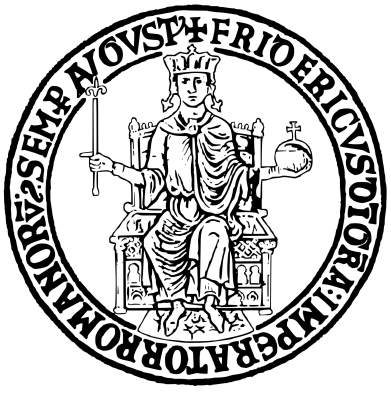
UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

BASI DI DATI

Progettazione e svilluppo di una base di dati relazionale per una applicazione di e-learning

Marzia PIROZZI N86003545
Noemi SPERA N86003717

Gennaio 2022



Indice

Capitolo 1

DESCRIZIONE DEL PROGETTO

1.1 Introduzione

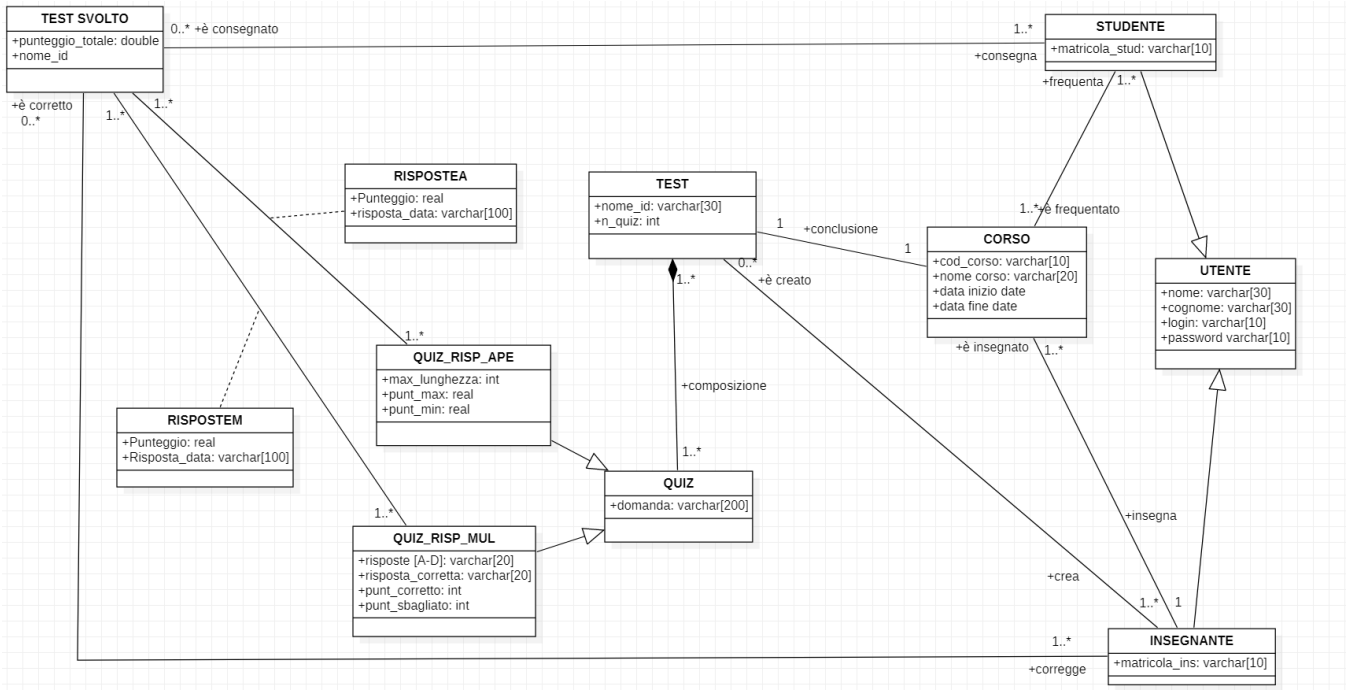
Abbiamo creato una base di dati relazionale in cui **studenti** e **insegnanti** si iscrivono ad una piattaforma di e_learning tramite un **login** e una **password**. Gli insegnanti caricano dei **test**, composti da **quiz**, e gli studenti possono svolgerli (se effettivamente seguono il **corso** 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 dagll'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.

