

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE
DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI - ANNO ACCADEMICO 2021/2022

Sistema di Gestione per Corsi di Formazione

Autori

Dario FRANZESE
N86003613
dari.franzese@studenti.unina.it

Renato ANTONELLI
N86003779
re.antonelli@studenti.unina.it

Docente

Prof. Adriano PERON

27 febbraio 2022

Indice

I	Introduzione	5
1	Approccio al problema	5
2	Guida alla lettura della documentazione	6
II	Progettazione Concettuale	7
3	Class Diagram	7
3.1	Analisi del Class Diagram	7
4	Class Diagram Ristrutturato	9
4.1	Analisi del Class Diagram Ristrutturato	9
5	Dizionari	10
5.1	Dizionario delle Classi	10
5.2	Dizionario delle Associazioni	12
5.3	Dizionario dei Vincoli	13
III	Progettazione Logica	15
6	Schema Logico	15
6.1	Tabelle	15
6.2	Riferimenti	16
IV	Progettazione Fisica	17
7	Definizione delle Tabelle	17
7.1	CORSI	17
7.2	OPERATORI	18
7.3	AREE_TEMATICHE	18
7.4	PAROLE_CHIAVE	19
7.5	LEZIONI	19
7.6	STUDENTI	20
7.7	ISCRIZIONI	20
7.8	PRESENZE	21

7.9	TEMI	22
7.10	CARATTERIZZA	23
7.11	DOMANDE_SICUREZZA	23
7.12	DOMANDE_OPERATORI	24
8	Implementazione dei vincoli	25
8.1	Vincoli Intrarelazionali	25
8.1.1	INSERIMENTO_IN_TERMINATO	25
8.1.2	DATA_LEZIONI	27
8.1.3	PRESENZA_CONTEMPORANEA	28
8.1.4	LUNGHEZZA_NOME_UTENTE_VIOLATA	30
8.1.5	LUNGHEZZA_PASSWORD_VIOLATA	31
8.1.6	PASSWORD_CARATTERI_SPECIALI	32
8.1.7	NOME_UTENTE_CARATTERI_SPECIALI	33
8.1.8	CREAZIONE_LEZIONI_CONTEMPORANE	34
8.1.9	DATA_LEZIONE_CORSO	36
8.2	Vincoli Interrelazionali	37
8.2.1	NO_LEZIONI_CORSO_TERMINATO	37
8.2.2	MAX PARTECIPANTI	39
8.2.3	ISCRIZIONE_IN_TERMINATO	40
8.2.4	LEZIONE_ISCRIZIONE	41
8.2.5	ELIMINAZIONE_PRESENZA	42
8.2.6	LEZIONE_CORSO_TERMINATO	43
9	Definizione delle Viste	44
9.1	PARTECIPANTI	44
9.2	PRESENZE_TOALI	44
9.3	PRESENZE_STUDENTE_CORSO	44
10	Definizione delle Sequenze	45
10.1	Sequenza ID_CORSO	45
10.2	Sequenza ID_OPERATORE	45
10.3	Sequenza ID_LEZIONE	45
10.4	Sequenza MATRICOLA	45
10.5	Sequenza ID_DOMANDA	46
v	Popolamento del DataBase	47

11 Tabelle	47
11.1 AREE_TEMATICHE	47
11.2 PAROLE_CHIAVE	48
11.3 OPERATORI	49
11.4 CORSI	50
11.5 STUDENTI	52
11.6 ISCRIZIONI	53
11.7 PRESENZE	55
11.8 TEMI	59
11.9 CARATTERIZZA	61
11.10DOMANDE_SICUREZZA	62
11.11DOMANDE_OPERATORI	63
 VI Guida all'utilizzo	 64
12 Registrazione	65
13 Creazione	66
13.1 Creazione Corso	66
13.2 Creazione Lezione	67
13.3 Creazione Studente	68
14 Modifica	69
14.1 Modifica Corso	69
14.2 Modifica Lezione	70
14.3 Modifica Studente	71
15 Statistiche	72
15.1 Statistiche Corso	72
15.2 Statistiche Lezione	73
15.3 Statistiche Studente	74

Parte I

Introduzione

1 Approccio al problema

La traccia prevedeva la creazione di un database relazionale che permettesse la gestione di un sistema di corsi di formazione.

Il database utilizzato per lo sviluppo é POSTGRES, per tanto il linguaggio utilizzato é POSTGRESQL.

L'applicazione permette la gestione di 3 entità principali: Corsi, Studenti e Lezioni.

A un corso possono essere iscritti più studenti, e può essere formato da più lezioni.

É stato scelto un approccio "real-time" del problema piuttosto che uno di tipo archivistico, questo implica l'impedimento di alcune funzioni attraverso l'implementazione di triggers.

Ad esempio non é possibile la creazione di corsi nel passato, di aggiungere studenti a lezioni concluse etc.

La base di dati tiene traccia delle presenze e delle iscrizioni dello studente.

Ogni corso:

- Richiede un numero di presenze minimo per accedere all'esame
- Possiede un numero massimo di partecipanti ammessi
- É caratterizzato da un elenco di parole chiave
- Può appartenere a più aree tematiche
- Può essere terminato o meno (indipendentemente dal numero di lezioni)

Più corsi possono appartenere, a una stessa area tematica .

É possibile effettuare operazioni di filtraggio sia sui corsi che sulle lezioni.

É possibile visualizzare statistiche sulle tre entità principali frutto di query avanzate come tassi di presenza, corsi ai quali uno studente é ammesso e numero di partecipanti a un corso

2 Guida alla lettura della documentazione

Per facilitare la lettura della documentazione si lasciano di seguito alcune indicazioni alla lettura:

- Nella "Guida all'uso" sono inserite solo alcune delle funzionalità dell'applicazione.
Per un'esperienza più completa si consiglia di provare l'applicativo
- Per favorire una lettura migliore i class diagram sono allegati in copia nella cartella.
- Al puro fine didattico nel popolamento del database sono inseriti valori tali da violare intenzionalmente alcuni trigger (come l'iscrizione di lezioni passate).
Per effettuare una prova si consiglia di effettuare il popolamento prima di inserire i trigger

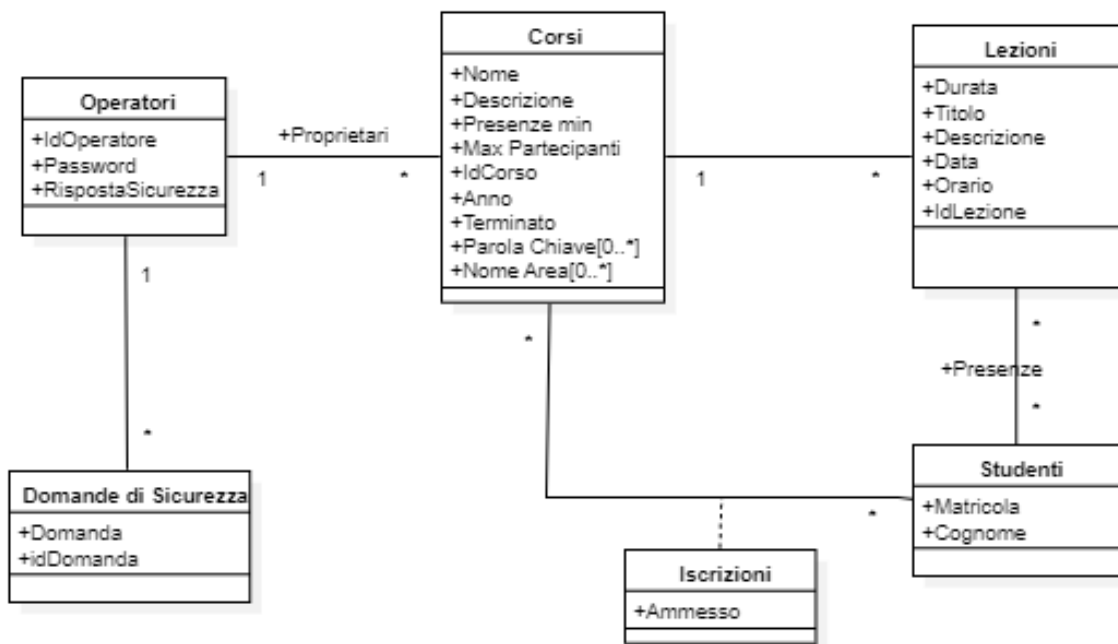
Parte II

Progettazione Concettuale

La progettazione concettuale consiste nel disegnare la base di dati ed adattarla allo schema relazionale nel caso di un sistema di questo tipo.

Questa progettazione é rappresentata dal Class Diagram, in particolare é stato scelto il modello UML per rappresentare taale schema.

3 Class Diagram



3.1 Analisi del Class Diagram

Il Class Diagram nella sua versione **non** ristrutturata presenta una serie di proprietà che non sono ammesse in una base di dati relazionale, in particolare non sono ammessi

