UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Progetto “Kvíz”

Applicazione E-learning tramite quiz

Traccia 2

Simeone Vitale N86003606

Anno Accademico 2022/2023

Docente: Silvio Barra

Sommario

Simeone Vitale
N86003606

Introduzione

Descrizione progetto

Kvíz è un applicativo e-learning capace di gestire test. Gli utenti interessati all'utilizzo dell'applicazione sono: gli studenti, che svolgono i test e visualizzano la loro correzione, e gli insegnanti che creano i test da somministrare agli studenti e li correggono.

Entrambe le categorie di utenti possono utilizzare l'applicativo previa registrazione con un username univoco e una password.

I test sono formati da una serie di quiz che sono distinti in due categorie:

- risposta 'multipla', le quali prevedono piu possibili risposte, di cui una sola è da ritenersi corretta,
- risposta 'aperta', alle quali si può rispondere con un testo apposito.

Ad entrambi è previsto un punteggio da assegnare in caso di risposta corretta o errata; le risposte 'aperte' sono destinate alla valutazione dell'insegnante che provvederà ad assegnare il relativo punteggio in base alla correttezza della risposta, invece per i quiz a risposta multipla sarà l'applicativo stesso in automatico ad assegnare il punteggio in base alla correttezza della risposta dello studente.

Simeone Vitale
N86003606

Requisiti identificati

Il requisito fondamentale è l'interazione tra gli utenti, i test e i quiz che varia in base alla tipologia di utente coinvolta:

- Gli insegnanti dovranno poter:
    - Creare i test con i relativi quiz associati
    - Correggere ed assegnare un punteggio ai quiz associati ai test svolti da ogni studente creati in precedenza
        - La valutazione avverrà in modo automatico se il quiz è a risposta multipla
        - La valutazione avverrà in modo manuale tramite l'insegnante creatore del test se il quiz è a risposta aperta
        - La valutazione deve tener conto di un range di valori numerici fissato dall'insegnante in fase di creazione del quiz

- Gli studenti dovranno poter:
    - Ricercare un test (creato dal docente di riferimento) tramite il suo nome univoco
    - Svolgere il test interessato
    - Rivedere il test dopo la correzione del docente

Entrambe le tipologie hanno in comune due operazioni che dovranno poter eseguire:

- Registrazione con
    - Nome
    - Cognome
    - Username
    - Password
- Login con
    - Username
    - Password

Dato che non sono previste limitazioni al numero di risposte in caso di quiz a risposta multipla, si è convenuto prevederne 4 (quattro).

Inoltre, non essendo previste limitazioni riguardo il numero massimo di quiz associabili ad ogni test si è convenuto prevederne al massimo 5 (cinque).

Dato che non è stato specificato alcun dominio per i voti minimi e massimi assegnabili ai quiz,

per i valori minimi (o risposta errata) sono permessi valori decimali negativi o uguali a 0(zero),