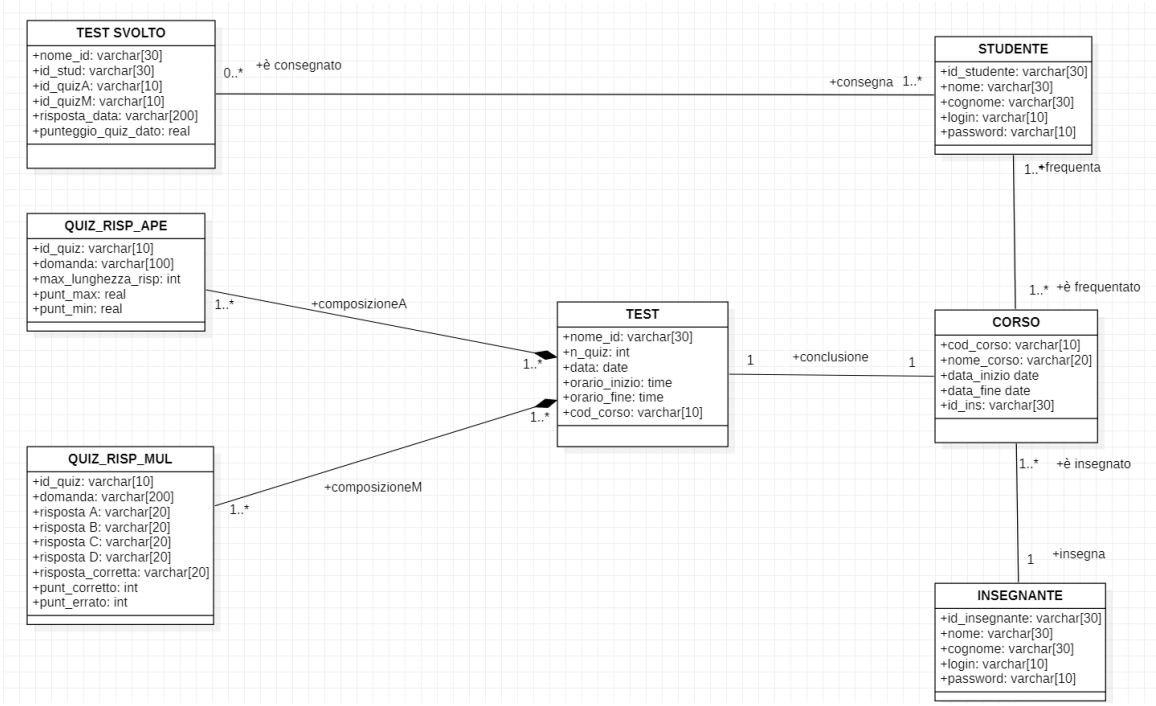


2.2 Descrizione Class Diagram

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 insegnate, 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 verrano risolte con la ristrutturazione.

2.3 Class Diagram ristrutturato

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



2.3.1 Analisi delle ridondanze

Sono state notate due ridondanze. La prima la riscontriamo tra test svolto, ripostaA 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 insegnate 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** *Varchar[30]*: Nome dell'utente **Cognome** *Varchar[30]*: Cognome dell'utente **Login** *varchar[10]*: Username **Password** *Varchar[10]*: Password
Studente	**Id_studente** *Varchar[30]*: Matricola dello studente
Insegnante	**Id_insegnante** *Varchar[30]*: Codice identificativo dell'insegnante
Corso	**Cod_corso** *Varchar[10]*: Codice identificativo del corso **Nome** *Varchar[20]*: Nome del corso **Data_inizio** *Date*: Data di inizio del corso **Data_fine** *Date*: Data di fine del corso **id_ins** *Varchar[30]*: chiave esterna che collega il corso all'insegnante
Test	**Nome_id** *Varchar[30]*: Identificativo del test **N_quiz** *Int*: Numero di quiz che costituiscono il test **Data** *Date*: Data di svolgimento del test **Orario_inizio** *Time*: Orario di inzio per lo svolgimento del test **Orario_fine** *Date*: Orario di consegna del test **cod_corso** *Varchar[30**:codice del corso alla fine del quale si svolge il test

