

UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

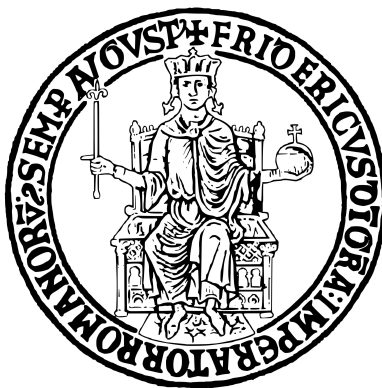
BASI DI DATI

**Progettazione e sviluppo di una base
di dati relazionale per una
applicazione di e-learning**

Marzia PIROZZI N86003545

Noemi SPERA N86003717

Gennaio 2022



Indice

| | | |
|----------|---|-----------|
| 1 | DESCRIZIONE DEL PROGETTO | 3 |
| 1.1 | Introduzione | 3 |
| 2 | PROGETTAZIONE CONCETTUALE | 4 |
| 2.1 | Class Diagram | 4 |
| 2.2 | Descrizione Class Diagram | 5 |
| 2.3 | Class Diagram ristrutturato | 5 |
| 2.3.1 | Analisi delle ridondanze | 6 |
| 2.3.2 | Gerarchie di specializzazione | 6 |
| 2.3.3 | Attributi multipli | 6 |
| 2.3.4 | Attributi calcolabili | 6 |
| 2.3.5 | Attributi composti | 6 |
| 2.3.6 | Chiavi primarie | 7 |
| 3 | DIZIONARIO DEI DATI | 8 |
| 3.1 | Dizionario delle classi | 8 |
| 3.2 | Dizionario delle associazioni | 10 |
| 3.3 | Dizionario dei vincoli | 10 |
| 4 | PROGETTAZIONE LOGICA | 11 |
| 4.1 | Schema logico | 11 |
| 5 | PROGETTAZIONE FISICA | 12 |
| 5.1 | Domini | 12 |
| 5.1.1 | RISPOSTA_ESISTENTE | 12 |
| 5.1.2 | LUNGHEZZA_MASSIMA | 12 |
| 5.1.3 | PASSWORD | 12 |
| 5.2 | Definizione delle tabelle | 13 |
| 5.2.1 | STUDENTE | 13 |

| | | |
|--------|--|----|
| 5.2.2 | INSEGNANTE | 13 |
| 5.2.3 | CORSO | 13 |
| 5.2.4 | TEST | 14 |
| 5.2.5 | QUIZ_RISP_MUL | 14 |
| 5.2.6 | QUIZ_RISP_APE | 14 |
| 5.2.7 | TEST_SVOLTO | 15 |
| 5.2.8 | FREQUENTA | 15 |
| 5.2.9 | COMPOSIZIONE_A | 15 |
| 5.2.10 | COMPOSIZIONE_M | 16 |
| 5.3 | Implementazione dei vincoli | 17 |
| 5.3.1 | LOGIN_STUD | 17 |
| 5.3.2 | LOGIN_INS | 17 |
| 5.3.3 | CORRETTEZZA_DATA_FINE | 17 |
| 5.3.4 | NUMERO_QUIZ | 17 |
| 5.3.5 | CORRETTEZZA_ORARIO_FINE | 17 |
| 5.3.6 | UNICA_RISPOSTA_CORRETTAM | 17 |
| 5.3.7 | UNICA_RISPOSTA_CORRETTAA | 17 |
| 5.3.8 | QUIZ | 18 |
| 5.4 | Funzioni, Procedure ed altre automazioni | 19 |
| 5.4.1 | PUNT_TOT | 19 |
| 5.4.2 | DATA_QUIZ | 20 |
| 5.4.3 | INS_LOG | 21 |
| 5.4.4 | STUD_LOG | 22 |
| 5.4.5 | PUNTEGGIO_ASSEGNATO | 23 |
| 5.4.6 | AUTO_CORREZIONE | 24 |
| 5.4.7 | CORREZIONE_RISP_APE | 25 |
| 5.4.8 | TEST_QUIZ | 26 |