- Attributi Multipli
- Attributi Strutturati

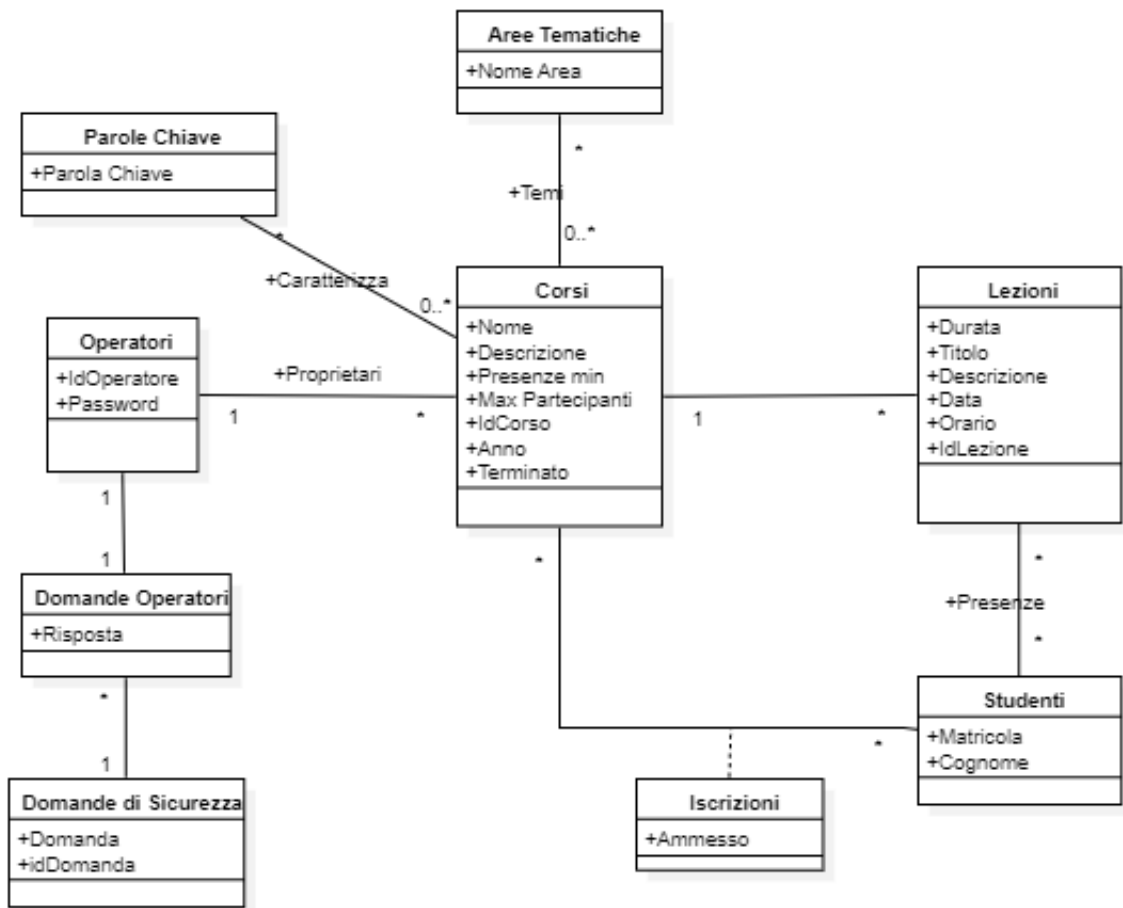
- Gerarchie

Inoltre é possibile, durante il processo di Ristrutturazione della base di dati, effettuare ottimizzazioni sulle ridondanze e sul partizionamento di entità e associazioni.

Nel nostro caso le problematiche sono:

1. L'entità Corsi presenta l'attributo multiplo "Parola Chiave"
2. L'entità Corsi presenta l'attributo multiplo "Aree Tematiche"
3. L'entità "Operatori" possiede un attributo "Risposta di sicurezza" che può essere isolato dalla classe in quanto dipende dall'entità "Domande di sicurezza"

4 Class Diagram Ristrutturato



4.1 Analisi del Class Diagram Ristrutturato

Il Class Diagram Ristrutturato é ottimizzato ed adattato alla base di dati relazionale. In particolare le modifiche si riflettono in:

1. L'attributo multiplo "Aree Tematiche" é ora un'associazione (0...*-*) con l'entitá corsi item L'attributo multiplo "Parole Chiave" é ora un'associazione (0...*) con l'entitá corsi
2. L'attributo Risposte di sicurezza é ora rappresentato all'interno dell'entitá "Domande Operatori"

3. L'entità "Domande Operatori" é in associazione (1-*) con l'entità "Domande di Sicurezza" e (1-1) con l'entità operatori (in modo tale da mantenere la dipendenza che prima era rappresentata dall'attributo interno all'entità)

5 Dizionari

I dizionari rappresentano una trasposizione scritta del class diagram, accompagnata da una breve descrizione esplicativa. Inoltre vengono esplicitati i vincoli che regolano l'interazione tra le entità e le associazioni.

I dizionari si suddividono in

- Dizionario delle Classi
- Dizionario delle Associazioni
- Dizionario dei Vincoli

5.1 Dizionario delle Classi

Seguono le Classi presenti nella Base di Dati

<i>Classe</i>	<i>Descrizione</i>	<i>Attributi</i>
<i>Operatori</i>	Contiene le informazioni relative all'operatore.	<i>id_operatore</i> (String): Primary Key dell'operatore. <i>nome_utente</i> (String): Nome dell'operatore. <i>password</i> (String): Password di accesso.
<i>Aree_Tematiche</i>	Contiene le varie aree tematiche disponibili.	<i>nome_area</i> (String): Primary Key e nome dell'area tematica.
<i>Parole_Chiave</i>	Contiene le varie parole chiave disponibili.	<i>parola_chiave</i> (String): Primary Key e nome della parola.

Corso	Contiene le informazioni relative al Corso.	<i>id_corso</i> (String): Primary Key del corso. <i>nome</i> (String): Nome del corso. <i>descrizione</i> (String): breve descrizione del corso. <i>presenze_min</i> (Integer): numero minimo di presenze per poter essere ammesso all'esame del corso. <i>max_partecipanti</i> (Integer): numero massimo di iscritti al corso. <i>anno</i> (String): Anno del corso. <i>terminato</i> (Boolean): informazione se il corso è terminato.
Lezioni	Contiene le informazioni relative ad una lezione.	<i>id_lezione</i> (String): Primary Key della lezione. <i>titolo</i> (String): Titolo della lezione. <i>descrizione</i> (String): breve descrizione sulla lezione. <i>durata</i> (Interval): durata in ore della lezione. <i>data</i> (Date): Data della lezione. <i>orario</i> (Time): orario inizio della lezione.
Studente	Contiene le informazioni relative allo studente.	<i>matricola</i> (String): Primary Key dello studente e numero di matricola. <i>nome</i> (String): Nome dello studente. <i>cognome</i> (String): Cognome dello studente.
Domande_Sicurezza	Contiene le domande di sicurezza disponibili per un eventuale recupero dati	<i>id_domanda</i> (String): Primary Key della domanda. <i>domanda</i> (String): Domanda di sicurezza.
Domande_Operator	Contiene la risposta dell'operatore alla domanda di sicurezza.	<i>id_domanda</i> (String): Riferimento alla Primary Key delle domande_sicurezza. <i>id_operatore</i> (String): Riferimento alla Primary Key dell'operatore. <i>risposta</i> (String): Risposta dell'operatore.

5.2 Dizionario delle Associazioni

Seguono le Associazioni presenti nella Base di Dati

Nome	Descrizione	Classi Coinvolte	Attributi
Iscrizioni	Indica l'iscrizione di uno studente ad un determinato corso.	<i>Corso</i> [*] ruolo <i>ha iscritti</i> : indica lo studente iscritto a quel corso. <i>Studenti</i> [*] ruolo <i>Iscritti</i> : indica tutti gli studenti iscritti al corso.	<i>ammesso</i> (Boolean): Informazione se lo studente è ammesso all'esame.
Presenze	Indica la presenza di uno studente ad una determinata lezione.	<i>Lezioni</i> [*] ruolo <i>ha prenotazioni</i> : indica la lezione alla quale gli studenti sono prenotati. <i>Studenti</i> [*] ruolo <i>Prenota</i> : indica lo studente prenotato alla lezione.	
Temi	Indica le aree tematiche di un determinato corso.	<i>Corso</i> [*] ruolo <i>si occupa di</i> : indica quali corsi si occupano di questo tema. <i>Aree_Tematiche</i> [0..*] ruolo <i>trattate da</i> : indica tutti i temi quali il corso si occupa.	
Caratterizza	Indica le parole chiavi associate ad un determinato corso.	<i>Corso</i> [*] ruolo <i>caratterizzato da</i> : indica quali corsi sono caratterizzati da queste parole. <i>Parole_chiave</i> [0..*] ruolo <i>caratterizza</i> : indica il corso da quali parole chiave è caratterizzato.	
Corsi_Lezioni	Indica le lezioni di un determinato corso	<i>Corso</i> [1] ruolo <i>è formato da</i> : indica a quale corso fanno parte. <i>Lezioni</i> [*] ruolo <i>fa parte di</i> : indica le lezioni che fanno parte del corso.	
Proprietari	Indica i corsi presidiati dall'operatore.	<i>Corso</i> [*] ruolo <i>presiedato da</i> : indica chi si occupa del corso. <i>Operatori</i> [1] ruolo <i>presiede</i> : indica chi si occupa del corso.	
Domande_Operatori_Sicurezza	Indica la domanda di sicurezza dell'operatore	<i>Domande_Operatori</i> [1] ruolo <i>risponde</i> : indica la risposta dell'operatore. <i>Domande_Sicurezza</i> [*] ruolo <i>domanda</i> : indica la domanda di sicurezza dell'operatore.	
Operatori_Risposta_Sicurezza	Indica la risposta alla domanda di sicurezza dell'operatore	<i>Domande_Operatori</i> [1] ruolo <i>recupera dati</i> : indica la risposta dell'operatore. <i>Operatori</i> [1] ruolo <i>chiede di recuperare dati</i> : indica la domanda di sicurezza dell'operatore.	

5.3 Dizionario dei Vincoli

Seguono i vincoli implementati nella Base di Dati

Nome Vincolo	Tipo	Descrizione
<i>Inserimento_in_terminato</i>	Intrarelazionali.	<i>Non è possibile creare una nuova lezione se il corso è terminato.</i>
<i>Ammissione</i>	Interrelazionali	<i>Update del valore ammesso se lo studente ha superato il numero minimo di presenze per poter accedere all'esame.</i>
<i>Max_partecipanti</i>	Interrelazionali	<i>Non è possibile iscrivere uno studente ad un corso se è stato già raggiunto il numero massimo di partecipanti.</i>
<i>Data_lezioni</i>	Intrarelazionali.	<i>Non è possibile iscriversi a lezioni già concluse.</i>
<i>Presenza_contemporanea</i>	Intrarelazionali.	<i>Non è possibile iscriversi ad una lezione se si è già prenotati per una lezione che si terrà allo stesso orario.</i>
<i>Iscrizione_in_terminato</i>	Interrelazionali	<i>Non è possibile iscriversi ad un corso terminato.</i>
<i>Lezione_iscrizione</i>	Interrelazionali	<i>Non è possibile iscriversi ad una lezione se non si è iscritti al corso della lezione stessa.</i>
<i>Lunghezza_password_violata</i>	Intrarelazionali.	<i>La password deve contenere minimo 6 caratteri.</i>
<i>Lunghezza_nome_utente_violata</i>	Intrarelazionali.	<i>Il nome utente deve contenere minimo 4 caratteri.</i>
<i>Nome_utente_caratteri_speciali</i>	Intrarelazionali.	<i>Il nome utente non deve contenere caratteri speciali quali (@,!,#).</i>

<i>Password_caratteri_speciali</i>	Intrarelazionali.	<i>La password non deve contenere caratteri speciali quali(@,!,#).</i>
<i>Eliminazione_presenza</i>	Interrelazionali	<i>Non è possibile annullare l'iscrizione ad una lezione se quest'ultima è terminata.</i>
<i>Creazione_lezione_contemporanea</i>	Intrarelazionali.	<i>Non è possibile creare due lezioni nello stesso orario.</i>
<i>Lezione_corso_terminato</i>	Interrelazionali	<i>Non è possibile iscriversi ad una lezione se il corso è terminato.</i>
<i>Data_lezione_corso</i>	Intrarelazionali.	<i>Non è possibile creare una lezione de la data di essa non corrisponde con l'anno del corso.</i>

Parte III

Progettazione Logica

La progettazione logica della base di dati consiste nella stesura degli schemi logici delle tabelle della base di dati, degli attributi di ogni tabella con relativi riferimenti ad altre tabelle (Foreign Keys)

6 Schema Logico

Si indica con la doppia sottolineatura una **foreign key** mentre con la singola sottolineatura una **primary key**.