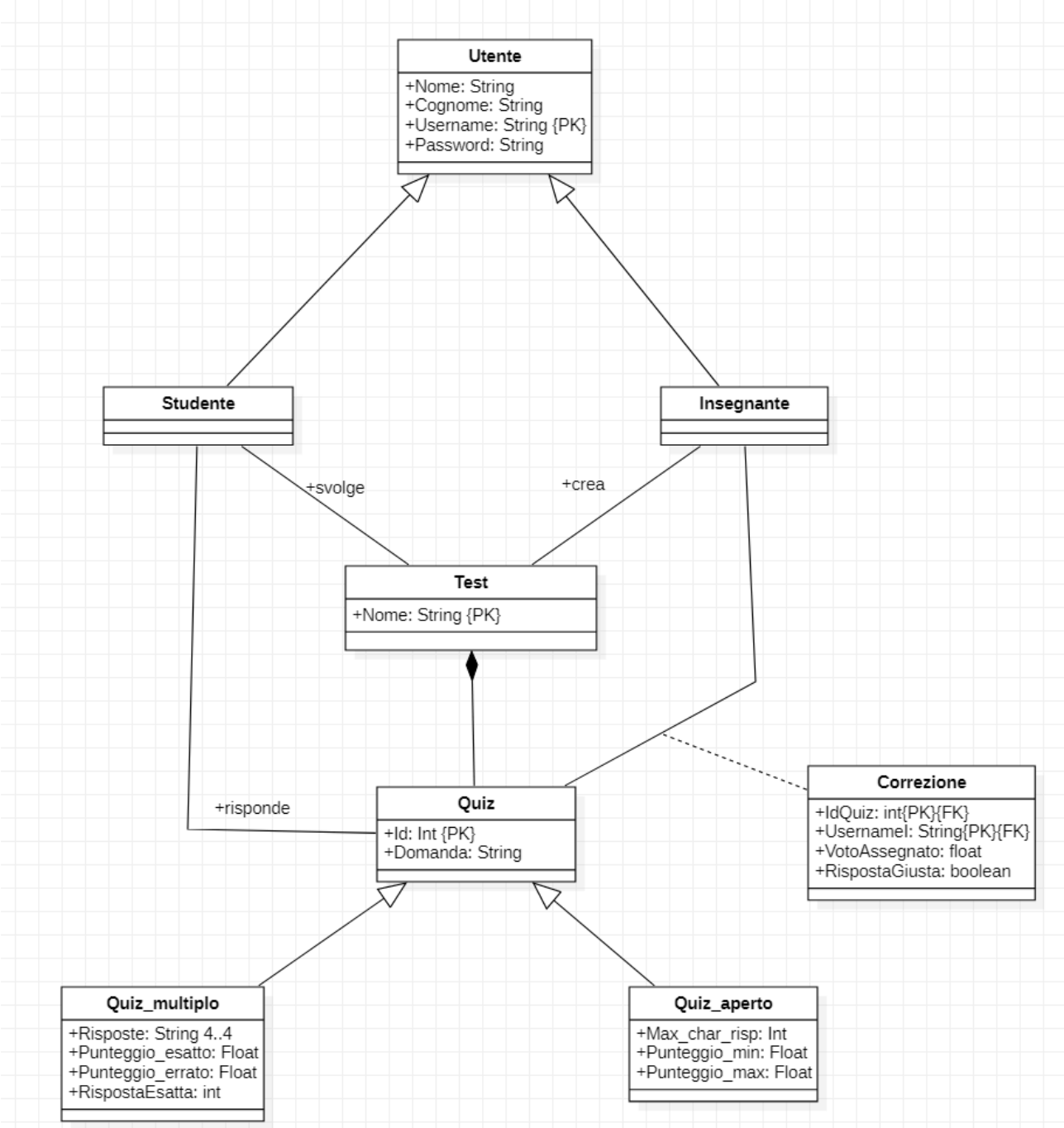
per i valori massimi (o risposta corretta) sono permessi valori decimali maggiori di 0(zero)

Progettazione concettuale

Prima bozza

Dopo un' attenta analisi dei requisiti si è proceduto alla realizzazione di una prima bozza di class diagram.

Simeone Vitale
N86003606



Simeone Vitale
N86003606

Ristrutturazione class diagram

Per effettuare una corretta ristrutturazione del class diagram, bisogna effettuare le seguenti analisi:

- Analisi delle ridondanze
- Analisi degli identificativi
- Analisi (e rimozione) degli attributi multipli
- Analisi (e rimozione) di attributi composti
- Analisi delle gerarchie
- Analisi delle associazioni

Analisi delle ridondanze

Non sono presenti ridondanze

Analisi degli identificativi

Dove esplicitamente richiesto dalla traccia, sono stati utilizzati gli attributi indicati come identificativi,

Esempi

Nome per Test

Username per Studente e Insegnante (vedi class diagram ristrutturato)

Invece, ove non previsti esplicitamente, si è preferito utilizzare un id numerico con il vincolo Auto Increment

Esempi

Id per Quiz multiplo e Quiz aperto (vedi class diagram ristrutturato)

Analisi (e rimozione) degli attributi multipli

L'unico attributo multiplo è 'Risposte' di Quiz multiplo.

Nel class diagram ristrutturato verrà sostituito con gli attributi

'Opzione1', 'Opzione2','Opzione3','Opzione4'.

