Progetto basi di dati modulo 2

Davide Belcastro,1962536

27 febbraio 2023

Sommario

1 Indice

2 Specifiche progetto: Corsi di Formazione

Il progetto consiste in una base dati di un'applicazione che deve poter gestire corsi di formazione.
I dipendenti della società che utilizzano l'applicazione devono poter inserire i docenti che insegnano i corsi,gli allievi che seguono i corsi,registrare le presenze,visualizzare i corsi tenuti in un certo periodo dell'anno,visualizzare le ore svolte da un allievo, etc..
Più nel dettaglio,la società organizza progetti,ogni progetto è formato da 1 o più corsi.
Ogni progetto ha una data di inizio,data di fine,un nome, un insieme di allievi iscritti al progetto e un insieme di corsi.
Un corso deve essere svolto nel periodo compreso tra inizio e fine del progetto a cui fa parte.
All'interno della società lavorano dipendenti che possono essere nominati responsabili di 0 o più progetti, per cui per ogni progetto è necessario sapere anche il responsabile.
I corsi vengono assegnati ad uno o più docenti.
I docenti non sono dipendenti della società ma l'applicazione deve comunque memorizzare i dati anagrafici dei docenti,oltre che dei dipendenti.
Dei corsi è necessario memorizzare,oltre che i docenti,il nome e il progetto.
Ogni progetto viene finanziato da un ente, perciò di ogni progetto dobbiamo memorizzare: nome,data inizio,data fine,corsi, allievi,responsabile,ente.
Una persona può essere solo uno tra: dipendente,docente,allievo.
Gli allievi sono persone che possono essere iscritte a 0 o più progetti,bisogna sapere per ogni allievo i dati anagrafici e i progetti a cui è iscritto.
Le lezioni dei corsi vengono fatte da un docente scelto tra uno dei docenti a cui è stato assegnato quel corso.
E' opportuno memorizzare tutte le lezioni che un corso ha svolto e/o che dovrà svolgere.
Ogni lezione viene svolta in un aula che non deve già prevedere una lezione in quella fascia oraria.
Una lezione si considera svolta quando il docente di quella lezione conferma la sua presenza.

Alle lezioni sono presenti, oltre che il docente,gli allievi.
Quest'ultimi devono quindi essere assegnati ad uno o più progetti e devono seguire tutte le lezioni dei corsi di quei progetti.
Anche gli allievi devono confermare la loro presenza.
Per ogni aula è opportuno memorizzare indirizzo e nome.
I progetti vengono finanziati da un ente finanziatore, di cui è necessario sapere(oltre che tutti i progetti che finanzia), il nome e l'id.
Sia gli allievi che i docenti quando confermano una presenza devono inserire ora entrata,ora uscita.
Per dati anagrafici si intende:nome,cognome,codice fiscale,cittanascita,cittaresidenza,data nascita,etc..).
I dipendenti della società devono memorizzare anche la partita iva(univoca per ogni dipendente).
I docenti possono avere 0 o più link a piattaforme social in modo da poter controllare ogni docente su più piattaforme: Facebook,Instagram,Linkedin,Twitter,etc..
Di sotto vengono illustrate anche tutte le funzionalità che il sistema deve permettere.

3 Soddisfacimento dei requisiti

A	9 entità escluse relazioni ISA e/o generalizzazioni.
B	Nello schema è presente almeno una generalizzazione e relazione IS-A.
C	Vedendo lo schema ER come un grafo si possono individuare dei cicli.
D	Sono presenti relazioni tra entità con cardinalità diversa da quella predefinita (0,N).
E	Lo schema ER presenta degli attibuti multi-valore e degli attributi facoltativi.
F	Nella sezione "vincoli esterni" sono presenti 6 vincoli esterni non realizzabili graficamente sullo schema ER (Sezione "Vincoli Esterni").
G	Nelle specifiche dello schema ER è presente un'indicazione dei volumi per le varie entità e relazioni (Sezione "Tavola dei volumi").
H	Nelle specifiche dello schema ER è presnete un'indicazione sul carico di lavoro delle query e delle operazioni (Sezione "Tavola delle operazioni") per un numero complessivo di 24 tra Query e operazioni comuni.