Nome	Descrizione
Quiz_Risp_ape	**Id_quiz** *Varchar[10]*: Identificativo di un quiz a risposta aperta **Domanda** *Varchar[100]*: Testo della domanda di un quiz a risposta aperta **Max_lunghezza** *Int*: Lunghezza massima della risposta espressa in intero (esempio 100= risposta lunga 100 caratteri) **Punt_max** *Real*: Il punteggio che verrà assegnato se l'insegnante ritiene che la risposta sia completamente corretta **Punt_min** *Real*: Il punteggio minimo che può essere assegnato in caso di risposta errata o parzialmente corretta
quiz_Risp_Mul	**Id_quiz** *Varchar[10]*: Identificativo di un quiz a risposta multipla **Domanda** *Varchar[100]*: Testo della domanda di un quiz a risposta multipla **Risposta A** *Varchar [20]*: Testo della risposta A **Risposta B** *Varchar [20]*: Testo della risposta B **Risposta C** *Varchar [20]*: Testo della risposta C **Risposta D** *Varchar [20]*: Testo della risposta D **Risposta_corretta** *Varchar[20]*: Quale delle alternative è effettivamente la risposta corretta **Punt_corretto** *Int*: Il punteggio che verrà assegnato dal sistema se la risposta registrata dallo studente corrisposde a quella corretta **Punt_errato** *Int*: Il punteggio che verrà assegnato dal sistema se la risposta registrata dallo studente non corrisposde a quella corretta (esempio 0 o un punteggio negativo)
Test_svolto	**nome_id** *Varchar[10]*: Identificativo di un test **Id_stud** *Varchar[10]*: Identificativo di uno studente **Id_quizM** *Varchar[10]*: Identificativo di un quiz a risposta multipla **Id_quizA** *Varchar(10)*: Identificativo di un quiz a risposta aperta **Risposta_data** *Varchar[200]*: Risposta data dallo studente **Punteggio_quiz_dato** *Real*: 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 inegnante
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 inzio
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	(id_studente, nome, cognome, login, password)
INSEGNANTE	(id_insegnante, nome, cognome, login, password)
CORSO	(cod_corso, nome_corso, id_ins*, data_inizio, data_fine)
TEST	(nome_id, n_quiz, data, orario_inizio, orario_fine, cod_corso*)
QUIZ_RISP_MUL	(id_quiz, domanda, rispostaA, rispostaB, rispostaC,rispostaD, risposta_corretta, punt_corretto, punt_errato)
QUIZ_RISP_APE	(id_quiz, 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*)
COMPOSIZIONEA	(nome_id*, id_quizA*)
COMPOSIZIONEM	(nome_id*, id_quizM*)

Capitolo 5

PROGETTAZIONE FISICA

5.1 Domini

5.1.1 RISPOSTA_ESISTENTE

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

5.1.2 LUNGHEZZA_MASSIMA

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

5.1.3 LUNGHEZZA_PASSWORD

```
CREATE DOMAIN pass AS VARCHAR (10);
```

5.2 Definizione delle tabelle

5.2.1 STUDENTE

```
CREATE TABLE STUDENTE (
    Id_stud varchar (30) NOT NULL,
    Nome varchar (30) NOT NULL,
    Cognome varchar (30) NOT NULL,
    Login varchar (10) NOT NULL,
    Password pass NOT NULL
);
ALTER TABLE STUDENTE
ADD CONSTRAINT studente_pk PRIMARY KEY (Id_stud);
```

5.2.2 INSEGNANTE

```
CREATE TABLE INSEGNANTE(
    Id_ins varchar(30) NOT NULL,
    Nome varchar(30) NOT NULL,
    Cognome varchar(30) NOT NULL,
    Login varchar(10) NOT NULL,
    Password pass NOT NULL
);
ALTER TABLE INSEGNANTE
ADD CONSTRAINT insegnante_pk PRIMARY KEY (Id_ins);
```

5.2.3 CORSO

```
CREATE TABLE CORSO(
    Cod_corso varchar(10) NOT NULL,
    Nome varchar(20) NOT NULL,
    Id_ins varchar (30) NOT NULL,
    Data_inizio date NOT NULL,
    Data_fine date NOT NULL
);
ALTER TABLE CORSO
ADD CONSTRAINT corso_pk PRIMARY KEY (Cod_corso),
ADD CONSTRAINT corso_fkb FOREIGN KEY (Id_ins)
REFERENCES INSEGNANTE (Id_ins)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.4 TEST