Seguono gli schemi logici delle tabelle e i riferimenti delle foreign keys

6.1 Tabelle

Indichiamo per ogni entità i suoi attributi indicando le Primary Keys ed eventuali Foreign Keys

- AREE_TEMATICHE(nome_area)
- PAROLE_CHIAVE(parola_chiave)
- OPERATORI(id_operatore, nome_utente, password)
- CORSI(id_corso, nome, descrizione, presenze_min, max_partecipanti, anno, terminato, id_operatore)
- LEZIONI(id_lezione, titolo, descrizione, durata , data, orario, id_corso)
- STUDENTI(matricola, nome, cognome)
- ISCRIZIONI(matricola, id_corso, ammesso)
- PRESENZE(matricola, id_lezione)
- TEMI(nome_area, id_corso)
- CARATTERIZZA(parola_chiave, id_corso)
- DOMANDE_SICUREZZA(id_domanda, domanda)
- DOMANDE_OPERATORI(id_operatore, id_domanda, risposta)

6.2 Riferimenti

Per ogni **Foreign Key** presente indichiamo la sua **Primary Key** di riferimento

- CORSO
 - $\text{id_operatore} \mapsto \text{operatori.id_operatore}$
- LEZIONI
 - $\text{id_corso} \mapsto \text{corsi.id_corso}$
- ISCRIZIONI
 - $\text{matricola} \mapsto \text{studenti.matricola}$
 - $\text{id_corso} \mapsto \text{corsi.id_corso}$
- PRESENZE
 - $\text{matricola} \mapsto \text{studenti.matricola}$
 - $\text{id_lezione} \mapsto \text{lezioni.id_lezione}$
- TEMI
 - $\text{nome_area} \mapsto \text{aree_tematiche.nome_area}$
 - $\text{id_corso} \mapsto \text{corsi.id_corso}$
- CARATTERIZZA
 - $\text{parola_chiave} \mapsto \text{parole_chiave.parola_chiave}$
 - $\text{id_corso} \mapsto \text{corsi.id_corso}$
- DOMANDE_OPERATORI
 - $\text{id_operatore} \mapsto \text{operatori.id_operatore}$
 - $\text{id_domanda} \mapsto \text{domande_sicurezza.id_domanda}$

Parte IV

Progettazione Fisica

La fase di Progettazione Fisica rappresenta l'implementazione della Progettazione Concettuale (o logica) arricchendo con gli appositi strumenti per garantire la robustezza della base di dati. Questi sono Vincoli, Viste e Sequenze.

7 Definizione delle Tabelle

Le tabelle rappresentano le entità del class diagram all'interno della base di dati, con tutti i loro attributi

7.1 CORSI

```
1 CREATE TABLE IF NOT EXISTS public.corsi
2 (
3     id_corso character varying NOT NULL,
4     id_operatore character varying NOT NULL,
5     nome character varying NOT NULL,
6     descrizione character varying NOT NULL,
7     presenze_min integer NOT NULL,
8     max_partecipanti integer NOT NULL,
9     anno character varying NOT NULL,
10    terminato boolean NOT NULL DEFAULT false,
11
12    CONSTRAINT nome_corsi_unique UNIQUE (nome),
13    CONSTRAINT "Corsi_pkey" PRIMARY KEY (id_corso),
14    CONSTRAINT caratteri_anno CHECK (anno SIMILAR TO '[1-2][0-9][0-9][0-9]'),
15
16    CONSTRAINT corsi_fk1 FOREIGN KEY (id_operatore)
17        REFERENCES public.operatori (id_operatore) MATCH SIMPLE
18        ON UPDATE CASCADE
19        ON DELETE CASCADE
20 )
21 )
```

7.2 OPERATORI

```
1 CREATE TABLE IF NOT EXISTS public.operatori
2 (
3     id_operatore character varying COLLATE NOT NULL,
4     nome_utente character varying NOT NULL,
5     password character varying NOT NULL,
6
7     CONSTRAINT "Operatori_pkey" PRIMARY KEY (id_operatore),
8     CONSTRAINT nome_utene_unique UNIQUE (nome_utente)
9 )
```

7.3 AREE_TEMATICHE

```
1 CREATE TABLE IF NOT EXISTS public.aree_tematiche
2 (
3     nome_area character varying NOT NULL,
4
5     CONSTRAINT "AreeTematiche_pkey" PRIMARY KEY (nome_area)
6 )
```

7.4 PAROLE_CHIAVE

```
1 CREATE TABLE IF NOT EXISTS public.parole_chiave
2 (
3     parola_chiave character varying NOT NULL,
4
5     CONSTRAINT parole_chiave_pkey PRIMARY KEY (parola_chiave)
6 )
```

7.5 LEZIONI

```
1 CREATE TABLE IF NOT EXISTS public.lezioni
2 (
3     id_lezione character varying NOT NULL,
4     titolo character varying NOT NULL,
5     descrizione character varying NOT NULL,
6     durata interval NOT NULL,
7     data date NOT NULL,
8     orario time NOT NULL,
9     id_corso character varying NOT NULL,
10
11     CONSTRAINT "Lezioni_pkey" PRIMARY KEY (id_lezione),
12     CONSTRAINT titolo_corso UNIQUE (id_corso, titolo),
13
14     CONSTRAINT lezioni_fk1 FOREIGN KEY (id_corso)
15         REFERENCES public.corsi (id_corso) MATCH SIMPLE
16         ON UPDATE CASCADE
17         ON DELETE CASCADE
18 )
```

7.6 STUDENTI

```
1 CREATE TABLE IF NOT EXISTS public.studenti
2 (
3     matricola character varying NOT NULL,
4     nome character varying NOT NULL,
5     cognome character varying NOT NULL,
6
7     CONSTRAINT "Studenti_pkey" PRIMARY KEY (matricola)
8 )
```

7.7 ISCRIZIONI

```
1 CREATE TABLE IF NOT EXISTS public.iscrizioni
2 (
3     matricola character varying NOT NULL,
4     id_corso character varying NOT NULL,
5     ammesso boolean NOT NULL DEFAULT false,
6
7     CONSTRAINT iscrizioni_unique UNIQUE (matricola, id_corso),
8     CONSTRAINT iscrizioni_fk1 FOREIGN KEY (id_corso)
9         REFERENCES public.corsi (id_corso) MATCH SIMPLE
10        ON UPDATE CASCADE
11        ON DELETE CASCADE,
12
13     CONSTRAINT iscrizioni_fk2 FOREIGN KEY (matricola)
14        REFERENCES public.studenti (matricola) MATCH SIMPLE
15        ON UPDATE CASCADE
16        ON DELETE CASCADE
17 )
```

7.8 PRESENZE

```
1 CREATE TABLE IF NOT EXISTS public.presenze
2 (
3     matricola character varying NOT NULL,
4     id_lezione character varying NOT NULL,
5
6     CONSTRAINT presenze_unique UNIQUE (matricola, id_lezione),
7     CONSTRAINT presenze_fk1 FOREIGN KEY (matricola)
8         REFERENCES public.studenti (matricola) MATCH SIMPLE
9         ON UPDATE CASCADE
10        ON DELETE CASCADE,
11
12     CONSTRAINT presenze_fk2 FOREIGN KEY (id_lezione)
13         REFERENCES public.lezioni (id_lezione) MATCH SIMPLE
14         ON UPDATE CASCADE
15        ON DELETE CASCADE
16 )
```

7.9 TEMI

```
1 CREATE TABLE IF NOT EXISTS public.temi
2 (
3     nome_area character varying NOT NULL,
4     id_corso character varying NOT NULL,
5
6     CONSTRAINT temi_unique UNIQUE (nome_area, id_corso),
7     CONSTRAINT temi_fk1 FOREIGN KEY (nome_area)
8         REFERENCES public aree_tematiche (nome_area) MATCH SIMPLE
9         ON UPDATE CASCADE
10        ON DELETE CASCADE,
11
12     CONSTRAINT temi_fk2 FOREIGN KEY (id_corso)
13         REFERENCES public corsi (id_corso) MATCH SIMPLE
14         ON UPDATE CASCADE
15        ON DELETE CASCADE
16 )
```

7.10 CARATTERIZZA

```
1 CREATE TABLE IF NOT EXISTS public.caratterizza
2 (
3     parola_chiave character varying NOT NULL,
4     id_corso character varying NOT NULL,
5
6     CONSTRAINT caratterizza_unique UNIQUE (parola_chiave, id_corso),
7
8     CONSTRAINT caratterizza_fk1 FOREIGN KEY (parola_chiave)
9         REFERENCES public.parole_chiave (parola_chiave) MATCH SIMPLE
10        ON UPDATE CASCADE
11        ON DELETE CASCADE,
12
13     CONSTRAINT caratterizza_fk2 FOREIGN KEY (id_corso)
14         REFERENCES public.corsi (id_corso) MATCH SIMPLE
15        ON UPDATE CASCADE
16        ON DELETE CASCADE
17 )
```

7.11 DOMANDE_SICUREZZA

```
1 CREATE TABLE IF NOT EXISTS public.domande_sicurezza
2 (
3     domanda character varying NOT NULL,
4     id_domanda character varying NOT NULL,
5
6     CONSTRAINT domande_sicurezza_pkey PRIMARY KEY (id_domanda)
7
8 )
```

7.12 DOMANDE_OPERATORI

```
1 CREATE TABLE IF NOT EXISTS public.domande_operatori
2 (
3     risposta character varying NOT NULL,
4     id_domanda character varying NOT NULL,
5     id_operatore character varying NOT NULL,
6
7     CONSTRAINT domande_operatori_pkey PRIMARY KEY (id_domanda, id_operatore, risposta),
8     CONSTRAINT domande_operatori_fk1 FOREIGN KEY (id_operatore)
9     REFERENCES public.operatori (id_operatore) MATCH SIMPLE
10    ON UPDATE CASCADE
11    ON DELETE CASCADE,
12
13    CONSTRAINT domande_operatori_fk2 FOREIGN KEY (id_domanda)
14    REFERENCES public.domande_sicurezza (id_domanda) MATCH SIMPLE
15    ON UPDATE CASCADE
16    ON DELETE CASCADE
17 )
```

8 Implementazione dei vincoli

I vincoli sulle tabelle permettono il corretto funzionamento della Base di Dati.

Possono essere implementati attraverso il costrutto **CONSTRAINT** oppure attraverso l'attivazione di **Trigger**.