4 Progettazione concettuale

4.1 Diagramma ER

Il diagramma ER è illustrato nella Figura 1

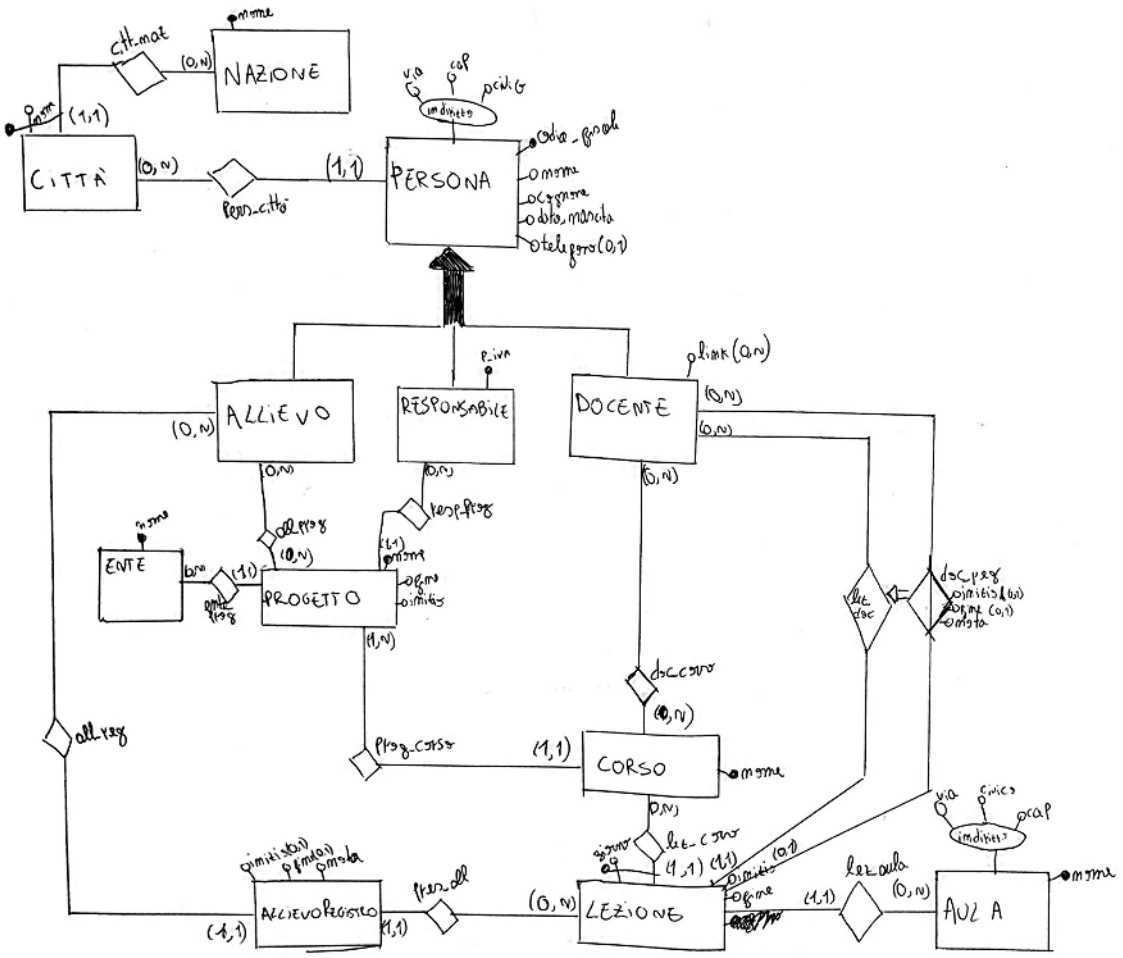

Figura 1: Diagramma ER

4.2 Glossario dei termini

Termine	Descrizione	Sinonimi	Collegamenti
Progetto	Organizzato dalla società, formato da un insieme di corsi		Ente,Allievo, Responsabile,Corso
Persona	Ogni istanza di Persona rappresenta una persona con i suoi dati anagrafici,ogni persona può essere solo uno tra: Allievo,Responsabile,Docente.		Città,Allievo, Responsabile,Docente
Docente	Rappresenta una persona esterna alla società che svolge le lezioni dei corsi a cui è stato assegnato.		Persona,Corso,Lezione
Responsabile	Rappresenta una persona(dipendente della società) che è responsabile di 0 o più progetti.	dipendente	Progetto,Persona
Allievo	Rappresenta una persona esterna alla società che può avere 1 o più progetti a cui fa parte e partecipa alle lezioni dei corsi appartenenti ai progetti a cui fa parte.		Persona,Progetto, AllievoRegistro
Ente	Rappresenta una società che finanzia progetti.	Ente finanziatore	Progetto
Corso	Viene organizzato dalla società ed è assegnato ad un progetto, può avere 1 o più docenti.		Progetto,Lezione,Docente
Aula	Luogo in cui si svolgono le lezioni dei corsi.		Lezione
Lezione	Memorizza le informazioni di una lezione relativa ad un corso,deve essere presente solo un docente (scelto tra i docenti disponibili tra quelli del corso), ci possono essere 0 o più allievi presenti.		AllievoRegistro,Corso, Docente,Aula
AllievoRegistro	Un istanza rappresenta le informazioni di un allievo ad una lezione,vengono memorizzate le ore di arrivo e uscita dell'allievo(se presente),se assente viene memorizzato solo 'assente'.		Lezione,Allievo
Città	Ogni istanza memorizza il nome di una città.		Nazione
Nazione	Ogni istanza memorizza il nome di una nazione.		Città

4.3 Dizionario dei dati

Entità/Relazione	Attributi	Identificatori
Persona(E)	nome/stringa;codice-fiscale/stringa;cognome/stringa;data-nascita/data;telefono/stringa(0,1);indirizzo/Indirizzo	codice-fiscale
Città(E)	nome/stringa	
Nazione(E)	nome/stringa	nome
Docente(E)	link/stringa(0,N)	
Allievo(E)		
Responsabile(E)	p-iva/stringa	p-iva
Progetto(E)	nome/stringa;inizio/data;fine/data	nome
Ente(E)	nome/stringa	nome
Corso(E)	nome/stringa	nome
Lezione(E)	giorno/data;inizio/time;fine/time	
Aula(E)	nome/stringa;indirizzo/Indirizzo	nome
AllievoRegistro(E)	inizio(0,1)/time;fine(0,1)/time;nota/Nota	
lez-aula(R)		
doc-reg(R)	inizio(0,1)/time;fine(0,1)/time;nota/Nota	
lez-doc(R)		
lez-corso(R)		
pres-all(R)		
all-reg(R)		
doc-corso(R)		
prog-corso(R)		
all-prog(R)		
resp-prog(R)		
ente-prog(R)		
pers-citta(R)		
cit-naz(R)		

4.4 Domini non standard

Nota:Presente,Assente
Indirizzo:via/stringa,civico/integer,cap/stringa

4.5 Vincoli

4.5.1 Vincoli su Entità/Relationship

1. l'attributo inizio deve essere minore dell'attributo fine su Entità Progetto e Lezione
2. in doc-reg e AllievoRegistro: l'attributo inizio deve essere minore dell'attributo fine and(inizio and fine != Null) se e solo se nota = Presente

4.5.2 Vincoli Esterni

1. Sia c l'istanza di Corso e l istanza di Lezione tale che (l,c) appartengono a lez-corso.
   Sia p l'istanza di Progetto tale che (p,c) appartiene a prog-corso.
   l'attributo "giorno" di l deve essere compreso tra inizio e fine di p.
2. Ogni istanza di Lezione deve avere un aula non occupata(senza una lezione in quella fascia oraria)
3. Sia (x,l) appartenente a doc-reg tale che x è istanza di Docente e l è istanza di Lezione, inizio e fine di doc-reg devono essere compresi tra inizio e fine di l.
4. Ogni istanza di lez-doc deve avere un docente e una lezione tale che,sia c il corso della lezione allora la coppia docente,c deve appartenere a doc-corso
5. Ogni istanza di AllievoRegistro deve avere un allievo a, e una lezione l tale che, sia c il corso della lezione l allora (a,c) appartiene ad all-prog

6. Sia (reg,l) appartenente a pres-all tale che reg è istanza di AllievoRegistro e l è istanza di Lezione, inizio e fine di reg devono essere compresi tra inizio e fine di l

4.6 Tavola dei volumi

OSS: Le previsioni hanno una stima per eccesso.

Nome	Volume	Spiegazione
Persona(E)	335	Il numero è calcolato sulla somma della stima di Docente,Allievo,Responsabile
Città(E)	4416	4416 è il numero di città nel mondo
Nazione(E)	193	193 è il numero di nazioni nel mondo
Docente(E)	20	Si è stimato che il numero dei docenti che lavoreranno per la società non supererà 20
Allievo(E)	300	Il numero di allievi previsti non dovrebbe superare il 300
Responsabile(E)	15	15 è il numero di dipendenti della società
Progetto(E)	50	La società crede di non organizzare più di 50 progetti
Ente(E)	50	Sicuramente non ci possono essere più enti memorizzati rispetto ai progetti
Corso(E)	300	Il numero di corsi deve essere maggio del numero dei progetti,con una stima di 6 corsi a progetto possiamo stimare 300 corsi.
Lezione(E)	90000	Stima calcolata moltiplicando Allievi * Corsi
Aula(E)	1324800	Stima calcolata moltiplicando 300*Città,stimando di non memorizzare più di 300 aule per città
AllievoRegistro(E)	90000	Stessa previsione di Lezione
lez-aula(R)	90000	Ogni lezione ha una sola aula
doc-reg(R)	90000	Ogni lezione ha un solo docente
lez-doc(R)	90000	Ogni lezione ha un solo docente
lez-corso(R)	90000	Ogni lezione ha un solo corso
pres-all(R)	90000	Stesso numero di AllievoRegistro
all-reg(R)	90000	Stesso numero di AllievoRegistro
doc-corso(R)	15000	Previsioni Docente * previsioni Corso
prog-corso(R)	300	Stesso numero di Corso
all-prog(R)	750	Previsione Allievo*previsione Progetto
resp-prog(R)	50	Previsione di Progetto
ente-prog(R)	50	Previsione di Progetto
pers-citta(R)	335	Previsione di Persona
cit-naz(R)	4416	Previsione di Città