Capitolo 1

DESCRIZIONE DEL PROGETTO

1.1 Introduzione

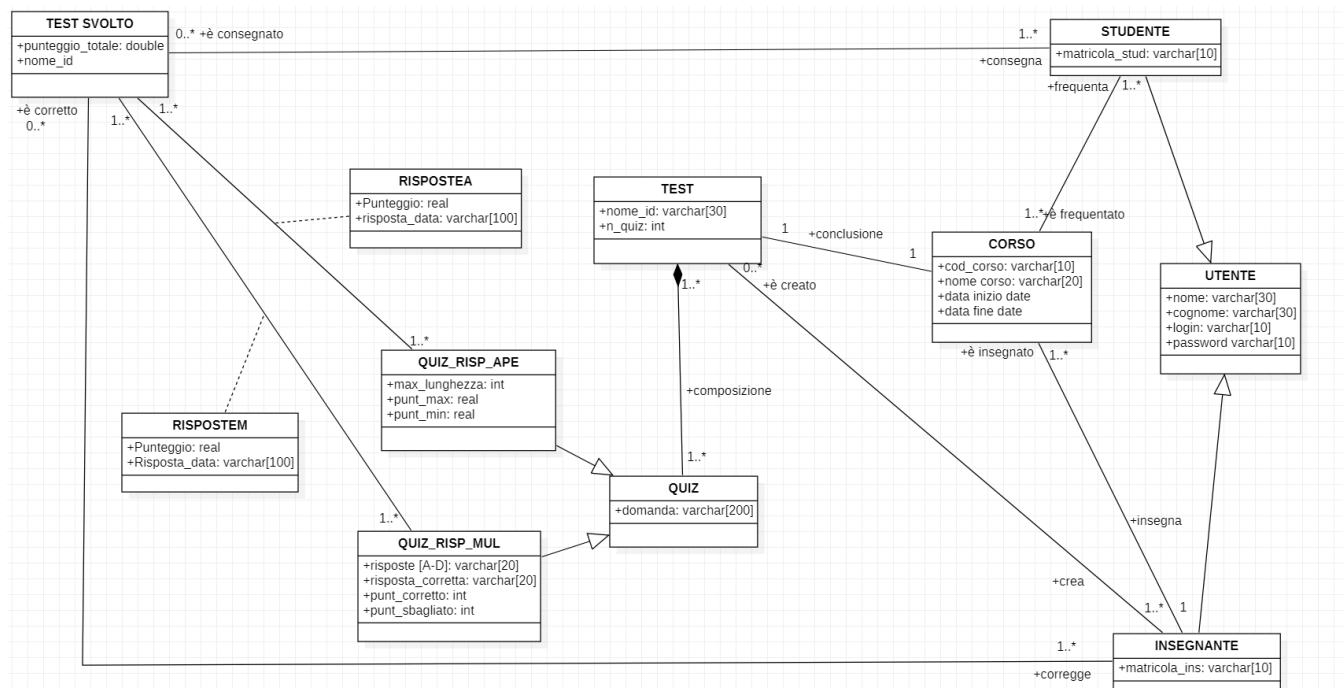
Abbiamo creato una base di dati relazionale in cui **studenti** e **insegnanti** si iscrivono ad una piattaforma di e_learning tramite un **login** e una **password**. Gli insegnanti caricano dei **test**, composti da **quiz**, e gli studenti possono svolgerli (se effettivamente seguono il **corso** tenuto da quell'insegnante). I quiz che compongono un test possono essere a **risposta aperta** o a **risposta multipla**. I quiz a risposta multipla verranno **corretti automaticamente**, mentre i quiz a risposta aperta vanno **corretti dagl'insegnante**. Al termine della correzione potrà essere visualizzato il **punteggio totalizzato** da ogni studente per quel test.

Capitolo 2

PROGETTAZIONE CONCETTUALE

2.1 Class Diagram

Di seguito il class diagram realizzato attraverso il software StarUML.