Analisi (e rimozione) degli attributi composti

Non sono presenti attributi composti

Analisi delle gerarchie

Sono presenti due gerarchie:

- Utente, Studente, Insegnante
  La classe padre Utente viene accorpata in Studente e Insegnante, in tal caso Studente ed Insegnante acquisiranno gli attributi di Utente.
  Questo è stato fatto per evitare query molto frequenti per ottenere il tipo di utente che va ad effettuare le operazioni.
- Quiz, Quiz multiplo, Quiz aperto
  La classe padre Quiz viene accorpata in Quiz multiplo e Quiz aperto, in tal caso Quiz multiplo e Quiz aperto acquisiranno gli attributi di Quiz.
  Si è preferito accorpare la classe padre nelle figlie per evitare valori Null, dato che le classi figlie presentano attributi esclusivi, oltre che per evitare query molto frequenti per ottenere il tipo di quiz su cui si va ad operare.

Simeone Vitale
N86003606

Analisi delle associazioni

È presente un'associazione tra le classi Quiz e Test.

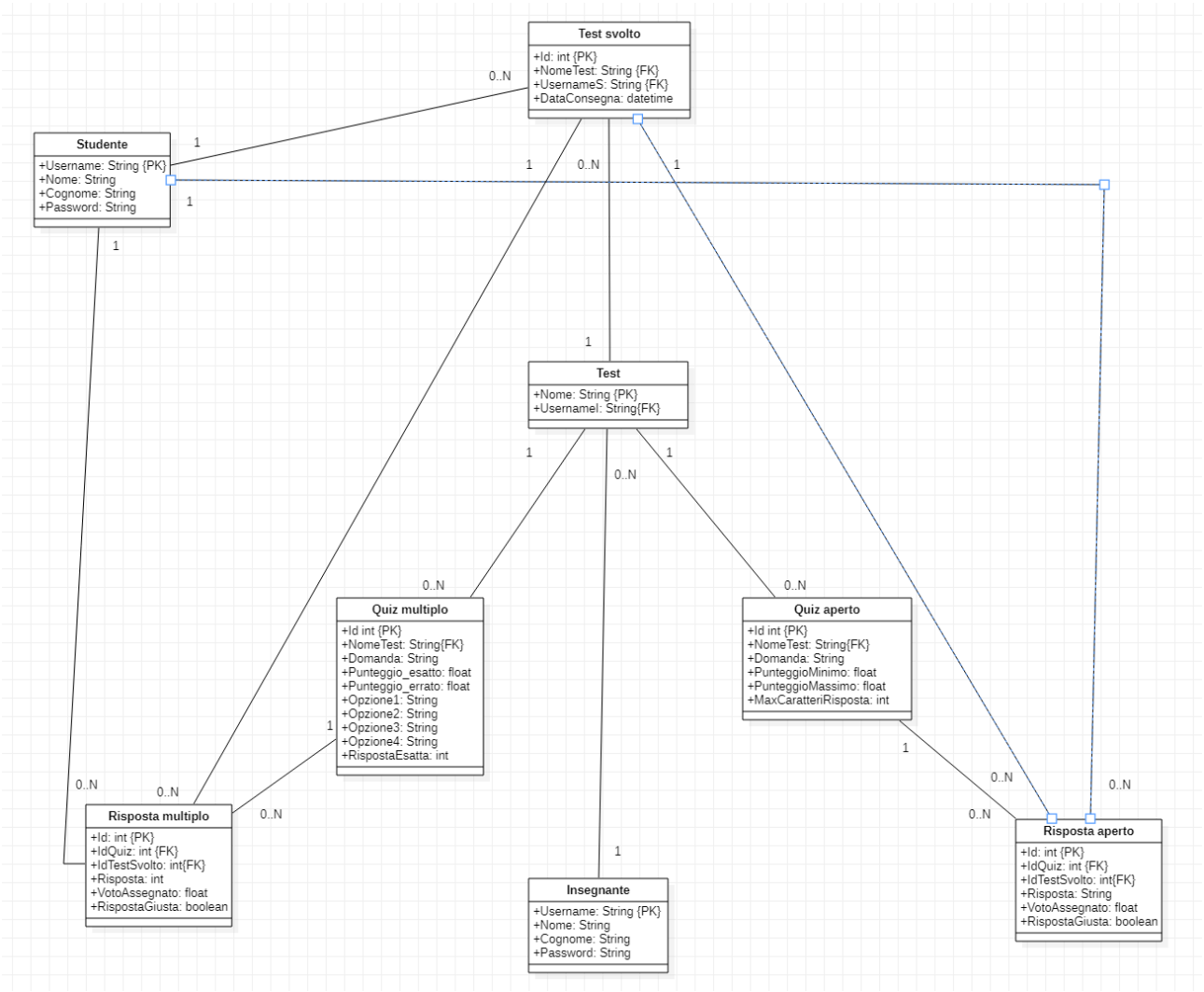
Nel class diagram ristrutturato, verrà creata la classe Quiz che,riporterà come chiave esterna la chiave primaria di Test.