```
CREATE TABLE TEST(
    Nome_id varchar(30) NOT NULL,
    N_quiz int NOT NULL,
    Data date NOT NULL,
    Orario_inizio time NOT NULL,
    Orario_fine time NOT NULL,
    Cod_corso varchar(10) NOT NULL
);
ALTER TABLE TEST
ADD CONSTRAINT test_pk PRIMARY KEY (Nome_id),
ADD CONSTRAINT test_fk FOREIGN KEY (Cod_corso) REFERENCES CORSO (Cod_corso
   );
```

5.2.5 QUIZ_RISP_MUL

```
CREATE TABLE QUIZ_RISP_MUL(
    Id_quiz varchar(10) NOT NULL,
    Domanda varchar(200)  NOT NULL,
    A varchar(100) NOT NULL,
    B varchar(100) NOT NULL,
    C varchar(100) NOT NULL,
    D varchar(100) NOT NULL,
    Risposta_c risp NOT NULL,
    Punt_c int NOT NULL,
    Punt_e int NOT NULL
);
ALTER TABLE QUIZ_RISP_MUL
ADD CONSTRAINT quiz_risp_mul_pk PRIMARY KEY (Id_quiz);
```

5.2.6 QUIZ_RISP_APE

```
CREATE TABLE QUIZ_RISP_APE(
    Id_quiz varchar (10) NOT NULL,
    Domanda varchar (200) NOT NULL,
    Lenght_risp max_lenght NOT NULL,
    Punt_max real NOT NULL,
    Punt_min real NOT NULL
);
ALTER TABLE QUIZ_RISP_APE
ADD CONSTRAINT quiz_risp_ape_pk PRIMARY KEY (Id_quiz);
```

5.2.7 TEST_SVOLTO

```
CREATE TABLE TEST_SVOLTO(
    Nome_id varchar(30) NOT NULL,
    Id_stud varchar(30) NOT NULL,
    Id_quizM varchar(10),
    Id_quizA varchar(10),
    Risposta_data varchar(200) NOT NULL,
    Punteggio_quiz_dato real
);
ALTER TABLE TEST_SVOLTO
ADD CONSTRAINT test_svolto_fka FOREIGN KEY(id_stud)
REFERENCES STUDENTE(id_stud)
ON UPDATE CASCADE ON DELETE RESTRICT,
ADD CONSTRAINT test_svolto_fkb FOREIGN KEY (Nome_id)
REFERENCES TEST (Nome_id)
ON UPDATE CASCADE ON DELETE RESTRICT,
ADD CONSTRAINT test_svolto_fkc FOREIGN KEY(id_quizA)
REFERENCES QUIZ_RISP_APE(id_quiz)
ON UPDATE CASCADE ON DELETE RESTRICT,
ADD CONSTRAINT test_svolto_fkd FOREIGN KEY(id_quizM)
REFERENCES QUIZ_RISP_MUL(id_quiz)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.8 FREQUENTA

```
CREATE TABLE FREQUENTA(
    Id_stud varchar(30) NOT NULL,
    Cod_corso varchar(10) NOT NULL
);
ALTER TABLE FREQUENTA
ADD CONSTRAINT frequenta_fka FOREIGN KEY(Id_stud)
REFERENCES STUDENTE (Id_stud)
ON UPDATE CASCADE ON DELETE RESTRICT,
ADD CONSTRAINT frequenta_fkb FOREIGN KEY(Cod_corso)
REFERENCES CORSO (Cod_corso)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.9 COMPOSIZIONEA

```
CREATE TABLE COMPOSIZIONEA(
    Id_quizA varchar (10) NOT NULL,
    Nome_id varchar (30) NOT NULL
);
```