Si suddividono in Intrarelazionali e Interrelazionali a seconda dell'impiego di una o più tabelle.

8.1 Vincoli Intrarelazionali

Si noti che i "CONSTRAINT" sono presenti direttamente nella definizione delle tabelle

8.1.1 INSERIMENTO_IN_TERMINATO

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una lezione,
3  controlla che la lezione non sia stata inserita in un corso "terminato".
4  Qualora dovesse verificarsi questa illegalità, il trigger elimina la
5  lezione appena inserita. */
6
7  CREATE FUNCTION check_inserimento_in_terminato()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE violazione INTEGER;
11     BEGIN
12         SELECT COUNT (*) INTO violazione
13         FROM corsi s
14         WHERE s.id_corso = NEW.id_corso AND s.terminato = true;
15
16         IF (violazione>0) THEN
17             RAISE SQLSTATE '10005';
18             ROLLBACK;
19             RAISE EXCEPTION 'ATTENZIONE : il corso e termianto';
20         END IF;
21         RETURN NEW;
22     END
23     $$ LANGUAGE plpgsql;
24
25     CREATE TRIGGER inserimento_in_terminato
```

```
26      AFTER INSERT ON lezioni
27      FOR EACH ROW
28      EXECUTE PROCEDURE check_inserimento_in_terminato();
```

8.1.2 DATA_LEZIONI

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una Presenza dello studente,
3  controlla se la lezione non sia già terminata.
4  Qualora dovesse verificarsi questa illegalità, il trigger elimina la
5  presenza appena inserita. */
6
7  CREATE OR REPLACE FUNCTION check_data_lezione()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE data_lezione DATE;
11     BEGIN
12         SELECT data INTO data_lezione
13         FROM lezioni l
14         WHERE l.id_lezione = NEW.id_lezione;
15
16         IF (data_lezione < CURRENT_DATE) THEN
17             RAISE SQLSTATE '10007';
18             ROLLBACK;
19             RAISE EXCEPTION 'La lezione si è già conclusa';
20         END IF;
21         RETURN NEW;
22     END
23     $$ LANGUAGE plpgsql;
24
25     CREATE OR REPLACE TRIGGER data_lezioni
26     AFTER INSERT ON presenze
27     FOR EACH ROW
28     EXECUTE PROCEDURE check_data_lezione();
```

8.1.3 PRESENZA_CONTEMPORANEA

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una Presenza dello studente,
3  controlla se si è già iscritti ad una lezione che si svolgerà nello stesso orario.
4  Qualora dovesse verificarsi questa illegalità, il trigger elimina la
5  presenza appena inserita. */
6
7  CREATE FUNCTION check_presenza_contemporanea()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE violazione INTEGER;
11             startt TIME WITHOUT TIME ZONE;
12             endt TIME WITHOUT TIME ZONE;
13     datat DATE;
14     BEGIN
15
16         SELECT l.orario INTO startt
17         FROM lezioni l
18         WHERE new.id_lezione = l.id_lezione;
19
20         SELECT l.orario+l.durata INTO endt
21         FROM lezioni l
22         WHERE new.id_lezione = l.id_lezione;
23
24         SELECT l.data into datat
25         FROM lezioni l
26         WHERE new.id_lezione = l.id_lezione;
27
28         SELECT COUNT (*) INTO violazione
29         FROM presenze pr JOIN lezioni le ON pr.id_lezione = le.id_lezione
30         WHERE pr.matricola = NEW.matricola AND le.data = datat
31         AND ((le.orario BETWEEN startt AND endt)
32         OR (le.orario+le.durata BETWEEN startt AND endt)
33         OR (le.orario < startt AND le.orario+le.durata > endt));
34
35     IF (violazione > 1) THEN
36         RAISE SQLSTATE '10008';
37         ROLLBACK;
```

```
38             RAISE EXCEPTION 'Ti sei gia iscritto ad
39             una lezione in contemporanea';
40         END IF;
41         RETURN NEW;
42     END
43
44 $$ LANGUAGE plpgsql;
45
46 CREATE TRIGGER presenza_contemporanea
47 AFTER INSERT ON presenze
48 FOR EACH ROW
49 EXECUTE PROCEDURE check_presenza_contemporanea();
```

8.1.4 LUNGHEZZA_NOME_UTENTE_VIOLATA

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una una nuova password,
3  controlla se la stringa viola il numero di caratteri minimo.
4  Qualora dovesse verificarsi questa illegalità, il trigger elimina la
5  password appena inserita. */
6
7  CREATE FUNCTION check_lunghezza_nome_utente()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE index INTEGER;
11             stringa VARCHAR(100);
12
13     BEGIN
14         SELECT o.nome_utente INTO stringa
15         FROM operatori o
16         WHERE o.nome_utente = NEW.nome_utente;
17
18         stringa = SUBSTRING(stringa, 4, 100);
19         IF(stringa = '')THEN
20             RAISE SQLSTATE '10002';
21             ROLLBACK;
22             RAISE EXCEPTION 'Il nome utente deve
23             contenere almeno 4 caratteri';
24         END IF;
25         RETURN NEW;
26     END
27     $$ LANGUAGE plpgsql;
28
29     CREATE TRIGGER lunghezza_nome_utente_violata
30     AFTER INSERT ON operatori
31     FOR EACH ROW#
32     EXECUTE PROCEDURE check_lunghezza_nome_utente();
```

8.1.5 LUNGHEZZA_PASSWORD_VIOLATA

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una una nuova password,
3  controlla se la stringa viola il numero di caratteri minimo.
4  Qualora dovesse verificarsi questa illegalità, il trigger elimina la
5  password appena inserita. */
6
7  CREATE FUNCTION check_lunghezza_password()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE index INTEGER;
11             stringa VARCHAR(100);
12
13     BEGIN
14         SELECT o.password INTO stringa
15         FROM operatori o
16         WHERE o.nome_utente = NEW.nome_utente;
17
18         stringa = SUBSTRING(stringa, 6, 100);
19         IF(stringa = '')THEN
20             RAISE SQLSTATE '10001';
21             ROLLBACK;
22             RAISE EXCEPTION 'La password deve contenere almeno 6 caratteri';
23         END IF;
24         RETURN NEW;
25     END
26     $$ LANGUAGE plpgsql;
27
28     CREATE TRIGGER lunghezza_password_violata
29     AFTER INSERT ON operatori
30     FOR EACH ROW
31     EXECUTE PROCEDURE check_lunghezza_password();
```

8.1.6 PASSWORD_CARATTERI_SPECIALI

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una nuova password,
3  controlla se la stringa contiene dei caratteri speciali come: "@, !, ecc".
4  Qualora dovesse verificarsi questa illegalità, il trigger elimina la
5  password appena inserita. */
6
7  CREATE FUNCTION check_caratteri_speciali_password()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE violazione INTEGER;
11     BEGIN
12         SELECT COUNT(*) INTO violazione
13         FROM operatori o
14         WHERE o.password like '%!%' OR
15                o.password like '%@%' OR
16                o.password like '%#%' OR
17                o.password like '%$%';
18
19         IF(violazione>0)THEN
20             RAISE SQLSTATE '10004';
21             ROLLBACK;
22             RAISE EXCEPTION 'La password non puo
23                 contenere caratteri speciali';
24         END IF;
25         RETURN NEW;
26     END
27     $$ LANGUAGE plpgsql;
28
29     CREATE TRIGGER password_caratteri_speciali
30     AFTER INSERT ON operatori
31     FOR EACH ROW
32     EXECUTE PROCEDURE check_caratteri_speciali_password();
```

8.1.7 NOME_UTENTE_CARATTERI_SPECIALI

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di un nuovo nome utente,
3  controlla se la stringa contiene dei caratteri speciali come: "@, !, ecc".
4  Qualora dovesse verificarsi questa illegalità,
5  il trigger elimina il nome utente appena inserito. */
6
7  CREATE FUNCTION check_caratteri_speciali_nome_utente()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE violazione INTEGER;
11     BEGIN
12         SELECT COUNT(*) INTO violazione
13         FROM operatori o
14         WHERE o.nome_utente = NEW.nome_utente AND
15             ( o.nome_utente like '%!%' OR
16              o.nome_utente like '%@%' OR o.nome_utente like '%#%' OR
17              o.nome_utente like '%$%');
18
19         IF(violazione>0)THEN
20             RAISE SQLSTATE '10003';
21             ROLLBACK;
22             RAISE EXCEPTION 'Il nome utente
23             non puo contenere caratteri speciali';
24         END IF;
25         RETURN NEW;
26     END
27     $$ LANGUAGE plpgsql;
28
29     CREATE TRIGGER nome_utente_caratteri_speciali
30     AFTER INSERT ON operatori
31     FOR EACH ROW
32     EXECUTE PROCEDURE check_caratteri_speciali_nome_utente();
```

8.1.8 CREAZIONE_LEZIONI_CONTEMPORANE

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una nuova lezione,
3  controlla se ci sono già lezioni allo stesso orario,
4  Qualora dovesse verificarsi questa illegalità,
5  il trigger elimina la lezione appena inserita. */
6
7  CREATE FUNCTION check_creazione_lezione_contemporanea()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE violazione INTEGER;
11             startt TIME WITHOUT TIME ZONE;
12             endt TIME WITHOUT TIME ZONE;
13     datat DATE;
14     BEGIN
15
16         SELECT l.orario INTO startt
17         FROM lezioni l
18         WHERE new.id_lezione = l.id_lezione;
19
20         SELECT l.orario+l.durata INTO endt
21         FROM lezioni l
22         WHERE new.id_lezione = l.id_lezione;
23
24         SELECT l.data into datat
25         FROM lezioni l
26         WHERE new.id_lezione = l.id_lezione;
27
28         SELECT COUNT (*) INTO violazione
29         FROM lezioni le
30         WHERE le.id_corso = NEW.id_corso AND le.data = datat
31         AND ((le.orario BETWEEN startt AND endt) OR
32         (le.orario+le.durata BETWEEN startt AND endt)
33         OR (le.orario < startt AND le.orario+le.durata > endt));
34
35     IF (violazione > 1) THEN
36         RAISE SQLSTATE '10012';
37         ROLLBACK;
```

```
38             RAISE EXCEPTION 'Hai gia lezioni allo stesso orario
39             dello stesso giorno';
40         END IF;
41         RETURN NEW;
42     END
43
44 $$ LANGUAGE plpgsql;
45
46 CREATE TRIGGER creazione_lezione_contemporanea
47 AFTER INSERT ON lezioni
48 FOR EACH ROW
49 EXECUTE PROCEDURE check_creazione_lezione_contemporanea();
```