Per evitare numerosi associazioni con cardinalità N..M si è reso necessario introdurre le classi:

- **Risposta multiplo e risposta aperto**: per registrare le risposte date ai quiz dagli studenti e per la loro correzione
- **Test svolto**: per associare tutti i test svolti dagli studenti (con i relativi quiz) a quelli creati dagli insegnanti.

Class diagram ristrutturato

Tenendo conto delle analisi svolte in precedenza, si è ricavato il seguente class diagram



Simeone Vitale
N86003606

Dizionario delle classi

Classe	Descrizione	Attributi
Studente	Descrittore dell'utente Studente	**Username** (String): Chiave primaria. Identifica univocamente un utente studente. **Password** (String): Password associata all'username. **Nome** (String): Nome dello studente. **Cognome** (String): Cognome dello studente.
Insegnante	Descrittore dell'utente Insegnante	**Username** (String): Chiave primaria. Identifica univocamente un utente insegnante. **Password** (String): Password associata all'username. **Nome** (String): Nome dell'insegnante. **Cognome** (String): Cognome dell'insegnante.
Test	Descrittore di un test	**Nome**(String): Chiave primaria. Identifica univocamente un test. **UsernameI**(String): Chiave esterna. Identifica il creatore del test
Test svolto	Descrittore della classe creata per la gestione di un test svolto da uno studente	**Id**(int):Chiave primaria. Identifica univocamente un test svolto. **NomeTest**(String): Chiave esterna. Identifica il test associato. **UsernameS**(String): Chiave esterna. Identifica lo studente che ha svolto il test associato. **DataConsegna**(datetime): istante di consegna del test svolto da parte dello studente.
Quiz multiplo	Descrittore di un quiz di tipo multiplo	**Id**(int):Chiave primaria. Identifica univocamente un quiz multiplo. **NomeTest**(String): Chiave esterna. Identifica il test a cui appartiene il quiz multiplo. **Domanda**(String): domanda del quiz. **Punteggio_esatto**(float): punteggio assegnato in caso di risposta corretta.

Simeone Vitale
N86003606

		Punteggio_errato(float): punteggio assegnato in caso di risposta errata. **Opzione1**(String): testo dell'opzione numero 1. **Opzione2**(String): testo dell'opzione numero 2. **Opzione3**(String): testo dell'opzione numero 3. **Opzione4**(String): testo dell'opzione numero 4. **RispostaEsatta**(int): intero di riferimento per la risposta ritenuta esatta, può assumere valori da 1 a 4.
Quiz aperto	Descrittore di un quiz di tipo aperto	**Id**(int):Chiave primaria. Identifica univocamente un quiz aperto. **NomeTest**(String): Chiave esterna. Identifica il test a cui appartiene il quiz aperto. **Domanda**(String): domanda del quiz. **PunteggioMinimo**(float): punteggio minimo assegnabile da parte dell'insegnante al quiz. **PunteggioMassimo**(float): punteggio massimo assegnabile da parte dell'insegnante al quiz. **MaxCaratteriRisposta**(int): numero di caratteri massimi permessi per la risposta.
Risposta multiplo	Descrittore della classe creata per la gestione di un quiz di tipo multiplo svolto da uno studente	**Id**(int):Chiave primaria. Identifica univocamente una risposta a un quiz multiplo. **IdQuiz**(int): Chiave esterna identifica il quiz multiplo associato. **Risposta**(int): intero di riferimento per la risposta inserita, può assumere valori da 1 a 4. **VotoAssegnato**(float): voto assegnato in base alla correttezza della risposta inserita.
Risposta aperto	Descrittore della classe creata per la gestione di un quiz di tipo aperto svolto da uno studente	**Id**(int):Chiave primaria. Identifica univocamente una risposta a un quiz aperto.