```
ALTER TABLE COMPOSIZIONEA
ADD CONSTRAINT compa_fka FOREIGN KEY(Id_quizA)
REFERENCES QUIZ_RISP_APE (Id_quiz)
ON UPDATE CASCADE ON DELETE RESTRICT,
ADD CONSTRAINT compa_fkb FOREIGN KEY(Nome_id)
REFERENCES TEST (Nome_id)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2.10 COMPOSIZIONEM

```
CREATE TABLE COMPOSIZIONEM(
    Id_quizM varchar(10) NOT NULL,
    Nome_id varchar(30) NOT NULL
);
ALTER TABLE COMPOSIZIONEM
ADD CONSTRAINT compm_fka FOREIGN KEY(Id_quizM)
REFERENCES QUIZ_RISP_MUL (Id_quiz)
ON UPDATE CASCADE ON DELETE RESTRICT,
ADD CONSTRAINT compm_fkb FOREIGN KEY(Nome_id)
REFERENCES TEST (Nome_id)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.3 Implementazione dei vincoli

5.3.1 LOGIN_STUD

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

5.3.2 LOGIN_INS

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

5.3.3 CORRETTEZZA_DATA_FINE

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

5.3.4 NUMERO_QUIZ

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

5.3.5 CORRETTEZZA_ORARIO_FINE

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

5.3.6 UNICA_RISPOSTA_CORRETTAM

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

5.3.7 UNICA_RISPOSTA_CORRETTAA

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

5.3.8 QUIZ

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

5.4 Funzioni,Procedure ed altre automazioni

5.4.1 PUNT_TOT

```
ALTER TABLE TEST_SVOLTO
ADD CONSTRAINT quiz
CHECK (Id_quizM IS NOT NULL AND Id_quizA IS NULL OR(Id_quizM IS NULL AND
   Id_quizA IS  NOT NULL));

create or replace FUNCTION punt_tot(
    test TEST.Nome_id%TYPE,
    studente STUDENTE.id_stud%TYPE)
returns text as $$
declare
somma real;
begin
SELECT SUM(Punteggio_quiz_dato) INTO somma
FROM TEST_SVOLTO
```

5.4.2 DATA_QUIZ

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

5.4.3 INS_LOG

```
AFTER INSERT ON TEST
FOR EACH ROW
EXECUTE FUNCTION data_quiz ();

CREATE OR REPLACE FUNCTION ins_log () RETURNS TRIGGER AS $log_ins$
DECLARE
stud CURSOR IS(
SELECT login
FROM Studente);
BEGIN

FOR el IN stud
LOOP
IF NEW.Login=el.login THEN
RAISE NOTICE 'Questo login    gi  esistente';
DELETE
FROM INSEGNANTE
WHERE Id_ins=NEW.Id_ins;

ELSE
END IF;
END LOOP;

RETURN NEW;
END; $log_ins$ LANGUAGE plpgsql;
```

5.4.4 STUD_LOG

```
AFTER INSERT ON INSEGNANTE
FOR EACH ROW
EXECUTE FUNCTION ins_log ();

CREATE OR REPLACE FUNCTION stud_log () RETURNS TRIGGER AS $log_stud$
DECLARE
ins CURSOR IS(
SELECT login
FROM INSEGNANTE);
BEGIN

FOR el IN ins
LOOP
IF NEW.login=el.login THEN
RAISE NOTICE 'Questo login    gi  esistente';
DELETE
FROM STUDENTE
WHERE Id_stud=NEW.Id_stud;

ELSE
END IF;
END LOOP;

RETURN NEW;
END; $log_stud$ LANGUAGE plpgsql;
```

5.4.5 PUNTEGGIO_ASSEGNATO

```
AFTER INSERT ON STUDENTE
FOR EACH ROW
EXECUTE FUNCTION stud_log ();

CREATE OR REPLACE FUNCTION punteggio_assegnato () RETURNS TRIGGER AS
   $punteggio$
DECLARE
punteggio_max QUIZ_RISP_APE.punt_max% TYPE;
punteggio_min QUIZ_RISP_APE.punt_min% TYPE;
BEGIN

SELECT punt_max INTO punteggio_max
FROM QUIZ_RISP_APE
WHERE Id_quiz=NEW.Id_quizA;

SELECT punt_min INTO punteggio_min
FROM QUIZ_RISP_APE
WHERE Id_quiz=NEW.Id_quizA;


IF NEW.Id_quizA IS NOT NULL AND (NEW.Punteggio_quiz_dato BETWEEN
   punteggio_min AND punteggio_max) THEN
RETURN NEW;
END IF;