4.7 Use case

Il sistema deve offrire le seguenti funzionalità ai gestori dell'applicazione

1. Restituire tutti gli allievi nati a Roma che hanno fatto almeno 5 ore totali presenze
2. Ritornare il numero di lezioni dei docenti nati a Roma
3. Ritornare il numero di progetti per ogni responsabile
4. Ritornare il numero di ore svolte da ogni allievo
5. Restituire tutti gli allievi presenti ad una lezione di un determinato corso
6. Ritornare il numero di ore previste per ogni corso
7. Ritornare il numero di ore previste per ogni progetto
8. Ritornare il numero di ore svote da un corso
9. Restituire il calendario di un corso
10. Restituire il calendario di un progetto
11. Restituire tutti i docenti di un corso
12. Restituire tutti i docenti di un progetto
13. Restituire tutti gli allievi di un progetto
14. Restituire tutti gli allievi di un corso
15. Restituire tutte le presenze di un docente
16. Restituire tutte le presenze di un allievo
17. Restituire tutte le lezioni svolte in un aula
18. Restituire tutti gli allievi presenti ad una lezione
19. Restituire i nomi delle aule in cui sono state tenute lezioni da docenti che hanno almeno 2 link
20. Inserimento presenza di un allievo
21. Inserimento presenza di un docente
22. Assegnare un docente ad un corso
23. Assegnare un allievo ad un progetto
24. Registrare una persona

4.7.1 Stima delle Operazioni

OSS:La stima viene effettuata per eccesso.

Operazione	Frequenza	Tipo
1	600 volte all'anno,stimando 2 volte ad allievo	S
2	150 volte all'anno,stimando 3 volte per ogni docente	S
3	30 volte all'anno,stimando 2 volte per ogni responsabile	S
4	15000 volte all'anno,stimando 50 volte ad allievo	S
5	90000 volte all'anno,stimando 1 volta per ogni lezione	S
6	900 volte all'anno,stimando 3 volte per ogni corso	S
7	150 volte all'anno,stimando 3 volte per ogni progetto	S
8	10000 volte all'anno,considerando che 90000 è il numero di lezioni stimate per eccesso	S
9	300 volte all'anno	S
10	50 volte all'anno	S
11	300 volte all'anno	S
12	50 volte all'anno	S
13	50 volte all'anno	S
14	300 volte all'anno	S
15	18250 volte all'anno, 365*docenti previsti	S
16	109500 volte all'anno, 365*allievi previsti	S
17	1450656000 volte l'anno,previsioni aule*previsioni di lezioni all'anno	S
18	1095 volte l'anno,365*3,stimando 3 lezioni al giorno	S
19	40 volte l'anno,2 volte per ogni docente,operazione di poco interesse per la società	S
20	109500 volte l'anno,365*previsione allievi	I
21	7300 volte l'anno,365*previsione docenti	I
22	6000 volte l'anno	I
23	15000 volte l'anno	I
24	80 volte l'anno	I

4.7.2 Tavola degli accessi

Op1

Concetto	Costrutto	Tipo accesso
Allievo	Entità	L
Persona	Entità	L
Città	Entità	L

Op2

Concetto	Costrutto	Tipo accesso
Docente	Entità	L
Persona	Entità	L
Città	Entità	L

Op3

Concetto	Costrutto	Tipo accesso
Responsabile	Entità	L
Persona	Entità	L
Progetto	Entità	L

Op4

Concetto	Costrutto	Tipo accesso
Responsabile	Entità	L
Persona	Entità	L
AllievoRegistro	Entità	L

Op5

Concetto	Costrutto	Tipo accesso
Responsabile	Entità	L
Persona	Entità	L
AllievoRegistro	Entità	L
Lezione	Entità	L

Op6

Concetto	Costrutto	Tipo accesso
lez-corso	Relazione	L
Corso	Entità	L
Lezione	Entità	L

Op7

Concetto	Costrutto	Tipo accesso
lez-prog	Relazione	L
Progetto	Entità	L
Lezione	Entità	L
Corso	Entità	L

Op8

Concetto	Costrutto	Tipo accesso
doc-reg	Relazione	L
Lezione	Entità	L

Op9

Concetto	Costrutto	Tipo accesso
Persona	Entità	L
Lezione	Entità	L
Aula	Entità	L

Op10

Concetto	Costrutto	Tipo accesso
Persona	Entità	L
Lezione	Entità	L
Aula	Entità	L
Corso	Entità	L

Op11

Concetto	Costrutto	Tipo accesso
Persona	Entità	L
doc-corso	Relazione	L

Op12

Concetto	Costrutto	Tipo accesso
Persona	Entità	L
doc-corso	Relazione	L
Corso	Relazione	L

Op13

Concetto	Costrutto	Tipo accesso
Persona	Entità	L
all-prog	Relazione	L

Op14

Concetto	Costrutto	Tipo accesso
Persona	Entità	L
all-prog	Relazione	L
Corso	Entità	L

Op15

Concetto	Costrutto	Tipo accesso
Lezione	Entità	L
doc-reg	Relazione	L
Corso	Entità	L

Op16

Concetto	Costrutto	Tipo accesso
Lezione	Entità	L
AllievoRegistro	Entità	L
Corso	Entità	L

Op17

Concetto	Costrutto	Tipo accesso
Lezione	Entità	L
Corso	Entità	L

Op18

Concetto	Costrutto	Tipo accesso
AllievoRegistro	Entità	L
Persona	Entità	L

Op19

Concetto	Costrutto	Tipo accesso
Aula	Entità	L
Lezione	Entità	L
link	Relazione	L

Op20

Concetto	Costrutto	Tipo accesso
all-prog	Relazione	S

Op21

Concetto	Costrutto	Tipo accesso
doc-corso	Relazione	S

Op22

Concetto	Costrutto	Tipo accesso
Persona	Entità	S

Op23

Concetto	Costrutto	Tipo accesso
AllievoRegistro	Entità	S

Op24

Concetto	Costrutto	Tipo accesso
doc-reg	Relazione	S

5 Ristrutturazione dello schema

5.1 Modifiche al diagramma ER

1. Eliminazione delle gerarchie isa: il modello relazionale non rappresenta le gerarchie, e quindi vanno sostituite da entità e relationship.(Aggiunte le relationship isDoc,isAll,isResp per eliminare la gerarchia isa tra Persona e Docente,Allievo,Responsabile)
2. Eliminare attributi multi valore aggiungendo un ulteriore entità(Aggiunta l'entità ValoreLink)
3. Ogni Entità deve avere un identificativo univoco(Aggiunto un identificatore primario per ogni entità)