8.1.9 DATA_LEZIONE_CORSO

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una nuova lezione,
3  controlla se l'anno della lezione corrisponde all'anno del corso.
4  Qualora dovesse verificarsi questa illegalità,
5  il trigger elimina la lezione appena inserita. */
6
7  CREATE OR REPLACE FUNCTION check_data_lezione_corso()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE anno VARCHAR(100);
11     BEGIN
12
13         select c.anno into anno
14         from corsi c
15         where c.id_corso = new.id_corso;
16
17         IF (CAST(EXTRACT(YEAR FROM new.data) AS VARCHAR(100)) <> anno) THEN
18             RAISE SQLSTATE '10013';
19             ROLLBACK;
20             RAISE EXCEPTION 'la data della lezione
21 non corrisponde con l anno del corso';
22         END IF;
23
24         RETURN NEW;
25     END
26     $$ LANGUAGE plpgsql;
27
28     CREATE OR REPLACE TRIGGER data_lezione_corso
29     AFTER INSERT ON lezioni
30     FOR EACH ROW
31     EXECUTE PROCEDURE check_data_lezione_corso();
```

8.2 Vincoli Interrelazionali

8.2.1 NO_LEZIONI_CORSO_TERMINATO

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una Presenza dello studente,
3  controlla se il numero minimo di presenze
4  per poter esser ammesso all'esame sia superata o eguagliata.
5  Qualora dovesse superare il minimo,
6  andrà a settare il valore ammesso di "iscrizione" = true.  */
7
8  CREATE OR REPLACE FUNCTION check_ammissione()
9      RETURNS TRIGGER AS
10     $$
11     DECLARE p_min INTEGER;
12             p_studente INTEGER;
13             codice_corso VARCHAR(100);
14
15     BEGIN
16         SELECT l.id_corso INTO codice_corso
17         FROM presenze p JOIN lezioni l ON p.id_lezione = l.id_lezione
18         WHERE p.id_lezione = NEW.id_lezione;
19
20         SELECT COUNT(*) INTO p_studente
21         FROM presenze p JOIN lezioni l ON p.id_lezione = l.id_lezione
22         WHERE p.matricola = NEW.matricola
23         AND l.id_corso = codice_corso;
24
25         SELECT presenze_min INTO p_min
26         FROM corsi c
27         WHERE c.id_corso = codice_corso;
28
29         IF( p_studente >= p_min ) THEN
30             UPDATE iscrizioni i SET ammesso = TRUE
31             WHERE i.matricola = NEW.matricola
32             AND i.id_corso = codice_corso;
33         END IF;
34         RETURN NEW;
35     END
```

```
36
37  $$ LANGUAGE plpgsql;
38
39  CREATE OR REPLACE TRIGGER ammissione
40  AFTER INSERT ON presenze
41  FOR EACH ROW
42  EXECUTE PROCEDURE check_ammissione();
```

8.2.2 MAX_PARTICIPANTI

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di un Iscrizione dello studente ad un Corso,
3  controlla se il numero massimo di partecipanti è stato raggiunto.
4  Qualora dovesse superare il limite, il trigger elimina l'iscrizione appena inserita. */
5
6  CREATE FUNCTION check_max_partecipanti()
7      RETURNS TRIGGER AS
8      $$
9      DECLARE iscritti INTEGER;
10             max_p INTEGER;
11  BEGIN
12             SELECT COUNT (matricola) INTO iscritti
13             FROM iscrizioni i JOIN corsi c ON i.id_corso = c.id_corso
14             WHERE i.id_corso = new.id_corso;
15
16             SELECT c.max_partecipanti INTO max_p
17             FROM iscrizioni i JOIN corsi c ON i.id_corso = c.id_corso
18             WHERE i.id_corso = NEW.id_corso;
19
20             IF (iscritti > max_p) THEN
21                 RAISE SQLSTATE '10006';
22                 ROLLBACK;
23                 RAISE EXCEPTION 'e stato raggiunto il numero massimo
24                 di partecipanti al corso';
25             END IF;
26             RETURN NEW;
27  END
28  $$ LANGUAGE plpgsql;
29
30  CREATE TRIGGER max_partecipanti
31  AFTER INSERT ON iscrizioni
32  FOR EACH ROW
33  EXECUTE PROCEDURE check_max_partecipanti();
```

8.2.3 ISCRIZIONE_IN_TERMINATO

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di un Iscrizione dello studente ad un Corso,
3  controlla se il corso è terminato.
4  Qualora dovesse verificarsi questa illegalità,
5  il trigger elimina l'iscrizione appena inserita. */
6
7  CREATE FUNCTION check_iscrizione_terminato()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE violazione INTEGER;
11     BEGIN
12         SELECT COUNT (*) INTO violazione
13         FROM corsi c JOIN iscrizioni i ON c.id_corso = i.id_corso
14         WHERE i.id_corso = new.id_corso AND c.terminato = true;
15
16         IF violazione > 0 THEN
17             RAISE SQLSTATE '10009';
18             ROLLBACK;
19             RAISE EXCEPTION 'Il corso già è terminato';
20         END IF;
21         RETURN NEW;
22     END
23
24     $$ LANGUAGE plpgsql;
25
26     CREATE TRIGGER iscrizione_in_terminato
27     AFTER INSERT ON iscrizioni
28     FOR EACH ROW
29     EXECUTE PROCEDURE check_iscrizione_terminato();
```

8.2.4 LEZIONE_ISCRIZIONE

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'inserimento di una Presenza dello studente ad una Lezione,
3  controlla se si è iscritti al Corso della Lezione.
4  Qualora dovesse verificarsi questa illegalità,
5  il trigger elimina la presenza appena inserita.  */
6
7  CREATE FUNCTION check_lezione_iscrizione()
8      RETURNS TRIGGER AS
9      $$
10     DECLARE contatore INTEGER;
11     BEGIN
12         SELECT COUNT (*) INTO contatore
13         FROM presenze p JOIN lezioni l ON p.id_lezione = l.id_lezione
14         WHERE p.id_lezione = NEW.id_lezione AND l.id_corso IN (
15
16             SELECT i.id_corso
17             FROM iscrizioni i
18             WHERE i.matricola = NEW.matricola);
19
20         IF contatore = 0 THEN
21             RAISE SQLSTATE '10010';
22             ROLLBACK;
23             RAISE EXCEPTION 'Non sei iscritto al corso';
24         END IF;
25         RETURN NEW;
26     END;
27
28     $$ LANGUAGE plpgsql;
29
30     CREATE TRIGGER lezione_iscrizione
31     AFTER INSERT ON presenze
32     FOR EACH ROW
33     EXECUTE PROCEDURE check_lezione_iscrizione();
```

8.2.5 ELIMINAZIONE_PRESENZA

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'eliminazione di una Presenza dello studente ad una Lezione,
3  controlla se la Lezione sia già terminata,
4  in questo caso non è possibile annullare la prenotazione.
5  Qualora dovesse verificarsi questa illegalità,
6  il trigger non permette l'eliminazione della presenza. */
7
8  CREATE OR REPLACE FUNCTION check_eliminazione_presenza()
9      RETURNS TRIGGER AS
10     $$
11     DECLARE data_lezione DATE;
12     BEGIN
13         SELECT data INTO data_lezione
14         FROM presenze p JOIN lezioni l ON p.id_lezione = l.id_lezione
15         WHERE p.id_lezione = OLD.id_lezione;
16
17         IF(EXISTS (SELECT * FROM studenti WHERE matricola = old.matricola)) THEN
18             IF (data_lezione < CURRENT_DATE) THEN
19                 RAISE SQLSTATE '10011';
20                 ROLLBACK;
21                 RAISE EXCEPTION 'La lezione si e gia conclusa';
22             END IF;
23         END IF;
24         RETURN OLD;
25     END
26     $$ LANGUAGE plpgsql;
27
28     CREATE OR REPLACE TRIGGER eliminazione_presenza
29     AFTER DELETE ON presenze
30     FOR EACH ROW
31     EXECUTE PROCEDURE check_eliminazione_presenza();
```

8.2.6 LEZIONE_CORSO_TERMINATO

```
1  /* La seguente function restituisce un trigger che,
2  al momento dell'iscrizione di una Presenza dello studente ad una Lezione,
3  controlla se la Lezione faccia parte di un corso già terminata,
4  in questo caso non è possibile iscriversi alla lezione.
5  Qualora dovesse verificarsi questa illegalità,
6  il trigger non permette l'iscrizione alla lezione. */
7
8  CREATE FUNCTION check_lezione_corso_terminato()
9      RETURNS TRIGGER AS
10     $$
11     DECLARE violazione BOOLEAN;
12     BEGIN
13         SELECT co.terminato INTO violazione
14         FROM corsi co JOIN lezioni l on co.id_corso = l.id_corso
15         WHERE l.id_lezione = new.id_lezione;
16
17         IF (violazione) THEN
18             RAISE SQLSTATE '10013';
19             ROLLBACK;
20             RAISE EXCEPTION 'ATTENZIONE : il corso della lezione che
21             hai scelto e termianto';
22         END IF;
23         RETURN NEW;
24     END
25     $$ LANGUAGE plpgsql;
26
27     CREATE TRIGGER lezione_corso_terminato
28     AFTER INSERT ON presenze
29     FOR EACH ROW
30     EXECUTE PROCEDURE check_lezione_corso_terminato();
```

9 Definizione delle Viste

9.1 PARTECIPANTI

```
1 CREATE VIEW partecipanti(id_corso, partecipanti) AS (  
2     SELECT i.id_corso, count (matricola)  
3     from iscrizioni i  
4     group by (id_corso)  
5 )
```

9.2 PRESENZE_TOALI

```
1 CREATE VIEW presenze_totali(presenze, id_corso) AS(  
2     SELECT COUNT(matricola), co.id_corso  
3     FROM (presenze p JOIN lezioni l ON p.id_lezione = l.id_lezione) JOIN  
4     corsi co ON l.id_corso = co.id_corso  
5     GROUP BY co.id_corso  
6 )
```

9.3 PRESENZE_STUDENTE_CORSO

```
1 CREATE VIEW presenze_studente_corso(matricola, id_corso, presenze) AS(  
2     SELECT p.matricola, l.id_corso, COUNT(p.matricola)  
3     FROM presenze p join lezioni l on p.id_lezione = l.id_lezione  
4     GROUP BY p.matricola, l.id_corso  
5 )
```

10 Definizione delle Sequenze

10.1 Sequenza ID_CORSO

```
1 CREATE SEQUENCE sequenza_id_corso
2     START 1
3     INCREMENT 1
4     OWNED BY corsi.id_corso;
```

10.2 Sequenza ID_OPERATORE

```
1 CREATE SEQUENCE sequenza_id_operatore
2     START 1
3     INCREMENT 1
4     OWNED BY operatori.id_operatore;
```

10.3 Sequenza ID_LEZIONE

```
1 CREATE SEQUENCE sequenza_id_lezione
2     START 1
3     INCREMENT 1
4     OWNED BY lezioni.id_lezione;
```

10.4 Sequenza MATRICOLA

```
1 CREATE SEQUENCE sequenza_matricola
2     START 1
3     INCREMENT 1
4     OWNED BY studenti.matricola;
```

10.5 Sequenza ID_DOMANDA

```
1 CREATE SEQUENCE sequenza_id_domanda
2     START 1
3     INCREMENT 1
4     OWNED BY domande_sicurezza.id_domanda;
```

Parte V

Popolamento del DataBase

Si lascia di seguito il codice utilizzato per popolare la Base di Dati a fini dimostrativi e didattici.