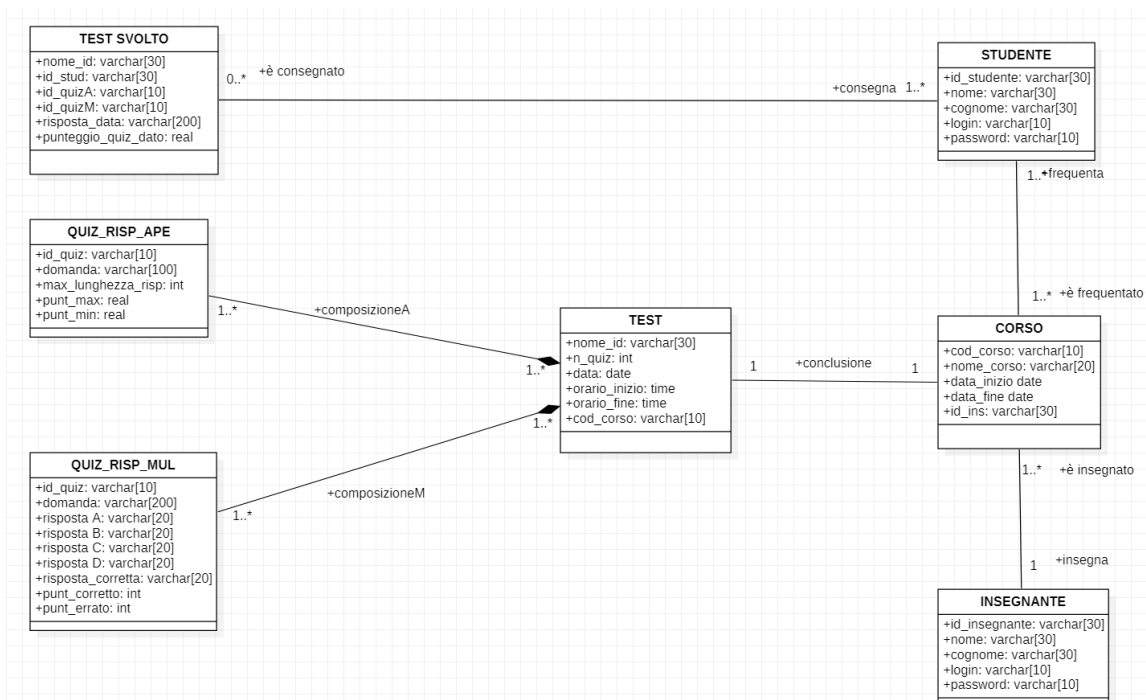
2.2 Descrizione Class Diagram

Il class diagram presenta le classi studente e insegnante come specializzazioni di utente. Essi hanno in comune come attributi: nome, cognome, login e password. Ogni studente segue un particolare corso tenuto da un insegnante, a conclusione del quale gli studenti devono svolgere un test. Ogni test contiene un numero di quiz i quali possono essere di due tipi: a risposta aperta o a risposta multipla. Le risposte date dagli studenti a ciascun quiz di un test vengono salvate in RispostaA per i quiz a risposta aperta e RispostaM per quelli a risposta multipla. Il sistema di e-learning assegna un punteggio ai quiz a risposta multipla (il massimo in caso di risposta corretta, 0 in caso di risposta errata) mentre l'insegnante deve assegnare un punteggio tra il min (deciso dall'insegnante, ad esempio 0) e il max ai quiz a risposta aperta. Il totale è calcolato nella classe test svolto. Questo class diagram è una bozza e in quanto tale presenta delle gerarchie e delle ridondanze, soprattutto nelle associazioni, che verranno risolte con la ristrutturazione.

2.3 Class Diagram ristrutturato

Di seguito il class diagram ristrutturato realizzato attraverso il software StarUML.

ATTENZIONE! matricola_stud e matricola_ins sono state modificate in id_stud e id_ins



2.3.1 Analisi delle ridondanze

Sono state notate due ridondanze. La prima la riscontriamo tra test svolto, rispostaA e rispostaM. Sono state accorpate le due associazioni nell'entità test svolto, in quanto la risposta data dallo studente nel caso di quiz a risposta aperta e a risposta multipla è una serie di caratteri che possono rappresentare nel primo caso un testo e nel secondo caso la lettera corrispondente alla risposta, dunque il datatype è lo stesso. Grazie a tale operazione Test svolto diventa un record delle risposte e dei punteggi ottenuti da ogni studente per ogni quiz che compone il test. La seconda la riscontriamo tra insegnante, corso e test, in quanto si possono raggiungere gli attributi della classe insegnante a partire da test anche eliminando l'associazione tra insegnante e test, perché un insegnante tiene **un** corso a conclusione del quale c'è **un** test quindi è ovvio che lo stesso insegnante creerà quel test.

2.3.2 Gerarchie di specializzazione

Il class diagram presenta due generalizzazioni. La prima vede le classi studente e insegnante come specializzazioni di utente. Tale generalizzazione è stata risolta accorpando la classe padre nelle classi figlie. Stesso ragionamento è stato fatto per la generalizzazione che ha come padre Quiz e come figlie Risposta Multipla e Risposta Aperta.

2.3.3 Attributi multipli

Nella classe Risposta Multipla è presente un attributo multiplo: risposte[A-D]: varchar[20] che rappresenta tutte le possibili risposte ad un quiz a risposta multipla, dalla A alla D. Si è scelto di scomporlo in 4 attributi nella stessa classe che rappresentano ciascuno una risposta: risposta A:(varchar[20]), risposta B:(varchar[20]), risposta C:(varchar[20]), risposta D:(varchar[20]).

2.3.4 Attributi calcolabili

È presente un solo attributo calcolabile, ovvero il punteggio totale di un test svolto, dato dalla somma dei punteggi ottenuti dallo studente nei singoli quiz.

2.3.5 Attributi composti

Non sono presenti attributi composti.

2.3.6 Chiavi primarie

Per identificare univocamente studenti e insegnanti sono state scelte rispettivamente le chiavi primarie `id_studente` e `id_insegnante`. `Nome_id` identifica un test che, alla fine di un corso identificato da `cod_corso`, tutti gli studenti devono svolgere. Ogni quiz all'interno di un test è indentificato da `id_quiz`.