		IdQuiz(int): Chiave esterna identifica il quiz aperto associato. **Risposta**(String): risposta inserita dallo studente **VotoAssegnato**(float): voto assegnato dall'insegnante in fase di correzione.

Dizionario delle associazioni

Di seguito è riportato il dizionario delle associazioni.

I nomi delle associazioni non sono segnati sul class diagram per migliorarne la leggibilità, ma di seguito è presente tutto ciò che occorre per una corretta interpretazione.

Nome associazione	Tabelle coinvolte
Creazione Test	**Insegnante**[*] ruolo (creatore): indica quale insegnate ha creato il test. **Test**[1] ruolo (creato): indica il test creato
Contenenza quesiti multipli	**Test**[*] ruolo (contiene): indica il test che contiene i quiz. **Quiz multiplo**[1] ruolo (contenuto): indica il quiz contenuto nel test
Contenenza quesiti aperti	**Test**[*] ruolo (contiene): indica il test che contiene i quiz. **Quiz aperto**[1] ruolo (contenuto): indica il quiz contenuto nel test
Associazione test svolto	**Test**[*] ruolo (associato): indica il test associato al test svolto da studente **Test svolto**[1] ruolo(associa): associa un test a uno studente
Svolgimento test	**Studente**[*] ruolo(svolge): indica lo studente che ha svolto i test **Test svolto**[1] ruolo (svolto): indica il test svolto dallo studente
Contenenza risposta multiplo	**Test svolto**[*] ruolo (contiene): indica il test svolto che contiene i quiz. **Risposta multiplo**[1] ruolo (contenuto): indica il quiz contenuto nel test con la risposta data e correzione
Contenenza quesiti aperti	**Test svolto**[*] ruolo (contiene): indica il test svolto che contiene i quiz. **Risposta aperto**[1] ruolo (contenuto): indica il quiz

	contenuto nel test con la risposta data e correzione
Risposta a risposta multiplo	**Studente**[*] ruolo (risponde): indica lo studente che risponde al quiz. **Risposta multiplo**[1] ruolo (risposto): indica il quiz risposto dallo studente
Risposta a risposta aperto	**Studente**[*] ruolo (risponde): indica lo studente che risponde al quiz. **Risposta aperto**[1] ruolo (risposto): indica il quiz risposto dallo studente
Associazione risposta multiplo a quiz multiplo	**Quiz multiplo**[*] ruolo (associato): indica il quiz multiplo associato alla risposta. **Risposta multiplo**[1] ruolo (associa): indica la risposta data dallo studente associata al quiz multiplo
Associazione risposta aperto a quiz aperto	**Quiz aperto**[*] ruolo (associato): indica il quiz aperto associato alla risposta. **Risposta aperto**[1] ruolo (associa): indica la risposta data dallo studente associata al quiz aperto

Dizionario dei vincoli

Vincoli	Tipo	Descrizione
Unicità chiave primaria	Intrarelazionale	Stabilisce l'unicità delle chiavi primarie definite nelle tabelle
Check_maxCharRisp	N-upla	Controlla che il massimo dei caratteri per la risposta a un quiz aperto inputabile dall'utente sia minore o uguale del limite impostato sul db (500)
Check_punteggioMax	N-upla	Controlla che il punteggio massimo sia sempre positivo e maggiore del punteggio minimo
Check_punteggioMin	N-upla	Controlla che il punteggio minimo sia sempre negativo o minore di 0 e minore del punteggio massimo
Check_dominioRisposta	N-upla	Controlla che l'indice che indica la risposta a un quiz multiplo sia compreso tra 1 e 4 (estremi inclusi)
Check_credenziali	N-upla	Controlla che username e password siano formati da almeno 8 caratteri e il nome e cognome da almeno 2 caratteri.
Check_userName	Interrelazionale	Controlla che l'username inserito non sia registrato ne tra gli studenti ne tra gli insegnanti
Check_numQuiz	Interrelazionale	Controlla che siano stati inseriti al massimo 5 quiz per il test in oggetto
Check_votoAssegnato	Interrelazionale	Controlla che il voto assegnato sia presente nel dominio stabilito in fase di creazione del quiz
Check_risposta	Interrelazionale	c
Correggi_quizMultiplo	Interrelazionale	Verifica che la risposta di un quiz multiplo sia giusta o meno e assegna il punteggio e imposta il campo booleano 'rispostaGiusta'

Simeone Vitale
N86003606

Progettazione logica

Si prosegue con la progettazione logica della base di dati.