5.2 Diagramma ER ristrutturato

Il diagramma ER ristrutturato è illustrato in figura 2

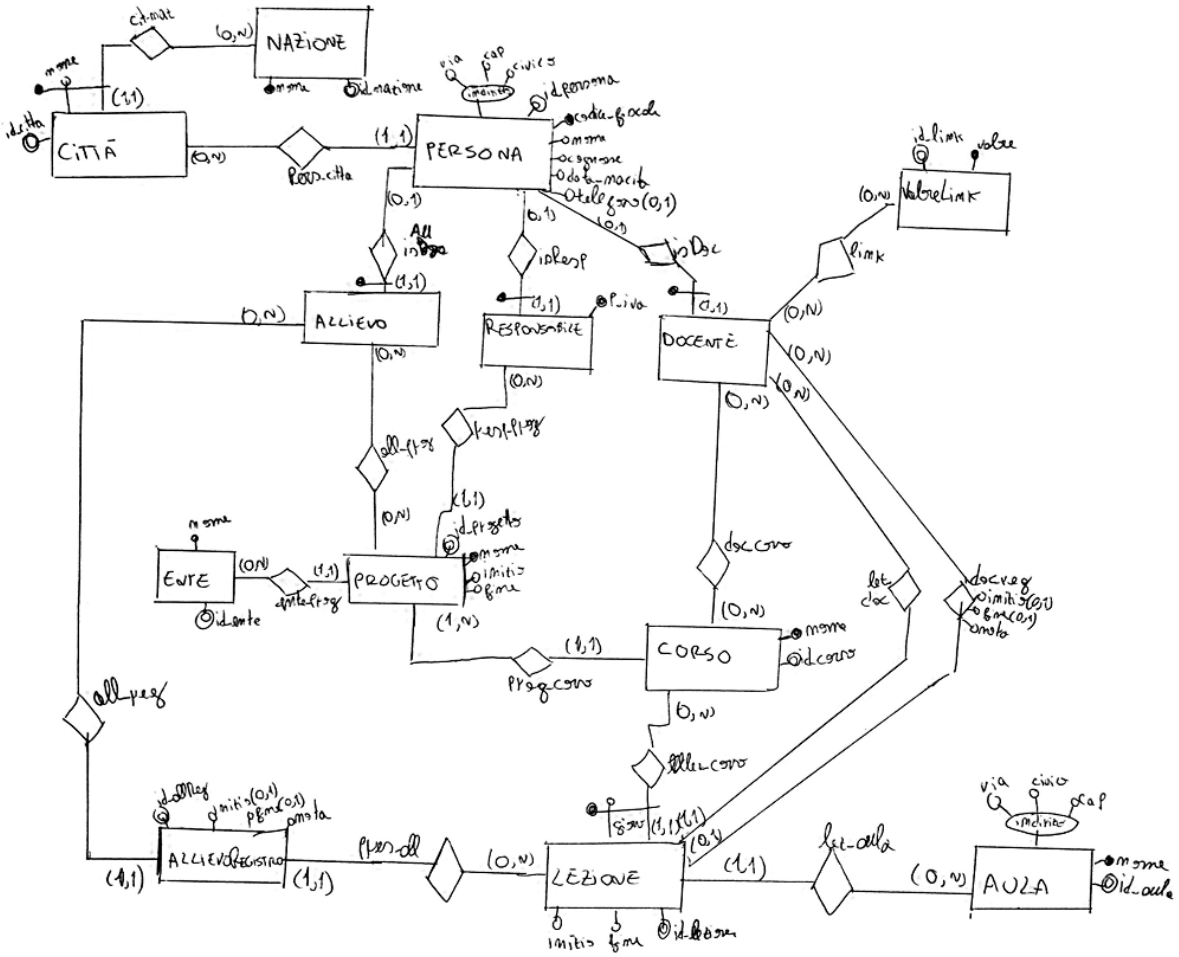

Figura 2: Diagramma ER ristrutturato

5.3 Dizionario dei dati

Entità	Attributi	Identificatori primari	Identificatori secondari
Persona(E)	nome/varchar(20);codice-fiscale/varchar(25); cognome/varchar(25);data-nascita/date;telefono/varchar(10)(0,1); indirizzo/Indirizzo;id-persona/integer auto-increment	id-persona	codice-fiscale
Città(E)	nome/varchar(10);id-citta/integer auto-increment	id-citta	
Nazione(E)	nome/varchar(15);id-nazione/integer auto-increment	id-nazione	nome
Docente(E)			
ValoreLink(E)	valore/varchar(25);id-link/integer auto-increment	id-link	valore
Allievo(E)			
Responsabile(E)	p-iva/varchar(25)		p-iva
Progetto(E)	nome/varchar(25);inizio/date;fine/date;id-progetto/integer auto-increment	id-progetto	nome
Ente(E)	nome/varchar(25);id-ente/integer auto-increment	id-ente	nome
Corso(E)	nome/varchar(25);id-corso/integer auto-increment	id-corso	nome
Lezione(E)	giorno/date;inizio/time;fine/time;id-lezione/integer auto-increment	id-lezione	
Aula(E)	nome/varchar(25);indirizzo/Indirizzo;id-aula/integer auto-increment	id-aula	nome
AllievoRegistro	inizio(0,1)/time;fine(0,1)/time;nota/Nota;id-allReg/integer auto-increment	id-allReg	
lez-aula(R)			
link(R)			
doc-reg(R)	inizio(0,1)/time;fine(0,1)/time;nota/Nota		
lez-doc(R)			
lez-corso(R)			
pres-all(R)			
all-reg(R)			
doc-corso(R)			
prog-corso(R)			
all-prog(R)			
resp-prog(R)			
ente-prog(R)			
pers-citta(R)			
cit-naz(R)			
isAll(R)			
isDoc(R)			
isResp(R)			

5.4 Domini non standard

create type Nota as enum('Presente','Assente');
create type Indirizzo as(via varchar(50), civico integer, citta varchar(25));

5.5 Tavola dei volumi

OSS: Le previsioni hanno una stima per eccesso.

Nome	Volume	Spiegazione
Persona(E)	335	Il numero è calcolato sulla somma della stima di Docente,Allievo,Responsabile
Città(E)	4416	4416 è il numero di città nel mondo
Nazione(E)	193	193 è il numero di nazioni nel mondo
Docente(E)	20	Si è stimato che il numero dei docenti che lavoreranno per la società non supererà 20
ValoreLink(E)	350	Previsione Docente * 7,stimando che ogni docente non abbia più di 7 link.
Allievo(E)	300	Il numero di allievi previsti non dovrebbe superare il 300
Responsabile(E)	15	15 è il numero di dipendenti della società
Progetto(E)	50	La società crede di non organizzare più di 50 progetti
Ente(E)	50	Sicuramente non ci possono essere più enti memorizzati rispetto ai progetti
Corso(E)	300	Il numero di corsi deve essere maggio del numero dei progetti,con una stima di 6 corsi a progetto possiamo stimare 300 corsi.
Lezione(E)	90000	Stima calcolata moltiplicando Allievi * Corsi
Aula(E)	1324800	Stima calcolata moltiplicando 300*Città,stimando di non memorizzare più di 300 aule per città
AllievoRegistro(E)	90000	Stessa previsione di Lezione
lez-aula(R)	90000	Ogni lezione ha una sola aula
link(R)	17500	Previsione ValoreLink * previsione Docente
doc-reg(R)	90000	Ogni lezione ha un solo docente
lez-doc(R)	90000	Ogni lezione ha un solo docente
lez-corso(R)	90000	Ogni lezione ha un solo corso
pres-all(R)	90000	Stesso numero di AllievoRegistro
all-reg(R)	90000	Stesso numero di AllievoRegistro
doc-corso(R)	15000	Previsioni Docente * previsioni Corso
prog-corso(R)	300	Stesso numero di Corso
all-prog(R)	750	Previsione Allievo*previsione Progetto
resp-prog(R)	50	Previsione di Progetto
ente-prog(R)	50	Previsione di Progetto
pers-citta(R)	335	Previsione di Persona
cit-naz(R)	4416	Previsione di Città
isAll(R)	300	Previsione di Allievo
isDoc(R)	50	Previsione di Docente
isResp(R)	15	Previsione di Responsabile

5.6 Vincoli modificati

5.6.1 Vincoli Esterni

1. Sia l l'istanza di Lezione tale che corso(l) = c.
   Sia p l'istanza di Progetto tale che progetto(c) = p.
   l'attributo "giorno" di l deve essere compreso tra inizio e fine di p.