Capitolo 3

DIZIONARIO DEI DATI

3.1 Dizionario delle classi

| Nome | Descrizione |
|-------------------|--|
| Utente | Nome <i>Varchar[30]</i> : Nome dell'utente Cognome <i>Varchar[30]</i> : Cognome dell'utente Login <i>varchar[10]</i> : Username Password <i>Varchar[10]</i> : Password |
| Studente | Id_studente <i>Varchar[30]</i> : Matricola dello studente |
| Insegnante | Id_insegnante <i>Varchar[30]</i> : Codice identificativo dell'insegnante |
| Corso | Cod_corso <i>Varchar[10]</i> : Codice identificativo del corso Nome <i>Varchar[20]</i> : Nome del corso Data_inizio <i>Date</i> : Data di inizio del corso Data_fine <i>Date</i> : Data di fine del corso id_ins <i>Varchar[30]</i> : chiave esterna che collega il corso all'insegnante |
| Test | Nome_id <i>Varchar[30]</i> : Identificativo del test N_quiz <i>Int</i> : Numero di quiz che costituiscono il test Data <i>Date</i> : Data di svolgimento del test Orario_inizio <i>Time</i> : Orario di inizio per lo svolgimento del test Orario_fine <i>Date</i> : Orario di consegna del test cod_corso <i>Varchar[30]*</i> : codice del corso alla fine del quale si svolge il test |

| Nome | Descrizione |
|----------------------|---|
| Quiz_Risp_ape | Id_quiz <i>Varchar[10]</i> : Identificativo di un quiz a risposta aperta Domanda <i>Varchar[100]</i> : Testo della domanda di un quiz a risposta aperta Max_lunghezza <i>Int</i> : Lunghezza massima della risposta espressa in intero (esempio 100= risposta lunga 100 caratteri) Punt_max <i>Real</i> : Il punteggio che verrà assegnato se l'insegnante ritiene che la risposta sia completamente corretta Punt_min <i>Real</i> : Il punteggio minimo che può essere assegnato in caso di risposta errata o parzialmente corretta |
| quiz_Risp_Mul | Id_quiz <i>Varchar[10]</i> : Identificativo di un quiz a risposta multipla Domanda <i>Varchar[100]</i> : Testo della domanda di un quiz a risposta multipla Risposta A <i>Varchar [20]</i> : Testo della risposta A Risposta B <i>Varchar [20]</i> : Testo della risposta B Risposta C <i>Varchar [20]</i> : Testo della risposta C Risposta D <i>Varchar [20]</i> : Testo della risposta D Risposta_corretta <i>Varchar[20]</i> : Quale delle alternative è effettivamente la risposta corretta Punt_corretto <i>Int</i> : Il punteggio che verrà assegnato dal sistema se la risposta registrata dallo studente corrisponde a quella corretta Punt_errato <i>Int</i> : Il punteggio che verrà assegnato dal sistema se la risposta registrata dallo studente non corrisponde a quella corretta (esempio 0 o un punteggio negativo) |
| Test_svolto | nome_id <i>Varchar[10]</i> : Identificativo di un test Id_stud <i>Varchar[10]</i> : Identificativo di uno studente Id_quizM <i>Varchar[10]</i> : Identificativo di un quiz a risposta multipla Id_quizA <i>Varchar(10)</i> : Identificativo di un quiz a risposta aperta Risposta_data <i>Varchar[200]</i> : Risposta data dallo studente Punteggio_quiz_dato <i>Real</i> : Il punteggio ottenuto dallo studente in base alla correttezza della risposta N.B. se Id_quizM è NOT NULL Id_quizA è NULL e viceversa |

3.2 Dizionario delle associazioni

| Nome | Descrizione |
|----------------------|---|
| Frequenta | Uno studente segue uno o più corsi, un corso è seguito da uno o più studenti |
| Insegnamento | Un insegnante può tenere 1 o più corsi, un corso è tenuto da un solo insegnante |
| Conclusione | Al termine di ogni corso si tiene un solo esame finale |
| Consegna | Uno o più studenti consegnano 0 (test non consegnato) o più test (di diversi corsi) |
| ComposizioneA | Una o più risposte aperte possono essere in uno o più test, uno o più test possono essere composti da una o più risposte aperte (per esempio un insegnante prende una risposta aperta di una sessione precedente) |
| ComposizioneM | Una o più risposte multiple possono essere in uno o più test, uno o più test possono essere composti da una o più risposte multiple (per esempio un insegnante prende una risposta multipla di una sessione precedente) |

3.3 Dizionario dei vincoli

Tabella con gli altri vincoli esclusi chiavi primarie e foreign key, sono già stati definiti.

| Nome | Descrizione |
|---------------------------------|---|
| Risp | La risposta corretta ad un quiz a risposta multipla deve essere tra le opzioni date (A-B-C-D) |
| Max_lenght | La risposta ad un quiz a risposta aperta deve essere di almeno 50 caratteri |
| Pass | Una password ha lunghezza massima di 10 caratteri |
| Login_stud | Non possono esistere due studenti con lo stesso login |
| Login_ins | Non possono esistere due insegnanti con lo stesso login |
| Correttezza_data_fine | La data di fine di un corso deve essere successiva alla data di inizio |
| Numero_quiz | Ogni test deve avere almeno un quiz |
| Correttezza_orario_fine | L'orario di fine di un test deve essere successivo a quello di inizio |
| Unica_risposta_correttaM | La risposta corretta ad un quiz a risposta multipla è una sola quindi lo studente deve scegliere una sola opzione |
| Unica_risposta_correttaA | Lo studente non può dare due risposte ad un quiz a risposta aperta |
| Quiz | Nella tabella test svolto se Id_quizA è NULL allora Id_quizM è NOT NULL e viceversa |
| Test_quiz | Nella tabella test svolto non può essere inserita una risposta se il quiz non appartiene effettivamente al test |