Si chiarisce che alcuni di questi inserimenti violano **volontariamente** vincoli presenti nella base in quanto ricordiamo che il sistema permette solo di progettare corsi e lezioni future, non passate.

11 Tabelle

11.1 AREE_TEMATICHE

```
1  INSERT INTO aree_tematiche
2  VALUES('programmazione');
3
4  INSERT INTO aree_tematiche
5  VALUES('matematica');
6
7  INSERT INTO aree_tematiche
8  VALUES('calcolo');
9
10 INSERT INTO aree_tematiche
11 VALUES('strutture dati');
12
13 INSERT INTO aree_tematiche
14 VALUES('algoritmi');
15
16 INSERT INTO aree_tematiche
17 VALUES('filosofia');
18
19 INSERT INTO aree_tematiche
20 VALUES('scienze');
21
22 INSERT INTO aree_tematiche
23 VALUES('lingue');
24
```



```
25 INSERT INTO aree_tematiche
26 VALUES('ingegneria');
27
28 INSERT INTO aree_tematiche
29 VALUES('economia');
```

11.2 PAROLE_CHIAVE

```
1 INSERT INTO parole_chiave
2 VALUES('java');
3
4 INSERT INTO parole_chiave
5 VALUES('algebra relazionale');
6
7 INSERT INTO parole_chiave
8 VALUES('grafi');
9
10 INSERT INTO parole_chiave
11 VALUES('fsm');
12
13 INSERT INTO parole_chiave
14 VALUES('reticoli');
15
16 INSERT INTO parole_chiave
17 VALUES('funzioni');
18
19 INSERT INTO parole_chiave
20 VALUES('commercio');
21
22 INSERT INTO parole_chiave
23 VALUES('mercati');
24
25 INSERT INTO parole_chiave
26 VALUES('software');
27
28 INSERT INTO parole_chiave
29 VALUES('network');
```

```
30
31 INSERT INTO parole_chiave
32 VALUES('automazione');
33
34 INSERT INTO parole_chiave
35 VALUES('impianti');
```

11.3 OPERATORI

```
1 INSERT INTO operatori
2 VALUES(nextval('sequenza_id_operatore'), 's.demartino','password');
3
4 INSERT INTO operatori
5 VALUES(nextval('sequenza_id_operatore'),'a.peron','password');
6
7 INSERT INTO operatori
8 VALUES(nextval('sequenza_id_operatore'), 'm.benerecetti','password');
9
10 INSERT INTO operatori
11 VALUES(nextval('sequenza_id_operatore'), 'a.deluca','password');
12
13 INSERT INTO operatori
14 VALUES(nextval('sequenza_id_operatore'), 'f.oliva','password');
15
16 INSERT INTO operatori
17 VALUES(nextval('sequenza_id_operatore'), 'g.laccetti','password');
18
19 INSERT INTO operatori
20 VALUES(nextval('sequenza_id_operatore'), 'm.trombetti','password');
21
22 INSERT INTO operatori
23 VALUES(nextval('sequenza_id_operatore'), 'g.adamo','password');
```

11.4 CORSI

```
1  INSERT INTO corsi
2  VALUES(nextval('sequenza_id_corso'), 1, 'object_orientation',
3  'corso sul paradigma della programmazione ad oggetti', 1, 5, '2023');
4
5  INSERT INTO corsi
6  VALUES(nextval('sequenza_id_corso'), 2, 'basi_dati',
7  'corso sulla gestione di database', 1, 5, '2022');
8
9  INSERT INTO corsi
10 VALUES(nextval('sequenza_id_corso'), 3, 'algoritmi_strutture_dati',
11 'corso sullo complessita di algoritmi e strutture dati', 1, 5, '2022');
12
13 INSERT INTO corsi
14 VALUES(nextval('sequenza_id_corso'), 4, 'elementi_di_informatica_teorica',
15 'informatica teorica', 1, 5, '2022');
16
17 INSERT INTO corsi
18 VALUES(nextval('sequenza_id_corso'), 6, 'programmazione_1',
19 'approccio alla programmazione in c++', 2, 5, '2021');
20
21 INSERT INTO corsi
22 VALUES(nextval('sequenza_id_corso'), 1, 'analisi_1',
23 'teoremi base di analisi matematica', 2, 10, '2019');
24
25 INSERT INTO corsi
26 VALUES(nextval('sequenza_id_corso'), 1, 'ingegneria_del_software',
27 'studio dei software', 1, 5, '2021');
28
29 INSERT INTO corsi
30 VALUES(nextval('sequenza_id_corso'), 2, 'algebra',
31 'corso algebra lineare e booleana', 3, 10, '2019', true);
32
33 INSERT INTO corsi
34 VALUES(nextval('sequenza_id_corso'), 4, 'chimica',
35 'approfondimento elementi di chimica complementare', 1, 5, '2018', true);
36
37 INSERT INTO corsi
```

```

38 VALUES(nextval('sequenza_id_corso'), 8, 'astrofisica',
39 'studio dei fenomeni astronomici e astrofisici', 2, 10, '2022');
40
41 INSERT INTO corsi
42 VALUES(nextval('sequenza_id_corso'), 1, 'geometria',
43 'Conoscenze e capacità su teoria degli spazi vettoriali.', 1, 10, '2022', true);
44
45 INSERT INTO corsi
46 VALUES(nextval('sequenza_id_corso'), 2, 'diritto_privato',
47 'comprensione i basilari e principali istituti giusprivatistici', 1, 5, '2020', true);
48
49 INSERT INTO corsi
50 VALUES(nextval('sequenza_id_corso'), 1, 'economia_monetaria',
51 'dinamiche sui mercati monetari e finanziari internazionali', 2, 10, '2021');
52
53 INSERT INTO corsi
54 VALUES(nextval('sequenza_id_corso'), 8, 'logistica_industriale',
55 'governare e pianificare le principali attività logistiche', 1, 10, '2021');
56
57 INSERT INTO corsi
58 VALUES(nextval('sequenza_id_corso'), 4, 'tecnologia_meccanica',
59 'processi di produzione industriale completamente automatizzati', 1, 5, '2021', true);
60
61 INSERT INTO corsi
62 VALUES(nextval('sequenza_id_corso'), 7, 'ricerca_operativa',
63 ' studio scientifico dei metodi per risolvere problemi decisionali', 1, 5, '2022');

```

11.5 STUDENTI

```
1  INSERT INTO studenti
2  VALUES(nextval('sequenza_matricola'),'renato', 'antonelli');
3
4  INSERT INTO studenti
5  VALUES(nextval('sequenza_matricola'),'dario', 'franzese');
6
7  INSERT INTO studenti
8  VALUES(nextval('sequenza_matricola'),'gian marco', 'addati');
9
10 INSERT INTO studenti
11 VALUES(nextval('sequenza_matricola'),'alessia', 'ascolese');
12
13 INSERT INTO studenti
14 VALUES(nextval('sequenza_matricola'),'simone', 'negroni');
15
16 INSERT INTO studenti
17 VALUES(nextval('sequenza_matricola'),'simone', 'giordano');
18
19 INSERT INTO studenti
20 VALUES(nextval('sequenza_matricola'),'matteo', 'spavone');
21
22 INSERT INTO studenti
23 VALUES(nextval('sequenza_matricola'),'lorenzo', 'criscuolo');
24
25 INSERT INTO studenti
26 VALUES(nextval('sequenza_matricola'),'mario', 'molinaro');
27
28 INSERT INTO studenti
29 VALUES(nextval('sequenza_matricola'),'assunta', 'venturoli');
```

11.6 ISCRIZIONI

```
1  INSERT INTO iscrizioni
2  VALUES(1, 4, true);
3
4  INSERT INTO iscrizioni
5  VALUES(1, 5);
6
7  INSERT INTO iscrizioni
8  VALUES(1, 6, true);
9
10 INSERT INTO iscrizioni
11 VALUES(1, 14);
12
13 INSERT INTO iscrizioni
14 VALUES(2, 1);
15
16 INSERT INTO iscrizioni
17 VALUES(2, 4);
18
19 INSERT INTO iscrizioni
20 VALUES(2, 5);
21
22 INSERT INTO iscrizioni
23 VALUES(2, 6);
24
25 INSERT INTO iscrizioni
26 VALUES(2, 7);
27
28 INSERT INTO iscrizioni
29 VALUES(2, 10, false);
30
31 INSERT INTO iscrizioni
32 VALUES(2, 11);
33
34 INSERT INTO iscrizioni
35 VALUES(3, 2);
36
37 INSERT INTO iscrizioni
```

```
38 VALUES(3, 7);
39
40 INSERT INTO iscrizioni
41 VALUES(3, 8);
42
43 INSERT INTO iscrizioni
44 VALUES(4, 6);
45
46 INSERT INTO iscrizioni
47 VALUES(5, 6);
48
49 INSERT INTO iscrizioni
50 VALUES(6, 6);
51
52 INSERT INTO iscrizioni
53 VALUES(6, 15);
54
55 INSERT INTO iscrizioni
56 VALUES(7, 2);
57
58 INSERT INTO iscrizioni
59 VALUES(7, 6, true);
60
61 INSERT INTO iscrizioni
62 VALUES(7, 7);
63
64 INSERT INTO iscrizioni
65 VALUES(7, 8);
66
67 INSERT INTO iscrizioni
68 VALUES(8, 13);
69
70 INSERT INTO iscrizioni
71 VALUES(9, 1);
72
73 INSERT INTO iscrizioni
74 VALUES(9, 8);
75
76 INSERT INTO iscrizioni
```

```
77 VALUES(10, 1);
78
79 INSERT INTO iscrizioni
80 VALUES(10, 5);
81
82 INSERT INTO iscrizioni
83 VALUES(10, 6);
```