2. Ogni istanza di Lezione deve avere un aula non occupata(senza una lezione in quella fascia oraria)

3. Sia (x,l) appartenente a doc-reg tale che x è istanza di Docente e l è istanza di Lezione, inizio e fine di doc-reg devono essere compresi tra inizio e fine di l.

4. Ogni istanza di lez-doc deve avere un docente d e una lezione l tale che,sia c il corso di l allora la coppia (d,c) appartiene a doc-corso.

5. Ogni istanza di AllievoRegistro deve avere un allievo "a" e una lezione "l" tale che, sia c il corso di l e p il progetto di c allora (a,p) deve appartenere ad all-prog.

6. Sia (reg,l) appartenente a pres-all tale che reg è istanza di AllievoRegistro e l è istanza di Lezione, inizio e fine di reg devono essere compresi tra inizio e fine di l

7. Ogni istanza di doc-reg deve avere un docente d e una lezione l tale che (d,l) appartiene a lez-doc

8. Ogni Persona P è in relationship in una e una sola tra: isDoc,isAll,isResp

6 Traduzione diretta al modello relazionale

6.1 Schema logico

Procedura per creare lo schema logico dell'applicazione:
-le entità le trasformo in relazioni
-le relationship si traducono in relazioni e vincoli di chiave esterna,oppure le accorpo con una delle entità coinvolte qualora il vincolo d cardinalità massima tra la relationship e l'entità sia uguale ad 1
-vincoli di cardinalità si traducono in vincoli di chiave,vincoli di chiave esterna o vincoli esterni.

NB: gli attributi in maiuscolo rappresentano le chiavi
Gli asterisco indicano che può essere NULL
Qui risolvo tutti i vincoli interni(vincoli di ennupla,di chiave,etc..)

1. Persona(nome:varchar(20),CODICE-FISCALE:varchar(25),cognome:varchar(25),data-nascita:date,
   telefono:varchar(10)*,indirizzo:Indirizzo,ID-PERSONA:integer auto-increment,id-citta:integer)
   primary key(id-persona)
   unique(codice-fiscale)
   foreign key id-citta references Citta(id-citta)

2. Città(nome:varchar(10),ID-CITTA:integer auto-increment,id-nazione:integer)
   primary key(id-citta)
   unique(nome,id-nazione)
   foreign key id-nazione references Nazione(id-nazione)

3. Nazione(NOME:varchar(15),ID-NAZIONE:integer auto-increment)
   primary key(id-nazione)
   unique(nome)

4. Docente(ID-DOCENTE:integer auto-increment)
   primary key(id-docente)
   foreign key id-docente references Persona(id-persona)

5. ValoreLink(VALORE:varchar(25),ID-LINK:integer auto-increment)
   unique(valore)
   primary key(id-link)

6. Allievo(ID-ALLIEVO:integer)
   primary key(id-allievo)
   foreign key id-allievo references Persona(id-persona)

7. Responsabile(ID-RESPONSABILE:integer,P-IVA/varchar(25))
   primary key(id-responsabile)
   unique(p-iva)
   foreign key id-responsabile references Persona(id-persona)

8. Progetto(NOME:varchar(25),inizio:date,fine:date,ID-PROGETTO:integer auto-increment,
   id-ente:integer,id-responsabile:integer)
   primary key(id-progetto)
   unique(nome)
   foreign key id-ente references Ente(id-ente)
   foreign key id-responsabile references Responsabile(id-responsabile)
   check (inizio ¡ fine)

9. Ente(NOME:varchar(25),ID-ENTE:integer auto-increment)
   primary key(id-ente)
   unique(nome)

10. Corso(NOME:varchar(25),ID-CORSO:integer auto-increment,id-progetto: integer)
    primary key(id-corso)
    unique(nome)
    foreign key id-progetto references Progetto(id-progetto)

11. Lezione(ID-LEZIONE:integer auto-increment,giorno:date,inizio:time,fine:time,id-docente:integer,id-
    aula:integer,id-corso:integer)
    unique(giorno,id-corso)
    primary key(id-lezione)
    foreign key id-aula references Aula(id-aula)
    foreign key id-corso references Corso(id-corso)
    foreign key id-docente references Docente(id-docente)
    check (inizio 'minore' fine)

12. Aula(ID-AULA:integer auto-increment,NOME: varchar(25),indirizzo:Indirizzo)
    primary key(id-aula)
    unique(nome)

13. AllievoRegistro(ID-ALL-REG:integer auto-increment,inizio*:time,fine*:time,nota:Nota,id-allievo:integer,
    id-lezione:integer)
    primary key(id-all-reg)
    foreign key id-allievo references Allievo(id-allievo)
    foreign key id-lezione references Lezione(id-lezione)
    check (inizio 'minore' fine and (inizio != null and fine != null se e solo se nota = Presente))

14. link(ID-LINK:integer,ID-DOCENTE:integer)
    primary key(id-link,id-docente)
    foreign key id-link references ValoreLink(id-link)
    foreign key id-docente references Docente(id-docente)

15. doc-reg(ID-LEZIONE:integer,inizio*:time,fine*:time,nota:Nota,id-lezione:integer)
    primary key(id-lezione)
    foreign key id-lezione references Lezione(id-lezione)
    check (inizio ¡ fine and (inizio != null and fine != null se e solo se nota = Presente))

16. doc-corso(ID-DOCENTE:integer,ID-CORSO:integer)
    primary key(id-docente,id-corso)
    foreign key id-docente references Docente(id-docente)
    foreign key id-corso references Corso(id-corso)

17. all-prog(ID-ALLIEVO:integer,ID-PROGETTO:integer)
    primary key(id-allievo,id-progetto)
    foreign key id-allievo references Allievo(id-allievo)
    foreign key id-progetto references Progetto(id-progetto)

6.2 Specifica del carico dell'applicazione

Analizzando le tabelle della base dati si può notare come il carico di lavoro maggiore è incentrato nelle tabelle Lezione,AllievoRegistro e doc-reg essendo tabelle che prevedono l'inserimento di lezioni giornaliere per ogni corso;
AllievoRegistro prevede l'inserimento della presenza/assenza di ogni allievo per ogni lezione del corso a cui è iscritto, mentre doc-reg prevede la memorizzazione della presenza/assenza del docente ad ogni lezione che deve svolgere.
Lezione prevede l'inserimento di tutti i calendari dei corsi.
Infine,tutte e tre, sono necessarie per ricavare le informazioni riguardo: -ore svolte da un docente
-ore svolte da un allievo
-ore previste/svolte di un corso
-ore di assenza di un docente/allievo
-etc...
Di conseguenza prevedono un numero elevato di accessi in scrittura e lettura.