Capitolo 4

PROGETTAZIONE LOGICA

4.1 Schema logico

| | |
|-----------------------|---|
| STUDENTE | (<u>id_studente</u> , nome, cognome, login, password) |
| INSEGNANTE | (<u>id_insegnante</u> , nome, cognome, login, password) |
| CORSO | (<u>cod_corso</u> , nome_corso, id_ins*, data_inizio, data_fine) |
| TEST | (<u>nome_id</u> , n_quiz, data, orario_inizio, orario_fine, cod_corso*) |
| QUIZ_RISP_MUL | (<u>id_quiz</u> , domanda, rispostaA, rispostaB, rispostaC,rispostaD, risposta_corretta, punt_corretto, punt_errato) |
| QUIZ_RISP_APE | (<u>id_quiz</u> , domanda, max_lunghezza_risp, punt_max, punt_min) |
| TEST_SVOLTO | (nome_id*, id_stud*, id_quizM*,id_quizA*, risposta_data, punteggio_quiz_dato) |
| FREQUENTA | (id_stud*, cod_corso*) |
| COMPOSIZIONE A | (nome_id*, id_quizA*) |
| COMPOSIZIONE M | (nome_id*, id_quizM*) |

Capitolo 5

PROGETTAZIONE FISICA

5.1 Domini

5.1.1 RISPOSTA_ESISTENTE

```
1 CREATE DOMAIN risp AS VARCHAR (20)
2 CHECK (VALUE = 'A' OR VALUE = 'B' OR VALUE = 'C' OR VALUE = 'D');
```

5.1.2 LUNGHEZZA_MASSIMA

```
1 CREATE DOMAIN max_lenght AS INT
2 CHECK (VALUE > 50);
```

5.1.3 PASSWORD

```
1 CREATE DOMAIN pass AS VARCHAR (30)
2 CHECK (VALUE ~ '^(?=[a-zA-Z])(?=[0-9])(?=.*[!@#$%^&*])*$'
3 AND VALUE LIKE '_____%');
```

5.2 Definizione delle tabelle

5.2.1 STUDENTE

```
1 CREATE TABLE STUDENTE (  
2     Id_stud varchar (30) NOT NULL,  
3     Nome varchar (30) NOT NULL,  
4     Cognome varchar (30) NOT NULL,  
5     Login varchar (10) NOT NULL,  
6     Password pass NOT NULL  
7 );  
8 ALTER TABLE STUDENTE  
9 ADD CONSTRAINT studente_pk PRIMARY KEY (Id_stud);
```

5.2.2 INSEGNANTE

```
1 CREATE TABLE INSEGNANTE(  
2     Id_ins varchar(30) NOT NULL,  
3     Nome varchar(30) NOT NULL,  
4     Cognome varchar(30) NOT NULL,  
5     Login varchar(10) NOT NULL,  
6     Password pass NOT NULL  
7 );  
8 ALTER TABLE INSEGNANTE  
9 ADD CONSTRAINT insegnante_pk PRIMARY KEY (Id_ins);
```

5.2.3 CORSO

```
1 CREATE TABLE CORSO(  
2     Cod_corso varchar(10) NOT NULL,  
3     Nome varchar(20) NOT NULL,  
4     Id_ins varchar (30) NOT NULL,  
5     Data_inizio date NOT NULL,  
6     Data_fine date NOT NULL  
7 );  
8 ALTER TABLE CORSO  
9 ADD CONSTRAINT corso_pk PRIMARY KEY (Cod_corso),  
10 ADD CONSTRAINT corso_fkb FOREIGN KEY (Id_ins)  
11 REFERENCES INSEGNANTE (Id_ins)  
12 ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.4 TEST

```
1 CREATE TABLE TEST(  
2     Nome_id varchar(30) NOT NULL,  
3     N_quiz int NOT NULL,  
4     Data date NOT NULL,  
5     Orario_inizio time NOT NULL,  
6     Orario_fine time NOT NULL,  
7     Cod_corso varchar(10) NOT NULL  
8 );  
9 ALTER TABLE TEST  
10 ADD CONSTRAINT test_pk PRIMARY KEY (Nome_id),  
11 ADD CONSTRAINT test_fk FOREIGN KEY (Cod_corso) REFERENCES CORSO (Cod_corso  
    );
```

5.2.5 QUIZ_RISP_MUL

```
1 CREATE TABLE QUIZ_RISP_MUL(  
2     Id_quiz varchar(10) NOT NULL,  
3     Domanda varchar(200) NOT NULL,  
4     A varchar(100) NOT NULL,  
5     B varchar(100) NOT NULL,  
6     C varchar(100) NOT NULL,  
7     D varchar(100) NOT NULL,  
8     Risposta_c risp NOT NULL,  
9     Punt_c int NOT NULL,  
10    Punt_e int NOT NULL  
11 );  
12 ALTER TABLE QUIZ_RISP_MUL  
13 ADD CONSTRAINT quiz_risp_mul_pk PRIMARY KEY (Id_quiz);
```

5.2.6 QUIZ_RISP_APE