11.7 PRESENZE

```
1 INSERT INTO presenze
2 VALUES(1, 14);
3
4 INSERT INTO presenze
5 VALUES(1, 16);
6
7 INSERT INTO presenze
8 VALUES(1, 19);
9
10 INSERT INTO presenze
11 VALUES(1, 20);
12
13 INSERT INTO presenze
14 VALUES(1, 25);
15
16 INSERT INTO presenze
17 VALUES(1, 26);
18
19 INSERT INTO presenze
20 VALUES(2, 14);
21
22 INSERT INTO presenze
23 VALUES(2, 15);
24
25 INSERT INTO presenze
```



```
26 VALUES(2, 16);
27
28 INSERT INTO presenze
29 VALUES(2, 17);
30
31 INSERT INTO presenze
32 VALUES(2, 20);
33
34 INSERT INTO presenze
35 VALUES(2, 25);
36
37 INSERT INTO presenze
38 VALUES(2, 26);
39
40 INSERT INTO presenze
41 VALUES(3, 9);
42
43 INSERT INTO presenze
44 VALUES(3, 28);
45
46 INSERT INTO presenze
47 VALUES(3, 29);
48
49 INSERT INTO presenze
50 VALUES(3, 24);
51
52 INSERT INTO presenze
53 VALUES(4, 21);
54
55 INSERT INTO presenze
56 VALUES(4, 23);
57
58 INSERT INTO presenze
59 VALUES(4, 24);
60
61 INSERT INTO presenze
62 VALUES(4, 25);
63
64 INSERT INTO presenze
```

```
65 VALUES(5, 8);
66
67 INSERT INTO presenze
68 VALUES(5, 23);
69
70 INSERT INTO presenze
71 VALUES(5, 24);
72
73 INSERT INTO presenze
74 VALUES(5, 26);
75
76 INSERT INTO presenze
77 VALUES(6, 2);
78
79 INSERT INTO presenze
80 VALUES(6, 20);
81
82 INSERT INTO presenze
83 VALUES(6, 21);
84
85 INSERT INTO presenze
86 VALUES(6, 23);
87
88 INSERT INTO presenze
89 VALUES(7, 9);
90
91 INSERT INTO presenze
92 VALUES(7, 21);
93
94 INSERT INTO presenze
95 VALUES(7, 24);
96
97 INSERT INTO presenze
98 VALUES(9, 1);
99
100 INSERT INTO presenze
101 VALUES(9, 4);
102
103 INSERT INTO presenze
```

```
104 VALUES(9, 6);
105
106 INSERT INTO presenze
107 VALUES(9, 28);
108
109 INSERT INTO presenze
110 VALUES(9, 29);
111
112 INSERT INTO presenze
113 VALUES(10, 3);
114
115 INSERT INTO presenze
116 VALUES(10, 4);
117
118 INSERT INTO presenze
119 VALUES(10, 5);
120
121 INSERT INTO presenze
122 VALUES(10, 8);
123
124 INSERT INTO presenze
125 VALUES(10, 18);
126
127 INSERT INTO presenze
128 VALUES(10, 19);
129
130 INSERT INTO presenze
131 VALUES(10, 20);
132
133 INSERT INTO presenze
134 VALUES(10, 24);
135
136 INSERT INTO presenze
137 VALUES(10, 25);
```

11.8 TEMI

```
1  INSERT INTO temi
2  VALUES('programmazione', 1);
3
4  INSERT INTO temi
5  VALUES('strutture dati', 1);
6
7  INSERT INTO temi
8  VALUES('calcolo', 8);
9
10 INSERT INTO temi
11 VALUES('matematica', 8);
12
13 INSERT INTO temi
14 VALUES('matematica', 6);
15
16 INSERT INTO temi
17 VALUES('calcolo', 6);
18
19 INSERT INTO temi
20 VALUES('programmazione', 7);
21
22 INSERT INTO temi
23 VALUES('strutture dati', 3);
24
25 INSERT INTO temi
26 VALUES('algoritmi', 3);
27
28 INSERT INTO temi
29 VALUES('programmazione', 5);
30
31 INSERT INTO temi
32 VALUES('programmazione', 2);
33
34 INSERT INTO temi
35 VALUES('calcolo', 2);
36
37 INSERT INTO temi
```

```
38 VALUES('scienze', 10);
39
40 INSERT INTO temi
41 VALUES('ingegneria', 7);
42
43 INSERT INTO temi
44 VALUES('filosofia', 4);
45
46 INSERT INTO temi
47 VALUES('economia', 13);
48
49 INSERT INTO temi
50 VALUES('scienze', 6);
51
52 INSERT INTO temi
53 VALUES('calcolo', 16);
54
55 INSERT INTO temi
56 VALUES('ingegneria', 15);
57
58 INSERT INTO temi
59 VALUES('strutture dati', 5);
60
61 INSERT INTO temi
62 VALUES('scienze', 16);
63
64 INSERT INTO temi
65 VALUES('programmazione', 15);
66
67 INSERT INTO temi
68 VALUES('lingue', 12);
69
70 INSERT INTO temi
71 VALUES('calcolo', 4);
```

11.9 CARATTERIZZA

```
1  INSERT INTO caratterizza
2  VALUES('java', 1);
3
4  INSERT INTO caratterizza
5  VALUES('fsm', 4);
6
7  INSERT INTO caratterizza
8  VALUES('grafi', 3);
9
10 INSERT INTO caratterizza
11 VALUES('funzioni', 6);
12
13 INSERT INTO caratterizza
14 VALUES('reticoli', 8);
15
16 INSERT INTO caratterizza
17 VALUES('algebra relazionale', 8);
18
19 INSERT INTO caratterizza
20 VALUES('algebra relazionale', 2);
21
22 INSERT INTO caratterizza
23 VALUES('software', 1);
24
25 INSERT INTO caratterizza
26 VALUES('impianti', 14);
27
28 INSERT INTO caratterizza
29 VALUES('network', 10 );
30
31 INSERT INTO caratterizza
32 VALUES('software', 11);
33
34 INSERT INTO caratterizza
35 VALUES('reticoli', 6);
36
37 INSERT INTO caratterizza
```

```
38 VALUES('algebra relazionale', 7);
39
40 INSERT INTO caratterizza
41 VALUES('software', 2);
```

11.10 DOMANDE_SICUREZZA

```
1 INSERT INTO domande_sicurezza
2 VALUES ('dove sei nato?' , nextval('sequenza_id_domanda'));
3
4 INSERT INTO domande_sicurezza
5 VALUES ('come si chiama il tuo primo genito ?' , nextval('sequenza_id_domanda'));
6
7 INSERT INTO domande_sicurezza
8 VALUES ('come si chiama il tuo animale domestico?' , nextval('sequenza_id_domanda'));
```

11.11 DOMANDE_OPERATORI

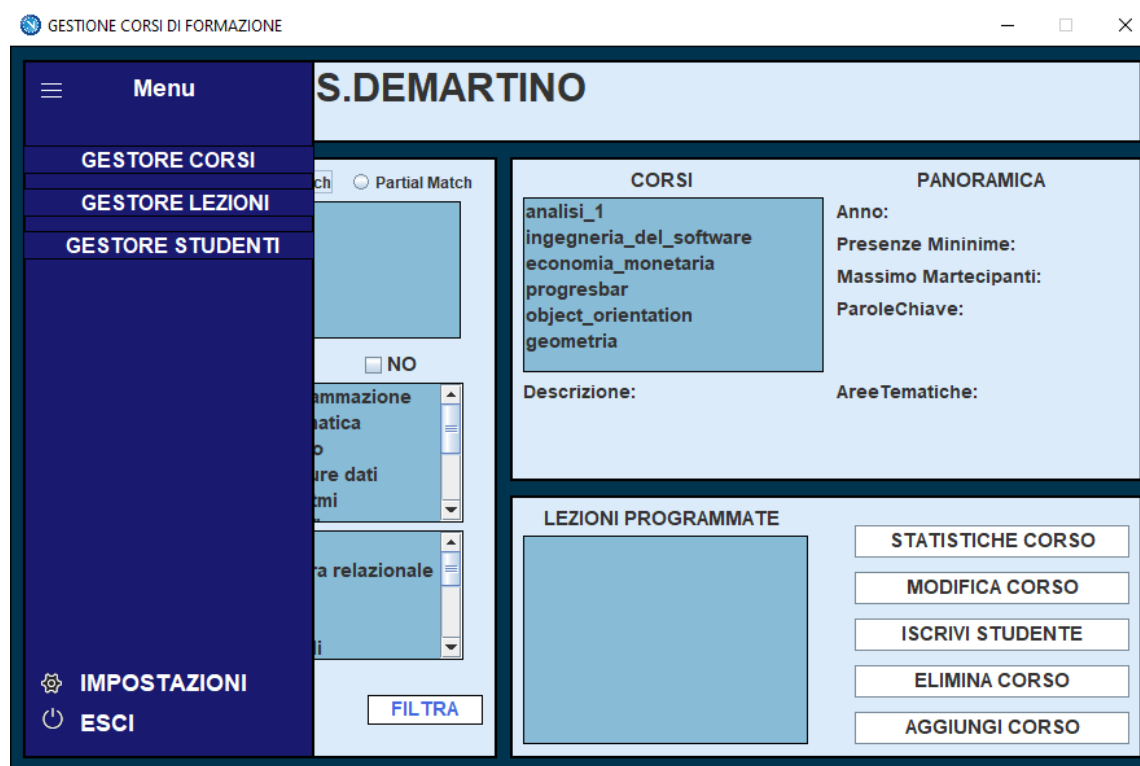
```
1  INSERT INTO domande_operatori
2  VALUES('napoli', '1', '1');
3
4  INSERT INTO domande_operatori
5  VALUES('kappa', '3', '2');
6
7  INSERT INTO domande_operatori
8  VALUES('luca', '2', '3');
9
10 INSERT INTO domande_operatori
11 VALUES('napoli', '1', '4');
12
13 INSERT INTO domande_operatori
14 VALUES('giulia', '2', '5');
15
16 INSERT INTO domande_operatori
17 VALUES('kira', '3', '6');
18
19 INSERT INTO domande_operatori
20 VALUES('marco', '2', '7');
21
22 INSERT INTO domande_operatori
23 VALUES('giotto', '3', '8');
```

Parte VI

Guida all'utilizzo

Nella seguente parte sono indicate le operazioni di creazione, modifica e visualizzazione delle statistiche delle 3 entità principali dell'applicativo (Corsi, Lezioni e Studenti)

Alla gestione di ognuna di queste è dedicata una schermata accessibile attraverso il menù a comparsa laterale



12 Registrazione

Vediamo anzitutto come creare un user che possa accedere all'applicazione ed avere pieno controllo di tutte le funzionalità.