7 Specifica del database in SQL

- Creazione del database:
  CREATE DATABASE corsi-formazione;

- Creazione dei domini non standard:
  create type Nota as enum('Presente','Assente');
  create type Indirizzo as(via varchar(50), civico integer, citta varchar(25));

- Creazione delle tabelle del database:
  La creazione delle tabelle è illustrata nelle figure 3,4,5,6,7 e 8.

7.1 Triggers

Implementazione trigger per risolvere i seguenti vincoli esterni(i vincoli interni sono stati risolti nella creazione delle tabelle aggiungendo vincoli di ennupla/chiave/etc..)

1. Sia l l'istanza di Lezione tale che id-corso(l) = c.
   Sia p l'istanza di Progetto tale che id-progetto(c) = p.
   l'attributo "giorno" di l deve essere compreso tra inizio e fine di p. Vedere figura 9

2. Ogni istanza di Lezione deve avere un aula non occupata(senza una lezione in quella fascia oraria). Vedere figura 10

3. Sia (x,l) appartenente a doc-reg tale che x è istanza di Docente e l è istanza di Lezione, inizio e fine di doc-reg devono essere compresi tra inizio e fine di l. Vedere figura 11

4. Ogni istanza l di Lezione deve avere un docente d e un corso c tale che (d,c) appartiene a doc-corso. Vedere figura 12

5. Ogni istanza di AllievoRegistro deve avere un allievo "a" e una lezione l, tale che, sia c il corso di l e p il progetto di c, allora (a,p) appartiene a all-prog. Vedere figura 13

```sql
CREATE TABLE Nazione(
    nome varchar(15) NOT NULL,
    id_nazione integer AUTO_INCREMENT,
    primary key(id_nazione),
    unique(nome)
);
CREATE TABLE Città(
    nome varchar(10) NOT NULL,
    id_citta integer AUTO_INCREMENT,
    id_nazione integer NOT NULL,
    primary key(id_citta),
    unique(nome,id_nazione),
    foreign key (id_nazione) references Nazione(id_nazione)
    );

CREATE TABLE Persona(nome varchar(20) NOT NULL,
                     codice_fiscale varchar(25) NOT NULL,
                     cognome varchar(25) NOT NULL,
                     data_nascita date NOT NULL,
                     telefono varchar(10),
                     indirizzo varchar(25) NOT NULL,
                     ID_PERSONA integer AUTO_INCREMENT,
                     id_citta integer NOT NULL,
                     primary key(id_persona),
                     unique(codice_fiscale),
                     foreign key (id_citta) references Citta(id_citta)
                     );

 CREATE TABLE Docente(id_docente integer AUTO_INCREMENT,
    primary key(id_docente),
    foreign key (id_docente) references Persona(id_persona)
                      );

CREATE TABLE ValoreLink(valore varchar(25)NOT NULL,
                        id_link integer AUTO_INCREMENT,
    unique(valore),
    primary key(id_link)
        );

CREATE TABLE Allievo(id_allievo integer NOT NULL,
    primary key(id_allievo),
    foreign key (id_allievo) references Persona(id_persona)
          );
CREATE TABLE Responsabile(id_responsabile integer NOT NULL,
                          p_iva varchar(25) NOT NULL,
    primary key(id_responsabile),
    unique(p_iva),
    foreign key (id_responsabile) references Persona(id_persona)
          );

CREATE TABLE Ente(nome varchar(25) NOT NULL,
    id_ente integer AUTO_INCREMENT,
    primary key(id_ente),
    unique(nome));
```

```
CREATE TABLE Progetto(nome varchar(25) NOT NULL,
                      inizio date NOT NULL,
                      fine date NOT NULL,
                      id_progetto integer AUTO_INCREMENT,
                      id_ente integer NOT NULL,
                      id_responsabile integer NOT NULL,
    primary key(id_progetto),
    unique(nome),
    foreign key (id_ente) references Ente(id_ente),
    foreign key (id_responsabile) references Responsabile(id_responsabile)
                      );
ALTER TABLE Progetto CHECK(inizio < fine);

CREATE TABLE Corso(nome varchar(25) NOT NULL,id_corso integer AUTO_INCREMENT,id_progetto integer NOT NULL,
    primary key(id_corso),
    unique(nome),
    foreign key (id_progetto) references Progetto(id_progetto));

CREATE TABLE Lezione(id_lezione integer AUTO_INCREMENT,
                     giorno date NOT NULL,inizio time NOT NULL,fine time NOT NULL,id_docente integer NOT NULL,id_aula integer NOT NULL,id_corso integer NOT NULL,
    unique(giorno,id_corso),
    primary key(id_lezione),
    foreign key (id_aula) references Aula(id_aula),
    foreign key (id_corso) references Corso(id_corso),
    foreign key (id_docente) references Docente(id_docente));


ALTER TABLE Lezione CHECK(inizio < fine);
CREATE TABLE Aula(id_aula integer AUTO_INCREMENT,nome varchar(25) NOT NULL,indirizzo varchar(25) NOT NULL,
    primary key(id_aula),
    unique(nome));
```

Figura 4: Comandi SQL

```
CREATE TABLE `AllievoRegistro` (
  `id_all_reg` int(11) NOT NULL,
  `inizio` time DEFAULT NULL,
  `fine` time DEFAULT NULL,
  `nota` enum('Presente','Assente') NOT NULL,
  `id_allievo` int(11) NOT NULL,
  `id_lezione` int(11) NOT NULL,
  primary key (`id_all_reg`),
  UNIQUE(`id_allievo`,`id_lezione`) ,
  FOREIGN KEY (`id_allievo`) REFERENCES `Allievo` (`id_allievo`),
  FOREIGN KEY (`id_lezione`) REFERENCES `Lezione` (`id_lezione`)
);
```

Figura 5: Comandi SQL

```
ALTER TABLE AllievoRegistro CHECK(
    ( inizio != null and fine != null and nota='Presente' and inizio < fine ) OR
    (inizio = null and fine  = null and nota <> 'Presente')
    );
CREATE TABLE link(id_link integer NOT NULL,id_docente integer NOT NULL,
primary key(id_link,id_docente),
foreign key (id_link) references ValoreLink(id_link),
foreign key (id_docente) references Docente(id_docente));
```

Figura 6: Comandi SQL

```
CREATE TABLE doc_reg(id_lezione integer NOT NULL,
                     inizio time,
                     fine time,
                     nota ENUM('Presente','Assente') NOT NULL,
              primary key(id_lezione),
                     FOREIGN KEY(id_lezione) REFERENCES Lezione(id_lezione));
```