```
1 CREATE TABLE QUIZ_RISP_APE(  
2     Id_quiz varchar (10) NOT NULL,  
3     Domanda varchar (200) NOT NULL,  
4     Lenght_risp max_lenght NOT NULL,  
5     Punt_max real NOT NULL,  
6     Punt_min real NOT NULL  
7 );  
8 ALTER TABLE QUIZ_RISP_APE  
9 ADD CONSTRAINT quiz_risp_ape_pk PRIMARY KEY (Id_quiz);
```

5.2.7 TEST_SVOLTO

```
1 CREATE TABLE TEST_SVOLTO(  
2     Nome_id varchar(30) NOT NULL,  
3     Id_stud varchar(30) NOT NULL,  
4     Id_quizM varchar(10),  
5     Id_quizA varchar(10),  
6     Risposta_data varchar(200) NOT NULL,  
7     Punteggio_quiz_dato real  
8 );  
9 ALTER TABLE TEST_SVOLTO  
10 ADD CONSTRAINT test_svolto_fka FOREIGN KEY(id_stud)  
11 REFERENCES STUDENTE(id_stud)  
12 ON UPDATE CASCADE ON DELETE RESTRICT,  
13 ADD CONSTRAINT test_svolto_fkb FOREIGN KEY (Nome_id)  
14 REFERENCES TEST (Nome_id)  
15 ON UPDATE CASCADE ON DELETE RESTRICT,  
16 ADD CONSTRAINT test_svolto_fkc FOREIGN KEY(id_quizA)  
17 REFERENCES QUIZ_RISP_APE(id_quiz)  
18 ON UPDATE CASCADE ON DELETE RESTRICT,  
19 ADD CONSTRAINT test_svolto_fkd FOREIGN KEY(id_quizM)  
20 REFERENCES QUIZ_RISP_MUL(id_quiz)  
21 ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.8 FREQUENTA

```
1 CREATE TABLE FREQUENTA(  
2     Id_stud varchar(30) NOT NULL,  
3     Cod_corso varchar(10) NOT NULL  
4 );  
5 ALTER TABLE FREQUENTA  
6 ADD CONSTRAINT frequenta_fka FOREIGN KEY(Id_stud)  
7 REFERENCES STUDENTE (Id_stud)  
8 ON UPDATE CASCADE ON DELETE RESTRICT,  
9 ADD CONSTRAINT frequenta_fkb FOREIGN KEY(Cod_corso)  
10 REFERENCES CORSO (Cod_corso)  
11 ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.9 COMPOSIZIONE

```
1 CREATE TABLE COMPOSIZIONE(  
2     Id_quizA varchar (10) NOT NULL,  
3     Nome_id varchar (30) NOT NULL  
4 );
```



```

5 ALTER TABLE COMPOSIZIONE
6 ADD CONSTRAINT compa_fka FOREIGN KEY(Id_quizA)
7 REFERENCES QUIZ_RISP_APE (Id_quiz)
8 ON UPDATE CASCADE ON DELETE RESTRICT,
9 ADD CONSTRAINT compa_fkb FOREIGN KEY(Nome_id)
10 REFERENCES TEST (Nome_id)
11 ON UPDATE CASCADE ON DELETE RESTRICT;

```

5.2.10 COMPOSIZIONEM

```

1 CREATE TABLE COMPOSIZIONEM(
2     Id_quizM varchar(10) NOT NULL,
3     Nome_id varchar(30) NOT NULL
4 );
5 ALTER TABLE COMPOSIZIONEM
6 ADD CONSTRAINT compm_fka FOREIGN KEY(Id_quizM)
7 REFERENCES QUIZ_RISP_MUL (Id_quiz)
8 ON UPDATE CASCADE ON DELETE RESTRICT,
9 ADD CONSTRAINT compm_fkb FOREIGN KEY(Nome_id)
10 REFERENCES TEST (Nome_id)
11 ON UPDATE CASCADE ON DELETE RESTRICT;

```

5.3 Implementazione dei vincoli

5.3.1 LOGIN_STUD

```
1 ALTER TABLE STUDENTE
2 ADD CONSTRAINT login_stud
3 UNIQUE(login);
```

5.3.2 LOGIN_INS

```
1 ALTER TABLE INSEGNANTE
2 ADD CONSTRAINT login_ins
3 UNIQUE(login);
```

5.3.3 CORRETTEZZA_DATA_FINE

```
1 ALTER TABLE CORSO
2 ADD CONSTRAINT correttezza_data_fine
3 CHECK (Data_fine > Data_inizio);
```

5.3.4 NUMERO_QUIZ

```
1 ALTER TABLE TEST
2 ADD CONSTRAINT numero_quiz
3 CHECK (n_quiz >= 1);
```

5.3.5 CORRETTEZZA_ORARIO_FINE