The screenshot shows a web browser window with the title "GESTIONE CORSI DI FORMAZIONE". The main content area is titled "ISCRIVITI" (Sign Up). It contains four input fields arranged in a 2x2 grid:

- Inserire Nome Utente**: A text input field with a red error message below it: "I Dati non devono contenere caratteri Speciali (!,\",@)".
- Inserire Password**: A text input field with a red error message below it: "La Password deve contenere almeno 6 caratteri!".
- Scegliere Domanda di Sicurezza**: A dropdown menu with the selected option "dove sei nato?".
- Inserire Risposta**: A text input field.

Below the input fields, there is a "CONFERMA" (Confirm) button. In the bottom left corner, there is an "INDIETRO" (Back) button.

1. Dalla schermata iniziale, selezionare "REGISTRATI"
2. Inserire negli appositi spazi i valori che caratterizzano un operatore.
3. Selezionare "CONFERMA"
4. Comparsa della schermata di conferma o fallimento della creazione
5. Selezionare accesso all'applicazione oppure ritorno alla pagina di Log In.

13 Creazione

La creazione di ogni elemento segue un iter molto simile, di raccolta dati dall'interfaccia e salvataggio in database, ad ogni entità é associata una schermata differente di selezione

13.1 Creazione Corso

The screenshot shows a web application window titled "GESTIONE CORSI DI FORMAZIONE". Inside, there is a form titled "CREAZIONE CORSO". The form contains several input fields and checkboxes:

- Nome:** A text input field.
- Anno:** A dropdown menu showing "2022".
- Presenze Minime:** A text input field.
- Terminato?:** A checkbox labeled "SI".
- Massimo Partecipanti:** A text input field.
- Descrizione:** A text input field.
- Parola Chiave:** A list box containing the following items: programmazione, matematica, calcolo, strutture dati, algoritmi, filosofia, scienze, lingue, and ingegneria.
- Area Tematica:** A list box containing the following items: java, algebra relazionale, grafi, fsm, reticoli, funzioni, commercio, mercati, and software.

At the bottom of the form, there are two buttons: "INDIETRO" on the left and "CONFERMA" on the right.

1. Aprire "GESTORE CORSI"
2. Selezionare "CREA CORSO"
3. Inserire negli appositi spazi i valori che caratterizzano un corso.
4. Selezionare "CONFERMA"
5. Comparsa della schermata di conferma o fallimento della creazione

13.2 Creazione Lezione

GESTIONE CORSI DI FORMAZIONE

CREAZIONE LEZIONE

Titolo:

Data: febbraio 2019

Orario: 08:30:00 ▼

Durata: 01:00:00 ▼

Descrizione:

	lun	mar	mer	gio	ven	sab	dom
05					1	2	3
06	4	5	6	7	8	9	10
07	11	12	13	14	15	16	17
08	18	19	20	21	22	23	24
09	25	26	27	28			

INDIETRO **CONFERMA**

1. Aprire "GESTORE LEZIONI"
2. Selezionare un corso
3. Inserisci i valori che caratterizzano una lezione
4. Selezionare "CONFERMA"
5. Comparsa della schemata di conferma o fallimento della creazione

13.3 Creazione Studente

INSERIRE DATI STUDENTE

Inserire Nome Studente:

Inserire cognome Studente:

Corsi dove poter iscrivere lo Studente

- ☐ economia_monetaria
- ☐ ingegneria_del_softwa
- ☐ asd
- ☐ analisi_1
- ☐ object_orientation

INDIETRO **CONFERMA**

1. Aprire "GESTORE STUDENTI"
2. Selezionare "CREA STUDENTE"
3. Inserire i valori che caratterizzano uno studente
4. Eventualmente, selezione corsi dell'operatore in uso al quale si vuole iscrivere uno studente
5. Selezionare "CONFERMA"
6. Comparsa della schemata di conferma o fallimento della creazione

14 Modifica

La modifica viene gestita da individualmente da ogni "gestore".
Tuttavia, eventuali modifiche possono avere effetti anche su altre entità (ad esempio modificando terminando un corso oppure modificando il numero di presenze minime per l'ammissione all'esame)

14.1 Modifica Corso

MODIFICHE CORSO

Nome: Anno:
Non è possibile modificare l'anno se ci sono lezioni svolte o programmate

Presenze Minime: Terminato? ☐ SI

Massimo Partecipanti: Descrizione:

Area Tematica:

- ☐ programmazione
- ☒ matematica
- ☒ calcolo
- ☐ strutture dati
- ☐ algoritmi
- ☐ filosofia
- ☒ scienze
- ☐ lingue
- ☐ ingegneria

Parola Chiave:

- ☐ java
- ☐ algebra relazionale
- ☐ grafi
- ☐ fsm
- ☒ reticoli
- ☒ funzioni
- ☐ commercio
- ☐ mercati
- ☐ software

1. Aprire "GESTORE CORSI"
2. Selezionare un corso
3. Selezionare "MODIFICA CORSO"
4. Inserire tutti i valori che eventualmente si vogliono modificare del Corso.
"CONFERMA"
5. Comparsa della schemata di conferma o fallimento della modifica

14.2 Modifica Lezione

GESTIONE CORSI DI FORMAZIONE

MODIFICHE LEZIONE

Titolo:

Data:

Orario:

Durata:

Descrizione:

	lun	mar	mer	gio	ven	sab	dom
44					1	2	3
45	4	5	6	7	8	9	10
46	11	12	13	14	15	16	17
47	18	19	20	21	22	23	24
48	25	26	27	28	29	30	

INDIETRO **CONFERMA**

1. Aprire "GESTORE LEZIONI"
2. Selezionare un corso
3. Selezionare una lezione
4. Selezionare "PANORAMICA"
5. Selezionare "MODIFICA LEZIONE"
6. Inserire tutti i valori che eventualmente si vogliono modificare della lezione.
"CONFERMA"
7. Comparsa della schemata di conferma o fallimento della modifica

14.3 Modifica Studente



GESTIONE CORSI DI FORMAZIONE

INSERIRE DATI STUDENTE

Nome Studente:

Cognome Studente:

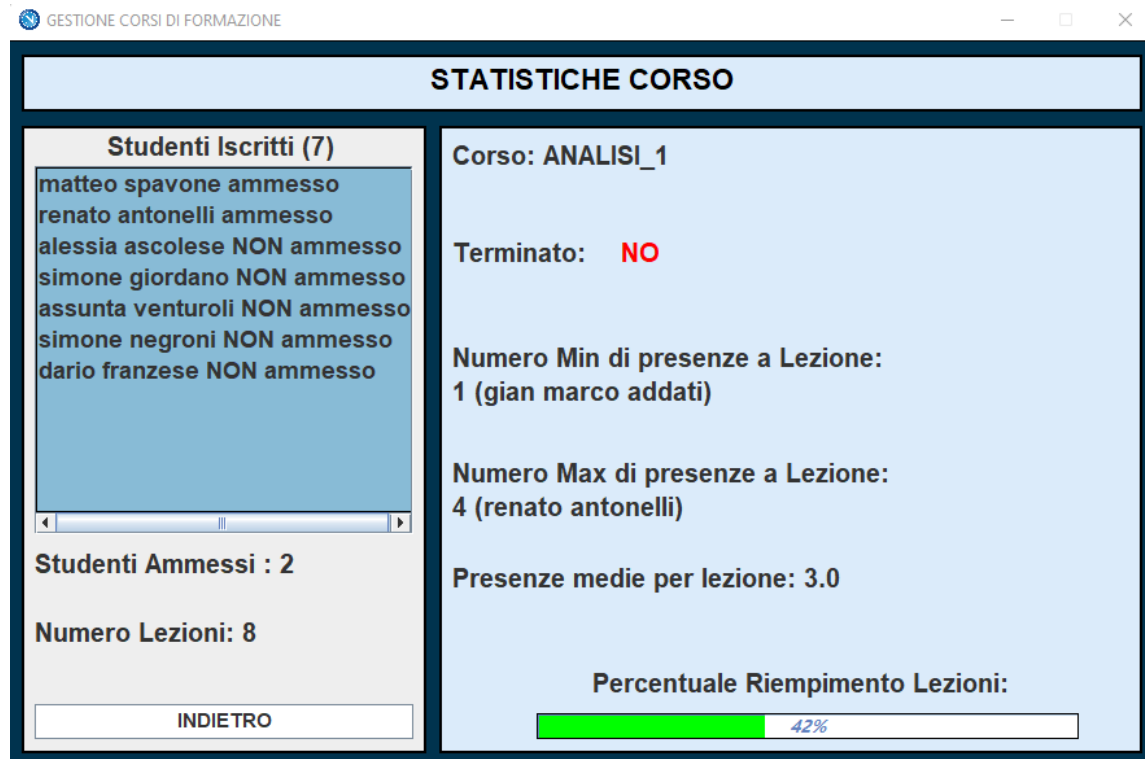
INDIETRO **CONFERMA**

1. Aprire "GESTORE STUDENTI"
2. Selezionare uno studente
3. Selezionare "PANORAMICA"
4. Selezionare "MODIFICA STUDENTE"
5. Inserire tutti i valori che eventualmente si vogliono modificare dello studente. "CONFERMA"
6. Comparsa della schermata di conferma o fallimento della modifica

15 Statistiche


Per ogni entità é possibile visualizzare delle statistiche specifiche ad essa associata

15.1 Statistiche Corso



1. Aprire "GESTORE CORSI"
2. Selezionare un corso
3. Selezionare "STATISTICHE CORSO"
4. Visualizzare le statistiche "INDIETRO" per tornare al Gestore

15.2 Statistiche Lezione

 GESTIONE CORSI DI FORMAZIONE

PANORAMICA LEZIONE

Studenti Iscritti

Dettagli Lezione

Titolo: sistemi lineari

Durata: 01:30:00

Orario: 08:30:00

Data: 2022-05-28

Descrizione: sistema composto da più equazioni lineari

INDIETRO

MODIFICA LEZIONE

AGGIUNGI STUDENTE ALLA LEZIONE

1. Aprire "GESTORE LEZIONI"
2. Selezionare un corso
3. Selezionare una lezione
4. Selezionare "PANORAMICA"
5. Visualizzare le statistiche "INDIETRO" per tornare al Gestore

15.3 Statistiche Studente

GESTIONE CORSI DI FORMAZIONE

STUDENTE: 1 ANTONELLI

MODIFICA STUDENTE

ELENCO CORSI:

analisi_1
asd

PRENOTAZIONI EFFETTUATE

successione limitata
serie
calcolo integrale
derivate

STATUS CORSO

Corso: analisi_1

Numero Lezioni: 8

Presenze: 4

Ammesso : SI

ISCRIVI AD UN CORSO DISISCRIVI DA UN CORSO

PRENOTA LEZIONE ANNULLA PRENOTAZIONE

1. Aprire "GESTORE STUDENTI"
2. Selezionare uno studente
3. Selezionare "PANORAMICA"
4. Visualizzare le statistiche "INDIETRO" per tornare al Gestore