IF NEW.Id_quizA IS NOT NULL AND (NEW.Punteggio_quiz_dato  NOT BETWEEN
   punteggio_min AND punteggio_max) THEN
DELETE
FROM TEST_SVOLTO
WHERE Nome_id=NEW.Nome_id;
RAISE NOTICE 'ERRORE, il punteggio non riesntra nel range consentito';
END IF;

RETURN NEW;
END; $punteggio$ LANGUAGE plpgsql;
```

5.4.6 AUTO_CORREZIONE

```
AFTER INSERT ON TEST_SVOLTO
FOR EACH ROW
EXECUTE FUNCTION punteggio_assegnato ();

CREATE OR REPLACE FUNCTION Auto_correzione () RETURNS TRIGGER AS
   $Correzione$
DECLARE
risp QUIZ_RISP_MUL.Risposta_c % TYPE;
BEGIN

IF NEW.Id_quizA IS NULL THEN

SELECT Risposta_c INTO risp
FROM QUIZ_RISP_MUL
WHERE Id_quiz=NEW.Id_quizM;

IF NEW.Risposta_data=risp THEN
UPDATE TEST_SVOLTO
SET Punteggio_quiz_dato =(SELECT punt_c
                          FROM QUIZ_RISP_MUL
                          WHERE Id_quiz= NEW.Id_quizM)
WHERE Nome_id=NEW.Nome_id AND Id_quizM=NEW.Id_quizM AND Id_Stud=NEW.
   Id_stud;

ELSE

UPDATE TEST_SVOLTO
SET Punteggio_quiz_dato =(SELECT punt_e
                          FROM QUIZ_RISP_MUL
                          WHERE Id_quiz= NEW.Id_quizM)
WHERE Nome_id=NEW.Nome_id AND Id_quizM=NEW.Id_quizM AND Id_Stud=NEW.
   Id_stud;

END IF;

END IF;
RETURN NEW ;
END; $Correzione$ LANGUAGE plpgsql;
```

5.4.7 CORREZIONE_RISP_APE

```
INSERT INTO TEST_SVOLTO VALUES('HSOW0384','N86002121',10,NULL,'B',NULL);
INSERT INTO TEST_SVOLTO VALUES('JAOD8479','N86002121',NULL,11, ' L IMPERO
    ROMANO  D OCCIDENTE     STATO ..',10);
INSERT INTO TEST_SVOLTO VALUES('FAG56455','N86006565',NULL,11, 'IFUYGFU'
   ,6);

create or replace PROCEDURE correzione_risp_ape(
    test TEST.Nome_id%TYPE,
    studente STUDENTE.id_stud%TYPE,
  quiz TEST_SVOLTO.id_quizA%TYPE,
  voto TEST_SVOLTO.punteggio_quiz_dato%TYPE)
as $$
declare

begin
UPDATE TEST_SVOLTO
```

5.4.8 TEST_QUIZ

```
WHERE Nome_id=test AND Id_quizA=quiz AND id_stud=studente;

end; $$ language plpgsql

CREATE OR REPLACE FUNCTION quiz_test () RETURNS TRIGGER AS $test_quiz$
DECLARE

BEGIN

IF NEW.Id_quizA IS NOT NULL THEN

IF (NEW.Id_quizA NOT IN (SELECT Id_quizA
    FROM COMPOSIZIONEA
    WHERE Nome_id=NEW.Nome_id) )THEN
RAISE NOTICE 'ERRORE,il quiz non appartiene al test';
DELETE
FROM TEST_SVOLTO
WHERE Nome_id=NEW.Nome_id AND Id_quizA=NEW.Id_quizA AND Id_stud=NEW.
   Id_stud;
END IF;

ELSE
    IF( NEW.Id_quizM NOT IN (SELECT Id_quizM
                        FROM COMPOSIZIONEM
                        WHERE Nome_id=NEW.Nome_id) )THEN
RAISE NOTICE 'ERRORE,il quiz non appartiene al test';
DELETE
FROM TEST_SVOLTO
WHERE Nome_id=NEW.Nome_id AND Id_quizM=NEW.Id_quizM AND Id_stud=NEW.
   Id_stud;
END IF;
END IF;

RETURN NEW ;
END; $test_quiz$ LANGUAGE plpgsql;

CREATE TRIGGER test_quiz
```