```
1 ALTER TABLE TEST
2 ADD CONSTRAINT correttezza_orario_fine
3 CHECK (orario_fine > orario_inizio);
```

5.3.6 UNICA_RISPOSTA_CORRETTAM

```
1 ALTER TABLE TEST_SVOLTO
2 ADD CONSTRAINT unica_risposta_correttaM
3 UNIQUE (Id_stud, Id_quizM, Risposta_data);
```

5.3.7 UNICA_RISPOSTA_CORRETTAA

```
1 ALTER TABLE TEST_SVOLTO
2 ADD CONSTRAINT unica_risposta_correttaA
3 UNIQUE (Id_stud, Id_quizA, Risposta_data);
```

5.3.8 QUIZ

```
1 ALTER TABLE TEST_SVOLTO
2 ADD CONSTRAINT unica_risposta_correttaA
3 UNIQUE (Id_stud, Id_quizA, Risposta_data);
```

5.4 Funzioni, Procedure ed altre automazioni

5.4.1 PUNT_TOT

```
1 create or replace FUNCTION punt_tot(  
2     test TEST.Nome_id%TYPE,  
3     studente STUDENTE.id_stud%TYPE)  
4 returns text as $$  
5 declare  
6 somma real;  
7 begin  
8 SELECT SUM(Punteggio_quiz_dato) INTO somma  
9 FROM TEST_SVOLTO  
10 WHERE Nome_id=test AND id_stud=studente;  
11  
12 return studente||'-'>|| test||'-'>||somma;  
13 end; $$ language plpgsql
```

5.4.2 DATA_QUIZ

```
1 CREATE OR REPLACE FUNCTION data_quiz () RETURNS TRIGGER AS $Dataq$
2 DECLARE
3 data_fine_corso CORSO.Data_fine % TYPE;
4 BEGIN
5
6 SELECT Data_fine INTO data_fine_corso
7 FROM CORSO
8 WHERE Cod_corso=NEW.Cod_corso;
9
10 IF NEW.Data > data_fine_corso THEN
11 RETURN NEW;
12
13 ELSE
14 RAISE NOTICE 'Il test non si pu svolgere prima della fine del corso';
15 DELETE
16 FROM TEST
17 WHERE Nome_id=NEW.Nome_id;
18
19 END IF;
20
21 RETURN NEW;
22 END; $Dataq$ LANGUAGE plpgsql;
23
24 CREATE OR REPLACE TRIGGER Dataq
25 AFTER INSERT ON TEST
26 FOR EACH ROW
27 EXECUTE FUNCTION data_quiz ();
```

5.4.3 INS_LOG

```
1 CREATE OR REPLACE FUNCTION ins_log () RETURNS TRIGGER AS $log_ins$
2 DECLARE
3 stud CURSOR IS(
4 SELECT login
5 FROM Studente);
6 BEGIN
7
8 FOR el IN stud
9 LOOP
10 IF NEW.Login=el.login THEN
11 RAISE NOTICE 'Questo login      gi      esistente';
12 DELETE
13 FROM INSEGNANTE
14 WHERE Id_ins=NEW.Id_ins;
15
16 ELSE
17 END IF;
18 END LOOP;
19
20 RETURN NEW;
21 END; $log_ins$ LANGUAGE plpgsql;
22
23 CREATE OR REPLACE TRIGGER log_ins
24 AFTER INSERT ON INSEGNANTE
25 FOR EACH ROW
26 EXECUTE FUNCTION ins_log ();
```

5.4.4 STUD_LOG

```
1 CREATE OR REPLACE FUNCTION stud_log () RETURNS TRIGGER AS $log_stud$
2 DECLARE
3 ins CURSOR IS(
4 SELECT login
5 FROM INSEGNANTE);
6 BEGIN
7
8 FOR el IN ins
9 LOOP
10 IF NEW.login=el.login THEN
11 RAISE NOTICE 'Questo login      gi      esistente';
12 DELETE
13 FROM STUDENTE
14 WHERE Id_stud=NEW.Id_stud;
15
16 ELSE
17 END IF;
18 END LOOP;
19
20 RETURN NEW;
21 END; $log_stud$ LANGUAGE plpgsql;
22
23 CREATE OR REPLACE TRIGGER log_stud
24 AFTER INSERT ON STUDENTE
25 FOR EACH ROW
26 EXECUTE FUNCTION stud_log ();
```

5.4.5 PUNTEGGIO_ASSEGNATO

