nome e cognome dei diffidati e il loro codice fiscale.

```sql
SELECT s.cf as codicefiscale, s.nome, s.cognome
FROM segnalazione sz
    JOIN segnalato s ON (sz.codicesegnalato = s.cf)
    JOIN evento e ON (sz.codiceevento = e.codice)
    JOIN stadio ON (e.stadio = stadio.codice)
WHERE stadio.codice = 'SGM'
    AND sz.iniziodaspo <= '2023-12-31'
    AND sz.finedaspo >= '2023-12-31';
```

4. Mostra il numero di eventi in cui ha lavorato un operatore, riportando il nome dello stadio e il relativo conteggio, richiedendo in input il codice fiscale dell'operatore interessato.

```sql
SELECT stadio.nome AS stadio, COUNT(e.codice) AS num
FROM lavorooperatori l
    JOIN evento e ON (l.codiceevento = e.codice)
    JOIN stadio ON (e.stadio = stadio.codice)
WHERE l.codiceoperatore = 'DGSGLI96P18C510I'
GROUP BY stadio.nome;
```

5.2 Indici

Un dato che realisticamente verrebbe periodicamento controllato e aggiornato è sicuramente lo stipendio orario da corrispondere ad uno steward. Inoltre, considerando sia la frequenza con cui si andrebbe a controllare la lista di operatori assegnati ad un evento, sia la grandezza del risultato aspettato, indicizzare il codice operatore e il codice evento nelle tabelle relative alle assegnazioni operatore-evento-settore sarebbe utile. Per questo motivo all'interno del file SQL relativo alla base di dati si definiscono i seguenti indici:

```sql
CREATE INDEX stipendio_idx ON Operatore (Stipendio);

CREATE INDEX lavoroop_idx ON LavoroOperatori (CodiceOperatore, CodiceEvento);

CREATE INDEX lavoropi_idx ON LavoroPI (CodiceOperatore, CodiceEvento);
```

6 Codice C++

6.1 Descrizione utilizzo del codice

Il codice C++ per l'esecuzione delle query consiste in un unico file *query.cpp*. La compilazione e l'esecuzione varia dal sistema operativo in utilizzo, di seguito è descritta la procedura in un sistema Unix/Linux.

1. Creare la cartella *dependencies/include* all'interno della cartella in cui è salvato il file *query.cpp*;
2. Copiare in essa i file *libpq-fe.h*, *pg_config_ext.h*, *postgres_ext.h* che solitamente si trovano nella cartella */usr/include/postgres*;
3. Compilare con il comando *g++ query.cpp -o query -I /usr/include/postgresql -lpq* assicurandosi di essere all'interno della cartella corretta;
4. Eseguire il file *query*.

All'esecuzione il programma chiederà di inserire i dati necessari per stabilire una connessione con la base di dati. Dopo essersi connessi al database apparirà un semplice menù dal quale è possibile scegliere la query che si vuole visualizzare. I comandi dal 5 all'8 eseguono le query parametriche e stampano a video delle tabelle per facilitare l'inserimento dei dati necessari. I comandi 9 e 10, invece, eseguono le query per i controlli dei vincoli che non possono essere espressi dai vincoli di integrità referenziale della base di dati.

6.2 Documentazione del codice

Le funzioni utilizzate all'interno del codice sono le seguenti:

- ```
  PGconn* connecttodb();
  ```
  Chiede all'utente di inserire i dati per stabilire una connessione con la base di dati. Se la connessione è stata creata la restituisce altrimenti stampa un messaggio di errore e termina il programma.
- ```
  int menu();
  ```
  Stampa il menu e chiede all'utente quale query voler visualizzare restituendo il valore scelto.
- ```
  void execquery(PGconn* conn, string query, bool control = false, const char** value = NULL, int npar = 0);
  ```
  Eseguisce la query passata come parametro. Control è true se la query è di controllo, value è l'array dei valori dei parametri della query e npar è il numero di parametri della query.
- ```
  void checkResults(PGresult* res , const PGconn* conn);
  ```
  Controlla se il risultato di una query ha prodotto errori o meno e in caso affermativo stampa un messaggio di errore.
- ```
  void printquery(PGresult* res);
  ```
  Stampa il risultato di una query.
- ```
  void printcontrolquery(PGresult* res);
  ```
  Stampa se una query di controllo ha prodotto un risultato corretto o meno.