Figura 7: Comandi SQL

```
 ALTER TABLE doc_reg CHECK(
    ( inizio != null and fine != null and nota='Presente' and inizio < fine ) OR
    (inizio = null and fine  = null and nota <> 'Presente')
    );
CREATE TABLE  doc_corso(id_docente integer NOT NULL,id_corso integer NOT NULL,
     primary key(id_docente,id_corso),
     foreign key (id_docente) references Docente(id_docente),
     foreign key (id_corso) references Corso(id_corso)
     );
CREATE TABLE all_prog(id_allievo integer NOT NULL,id_progetto integer NOT NULL,
    primary key(id_allievo,id_progetto),
    foreign key (id_allievo) references Allievo(id_allievo),
    foreign key (id_progetto) references Progetto(id_progetto)
 );
```

Figura 8: Comandi SQL

```
CREATE TRIGGER `check_legal` BEFORE INSERT ON
`Lezione`
 FOR EACH ROW if (new.giorno < (select
Progetto.inizio from Progetto where
Progetto.id_progetto = (select Corso.id_progetto
from Corso where Corso.id_corso = NEW.id_corso)) or
   new.giorno >
   (select Progetto.fine from Progetto where
Progetto.id_progetto = (select Corso.id_progetto
from Corso where Corso.id_corso = NEW.id_corso)))
 then
   SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT =
'inserire valori legali di inizio e fine';

end if
```

Figura 9: Trigger 1

```
CREATE TRIGGER `check_value` BEFORE INSERT ON
`Lezione`
 FOR EACH ROW if (new.id_aula in ( select
pippo.id_aula from (select DISTINCT Lezione.id_aula
from Lezione where ( Lezione.giorno = new.giorno and
( (new.inizio <= Lezione.inizio and new.fine >
Lezione.inizio) or (new.inizio < Lezione.fine and
new.fine >= Lezione.fine) or (Lezione.inizio <
new.inizio and Lezione.fine > new.fine) ) ) ) as
pippo)) then

SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'aula
occupata!';
 end if
```

Figura 10: Trigger 2

6. Sia (reg,l) appartenente a pres-all tale che reg è istanza di AllievoRegistro e l è istanza di Lezione, inizio e fine di reg devono essere compresi tra inizio e fine di l. Vedere figura 14
7. Ogni Persona P è in relationship in una e una sola tra: isDoc,isAll,isResp. Vedere figura 15, 16 e 17

7.2 Use Case

Implementazione delle seguenty Query SQL per risolvere gli usecase

1. Restituire tutti gli allievi nati a Roma che hanno fatto almeno 5 ore totali presenze
2. Ritornare il numero di lezioni dei docenti nati a Roma
3. Ritornare il numero di progetti per ogni responsabile
4. Ritornare il numero di ore svolte da ogni allievo
5. Restituire tutti gli allievi presenti ad una lezione di un determinato corso
6. Ritornare il numero di ore previste per ogni corso

```
CREATE TRIGGER `check_legal2` BEFORE INSERT ON
`doc_reg`
 FOR EACH ROW if
(
new.inizio < (select inizio from Lezione where
Lezione.id_lezione = new.id_lezione)  or
 new.fine >
(select fine from Lezione where Lezione.id_lezione =
new.id_lezione)
)
THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'inserire
periodo legale';
end if
```

Figura 11: Trigger 3

```
CREATE TRIGGER `check_docente` BEFORE INSERT ON
`Lezione`
 FOR EACH ROW if new.id_docente not in(select
doc_corso.id_docente from doc_corso where
doc_corso.id_corso = new.id_corso)
THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'corso
non assegnato a quel docente';
end if
```

Figura 12: Trigger 4

```
CREATE TRIGGER `check_allievo_legal` BEFORE INSERT ON
`AllievoRegistro`
 FOR EACH ROW if
(
   ( select id_progetto from Progetto where id_progetto = (select
id_corso from Lezione where id_lezione = new.id_lezione)) not in
    (select all_prog.id_progetto from all_prog where
all_prog.id_allievo = new.id_allievo)




 )
 THEN
 SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'questo allievo non può
registrare la presenza!';

end if
```

Figura 13: Trigger 5

```
1  CREATE TRIGGER `check_legal3` BEFORE INSERT ON
   `AllievoRegistro`
2   FOR EACH ROW if
3  (
4  new.inizio < (select inizio from Lezione where
   Lezione.id_lezione = new.id_lezione)  or
5   new.fine >
6  (select fine from Lezione where Lezione.id_lezione =
   new.id_lezione)
7  )
8  THEN
9  SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'inserire
   periodo legale';
10 end if
```

Figura 14: Trigger 6

```
1  CREATE TRIGGER `check_allievo` BEFORE INSERT ON
   `Allievo`
2   FOR EACH ROW if (new.id_allievo in (SELECT
   id_responsabile from Responsabile) or new.id_allievo
   in (select id_docente from Docente))
3  THEN
4  SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'questa
   persona non può essere un allievo!';
5  end if
```

Figura 15: Trigger 7

```
1 CREATE TRIGGER `check_docente2` BEFORE INSERT ON
`Docente`
2  FOR EACH ROW if (new.id_docente in (SELECT
id_responsabile from Responsabile) or new.id_docente
in (select id_allievo from Allievo))
3 THEN
4 SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'questa
persona non può essere un docente!';
5 end if
```

Figura 16: Trigger 7

```
1 CREATE TRIGGER `check_resp` BEFORE INSERT ON
`Responsabile`
2  FOR EACH ROW if (new.id_responsabile in (SELECT
id_docente from Docente) or new.id_responsabile in
(select id_allievo from Allievo))
3 THEN
4 SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'questa
persona non può essere un responsabile!';
5 end if
```

Figura 17: Trigger 7

```sql
    create view allievi_roma as SELECT Allievo.id_allievo from Allievo,Persona,Città WHERE
    Allievo.id_allievo = Persona.id_persona and
    Persona.id_citta = Città.id_citta AND
    Città.nome = 'Roma';
    create view allievi_ore as SELECT AllievoRegistro.id_allievo,sum(AllievoRegistro.fine-AllievoRegistro.inizio) / 10000 as ore from AllievoRegistro GROUP by
AllievoRegistro.id_allievo having ore >= 5;
    SELECT Persona.codice_fiscale from Persona,allievi_ore WHERE
    Persona.id_persona = allievi_ore.id_allievo and
    allievi_ore.id_allievo in (SELECT * from allievi_roma);


create view docenti_roma as SELECT Docente.id_docente from Docente,Persona,Città where Persona.id_persona = Docente.id_docente AND
Persona.id_citta = Città.id_citta and Città.nome = 'Roma';
SELECT count(*) from Lezione WHERE
Lezione.id_docente in (SELECT * from docenti_roma);


    create view id_resp as SELECT Responsabile.id_responsabile from Responsabile;
    SELECT Persona.codice_fiscale, count(Progetto.id_responsabile) tot_progetti
    from Persona,Progetto WHERE
    Progetto.id_responsabile = Persona.id_persona and
    Persona.id_persona in (SELECT * from id_resp) GROUP BY Progetto.id_responsabile;


create view appo2 as select sum(AllievoRegistro.fine -AllievoRegistro.inizio) / 10000 as tot_ore, AllievoRegistro.id_allievo from AllievoRegistro GROUP by
AllievoRegistro.id_allievo;
select appo2.tot_ore,Persona.cognome,Persona.nome from appo2,Persona where Persona.id_persona = appo2.id_allievo;


drop view allievi_lezione;
    create view allievi_lezione as SELECT AllievoRegistro.id_allievo from AllievoRegistro,Lezione where Lezione.id_lezione = AllievoRegistro.id_lezione and Lezione.giorno
='2023-02-22' and Lezione.id_corso = 1  and AllievoRegistro.nota = 'Presente';
    select count(*) from allievi_lezione;
    SELECT Persona.codice_fiscale from Persona WHERE Persona.id_persona in (SELECT * from allievi_lezione);


create view lez_corso as SELECT sum(Lezione.fine - Lezione.inizio)/10000 as ore,Lezione.id_corso from Lezione GROUP by Lezione.id_corso;
select lez_corso.ore,Corso.nome from Corso,lez_corso where Corso.id_corso = lez_corso.id_corso;


    create view lez_prog as select sum(Lezione.fine -Lezione.inizio) / 10000 as ore, Progetto.id_progetto from
    Lezione,Progetto,Corso WHERE
    Corso.id_corso = Lezione.id_corso AND
    Corso.id_progetto = Progetto.id_progetto
    group by Progetto.id_progetto;
    select lez_prog.ore,Progetto.nome from lez_prog,Progetto where Progetto.id_progetto = lez_prog.id_progetto;


select sum(doc_reg.fine - doc_reg.inizio) / 10000 as ore from doc_reg,Lezione
where
Lezione.id_lezione = doc_reg.id_lezione AND
doc_reg.nota = 'Presente' and
Lezione.id_corso = 1;
```