```
1 CREATE OR REPLACE FUNCTION punteggio_assegnato () RETURNS TRIGGER AS
    $punteggio$
2 DECLARE
3 punteggio_max QUIZ_RISP_APE.punt_max% TYPE;
4 punteggio_min QUIZ_RISP_APE.punt_min% TYPE;
5 BEGIN
6
7 SELECT punt_max INTO punteggio_max
8 FROM QUIZ_RISP_APE
9 WHERE Id_quiz=NEW.Id_quizA;
10
11 SELECT punt_min INTO punteggio_min
12 FROM QUIZ_RISP_APE
13 WHERE Id_quiz=NEW.Id_quizA;
14
15
16 IF NEW.Id_quizA IS NOT NULL AND (NEW.Punteggio_quiz_dato BETWEEN
    punteggio_min AND punteggio_max) THEN
17 RETURN NEW;
18 END IF;
19
20 IF NEW.Id_quizA IS NOT NULL AND (NEW.Punteggio_quiz_dato NOT BETWEEN
    punteggio_min AND punteggio_max) THEN
21 DELETE
22 FROM TEST_SVOLTO
23 WHERE Nome_id=NEW.Nome_id;
24 RAISE NOTICE 'ERRORE, il punteggio non rientra nel range consentito';
25 END IF;
26
27 RETURN NEW;
28 END; $punteggio$ LANGUAGE plpgsql;
29
30 CREATE OR REPLACE TRIGGER punteggio
31 AFTER INSERT ON TEST_SVOLTO
32 FOR EACH ROW
33 EXECUTE FUNCTION punteggio_assegnato ();
```


5.4.6 AUTO_CORREZIONE

```
1 CREATE OR REPLACE FUNCTION Auto_correzione () RETURNS TRIGGER AS
    $Correzione$
2 DECLARE
3 risp QUIZ_RISP_MUL.Risposta_c % TYPE;
4 BEGIN
5
6 IF NEW.Id_quizA IS NULL THEN
7
8 SELECT Risposta_c INTO risp
9 FROM QUIZ_RISP_MUL
10 WHERE Id_quiz=NEW.Id_quizM;
11
12 IF NEW.Risposta_data=risp THEN
13 UPDATE TEST_SVOLTO
14 SET Punteggio_quiz_dato =(SELECT punt_c
15                             FROM QUIZ_RISP_MUL
16                             WHERE Id_quiz= NEW.Id_quizM)
17 WHERE Nome_id=NEW.Nome_id AND Id_quizM=NEW.Id_quizM AND Id_Stud=NEW.
    Id_stud;
18
19 ELSE
20
21 UPDATE TEST_SVOLTO
22 SET Punteggio_quiz_dato =(SELECT punt_e
23                             FROM QUIZ_RISP_MUL
24                             WHERE Id_quiz= NEW.Id_quizM)
25 WHERE Nome_id=NEW.Nome_id AND Id_quizM=NEW.Id_quizM AND Id_Stud=NEW.
    Id_stud;
26
27 END IF;
28
29 END IF;
30 RETURN NEW ;
31 END; $Correzione$ LANGUAGE plpgsql;
32
33 CREATE OR REPLACE TRIGGER Correzione
34 AFTER INSERT ON TEST_SVOLTO
35 FOR EACH ROW
36 EXECUTE FUNCTION Auto_Correzione ();
```

5.4.7 CORREZIONE_RISP_APE

```
1 create or replace PROCEDURE correzione_risp_ape(  
2     test TEST.Nome_id%TYPE,  
3     studente STUDENTE.id_stud%TYPE,  
4     quiz TEST_SVOLTO.id_quizA%TYPE,  
5     voto TEST_SVOLTO.punteggio_quiz_dato%TYPE)  
6 as $$  
7 declare  
8  
9 begin  
10 UPDATE TEST_SVOLTO  
11 SET Punteggio_quiz_dato=voto  
12 WHERE Nome_id=test AND Id_quizA=quiz AND id_stud=studente;
```

5.4.8 TEST_QUIZ

```
1 CREATE OR REPLACE FUNCTION quiz_test () RETURNS TRIGGER AS $test_quiz$
2 DECLARE
3
4 BEGIN
5
6 IF NEW.Id_quizA IS NOT NULL THEN
7
8 IF (NEW.Id_quizA NOT IN (SELECT Id_quizA
9     FROM COMPOSIZIONE_A
10    WHERE Nome_id=NEW.Nome_id) ) THEN
11 RAISE NOTICE 'ERRORE,il quiz non appartiene al test';
12 DELETE
13 FROM TEST_SVOLTO
14 WHERE Nome_id=NEW.Nome_id AND Id_quizA=NEW.Id_quizA AND Id_stud=NEW.
    Id_stud;
15 END IF;
16
17 ELSE
18     IF( NEW.Id_quizM NOT IN (SELECT Id_quizM
19         FROM COMPOSIZIONE_M
20        WHERE Nome_id=NEW.Nome_id) ) THEN
21 RAISE NOTICE 'ERRORE,il quiz non appartiene al test';
22 DELETE
23 FROM TEST_SVOLTO
24 WHERE Nome_id=NEW.Nome_id AND Id_quizM=NEW.Id_quizM AND Id_stud=NEW.
    Id_stud;
25 END IF;
26 END IF;
27
28 RETURN NEW ;
29 END; $test_quiz$ LANGUAGE plpgsql;
30
31 CREATE TRIGGER test_quiz
32 AFTER INSERT ON TEST_SVOLTO
33 FOR EACH ROW
34 EXECUTE FUNCTION quiz_test ();
```