Figura 18: Select SQL

```sql

 select distinct Lezione.*,Persona.nome,Persona.cognome,Aula.nome as aula from Lezione,Persona,Aula
    where Lezione.id_corso = 1 and
    Lezione.id_aula = Aula.id_aula and
    Lezione.id_docente = Persona.id_persona;

select distinct Lezione.*,Persona.nome,Persona.cognome,Aula.nome as aula,Corso.nome as corso from Lezione,Persona,Aula,Corso
    where Lezione.id_corso = Corso.id_corso and
    Lezione.id_aula = Aula.id_aula and
    Lezione.id_docente = Persona.id_persona and
    Corso.id_progetto = 1;

 SELECT distinct Persona.nome,Persona.cognome FROM `doc_corso`,Persona WHERE `doc_corso`.`id_docente` = Persona.id_persona and `doc_corso`.`id_corso` = 1;
 SELECT distinct Persona.nome,Persona.cognome FROM `doc_corso`,Persona,Corso WHERE `doc_corso`.`id_docente` = Persona.id_persona and Corso.id_corso = `doc_corso`.`id_corso` AND
Corso.id_progetto = 1;

 select Persona.nome,Persona.cognome from Persona,all_prog where all_prog.id_allievo = Persona.id_persona and all_prog.id_progetto = 1;

 select Persona.nome,Persona.cognome from Persona,all_prog,Corso where all_prog.id_allievo = Persona.id_persona and
all_prog.id_progetto = Corso.id_corso AND
Corso.id_corso = 1;

select Lezione.giorno,doc_reg.inizio,doc_reg.fine,Corso.nome from Lezione,doc_reg,Corso where
doc_reg.nota = 'Presente' AND
Lezione.id_corso = Corso.id_corso AND
Lezione.id_lezione = doc_reg.id_lezione AND
Lezione.id_docente = 1;

select Lezione.giorno,AllievoRegistro.inizio,AllievoRegistro.fine,Corso.nome as corso from Lezione,AllievoRegistro,Corso where
AllievoRegistro.nota = 'Presente' AND
Lezione.id_corso = Corso.id_corso AND
Lezione.id_lezione = AllievoRegistro.id_lezione AND
AllievoRegistro.id_allievo = 1;

 select Lezione.giorno,Lezione.inizio,Lezione.fine,Corso.nome from Lezione,Corso where Corso.id_corso = Lezione.id_corso AND
Lezione.id_aula = 1;

select Persona.nome,Persona.cognome from AllievoRegistro,Persona where AllievoRegistro.id_allievo = Persona.id_persona AND
AllievoRegistro.id_lezione = 1 order by Persona.cognome;
```

Figura 19: Select SQL

```sql
INSERT INTO all_prog(id_allievo,id_progetto) VALUES (1,1);

INSERT INTO `doc_corso` (`id_docente`, `id_corso`) VALUES (1,2);

INSERT INTO `Persona` (`nome`, `codice_fiscale`, `cognome`, `data_nascita`, `telefono`, `indirizzo`, `id_citta`) VALUES ('Utente3', 'suocodice', 'suocognome', '2023-02-21', NULL, 'indirizzosuo', '1');

INSERT INTO AllievoRegistro('inizio','fine','nota','id_allievo','id_lezione') VALUES (10:00,12:00,'Presente',1,1);

INSERT INTO doc_reg('inizio','fine','nota','id_docente','id_lezione') VALUES (10:00,12:00,'Presente',1,1);
```

Figura 20: Insert SQL

```sql
select distinct Aula.nome from Aula where Aula.id_aula in
(select distinct Lezione.id_aula from Lezione where Lezione.id_docente in
            (select link.id_docente from link GROUP by link.id_docente HAVING count(link.id_link >=2))
);
```

Figura 21: Select SQL con 3 livelli

7. Ritornare il numero di ore previste per ogni progetto
8. Ritornare il numero di ore svote da un corso
9. Restituire il calendario di un corso
10. Restituire il calendario di un progetto
11. Restituire tutti i docenti di un corso
12. Restituire tutti i docenti di un progetto
13. Restituire tutti gli allievi di un progetto
14. Restituire tutti gli allievi di un corso
15. Restituire tutte le presenze di un docente
16. Restituire tutte le presenze di un allievo
17. Restituire tutte le lezioni svolte in un aula
18. Restituire tutti gli allievi presenti ad una lezione
19. Restituire i nomi delle aule in cui sono state tenute lezioni da docenti che hanno almeno 2 link
20. Inserimento presenza di un allievo
21. Inserimento presenza di un docente
22. Registrare una persona
23. Assegnare un docente ad un corso
24. Assegnare un allievo ad un progetto

Tutte le query e gli insert sono illustrati nelle figure 18, 19 e 20

7.2.1 Soddisfacimento dei requisiti sul punto H

1. Devono richiedere l'utilizzo degli operatori aggregati
2. Devono richiedere l'utilizzo di query annidate (di cui una con almeno tre livelli(la 19 ne ha 3)) vedere a fig 21
3. La metà delle query deve anche essere scritta utilizzando le viste
4. Devono coinvolgere almeno due tabelle. Ovviamente il vostro progetto può anche utilizzare altre query oltre a quelle necessarie per soddisfare i requisiti del punto H

8 Osservazioni

Il type Indirizzo non è stato usato in quanto ho implementato il database su mysql che non ha l'istruzione per creare un dominio ennupla.
Ho comunque illustrato il comando per creare il dominio tramite sintassi postgreSQL.
Anche il type Nota è stato usato in linea con la sintassi di mySQL.

9 Applicazione

Nel seguente video illustro il funzionamento dell'applicazione andando a far vedere qualche query/insert che ho implementato,illustro anche alcuni casi in cui gestisco l'errore(es: provo ad aggiungere una persona con lo stesso codice fiscale di un'altra già registrata,etc..)