Le chiavi primarie sono evidenziate in grassetto, le chiavi esterne sono sottolineate.

Insegnante	**username**, password, nome , cognome
Studente	**username**, password, nome , cognome
Test	**nome**,usernameI usernameI → Insegnante.username
Test svolto	**Id**, nomeTest, usernameS, dataConsegna nomeTest → Test.nome usernameS → Studente.username
Quiz multiplo	**Id**, nomeTest, domanda , punteggio_esatto, punteggio_errato, opzione1, opzione2, opzione3, opzione 4, rispostaEsatta nomeTest → Test.nome
Quiz aperto	**Id**, nomeTest, domanda , punteggioMinimo, punteggioMassimo, maxCaratteriRisposta nomeTest → Test.nome
Risposta multiplo	**Id**, idQuiz, idTestSvolto, risposta, votoAssegnato , rispostaGiusta idQuiz → Quiz_multiplo.id idTestSvolto → Test_svolto.id
Risposta aperto	**Id**, idQuiz, idTestSvolto, risposta, votoAssegnato , rispostaGiusta idQuiz → Quiz_multiplo.id idTestSvolto → Test_svolto.id

Simeone Vitale
N86003606

Progettazione fisica e definizioni SQL

Introduzione

Adesso si passa alle definizioni in SQL di tabelle , vincoli , trigger e liste.

Il DBMS utilizzato per realizzare questa base di dati è PostgresSQL 15.

Procedure individuate

Per poter realizzare i trigger e i vincoli necessari al corretto funzionamento della base di dati in Postgres, si è reso necessario creare delle procedure. Di seguito sono elencate le funzioni individuate:

check_numQuiz: Verifica che il numero di quiz associati a un test non superi il valore massimo stabilito (5).

check_votoAssegnato: Controlla che il voto assegnato sia presente nel dominio stabilito in fase di creazione del quiz.

checkCredenzialiI : Controlla che username e password dell'insegnate inserito siano formati da almeno 8 caratteri e il nome e cognome da almeno 2 caratteri.

checkCredenzialiS: Controlla che username e password dello studente inserito siano formati da almeno 8 caratteri e il nome e cognome da almeno 2 caratteri.

checkRispostaAperta: Controlla che in risposta aperto la lunghezza della risposta inserita sia minore o uguale al limite stabilito in fase di creazione del quiz.

checkUsername: Controlla che l'username inserito non sia registrato ne tra gli studenti ne tra gli insegnanti.

Automazioni

In aggiunta alle procedure individuate in precedenza, sono state individuate le seguenti automazioni:

correggi_quizMultiplo: corregge la risposta ai quiz multipli.

check_dataConsegna: aggiunge al record inserito la data di consegna

Viste implementate

Get_contTestSvolti: per ogni test, mostra quanti studenti lo hanno svolto.

Get_countTestSvoltiPerStudente: per ogni studente, mostra il numero di test effettuati.

Get_risposteQuizApertiCorretti: per ogni quiz a risposta aperta, mostra tutte le risposte corrette.

Get_risposteQuizMultipliCorretti: per ogni quiz a risposta multipla, mostra tutte le risposte corrette.

Simeone Vitale
N86003606

Definizioni tabelle

Studente

```
CREATE TABLE studente(
username VARCHAR(25) NOT NULL,
password VARCHAR(25) NOT NULL,
nome VARCHAR(25),
cognome VARCHAR(25),
PRIMARY KEY (username)
);
```

Insegnante

```
CREATE TABLE insegnante(
username VARCHAR(25) NOT NULL,
password VARCHAR(25) NOT NULL,
nome VARCHAR(25),
cognome VARCHAR(25),
PRIMARY KEY (username)
);
```

Test

```
CREATE TABLE test(
nome VARCHAR(25) PRIMARY KEY,
usernameI VARCHAR(25),
FOREIGN KEY (usernameI) REFERENCES insegnante (username)
);
```

Test_svolto

```
CREATE TABLE test_svolto(
id SERIAL PRIMARY KEY,
nomeTest VARCHAR(25) NOT NULL,
usernameS VARCHAR(25) NOT NULL,
dataConsegna TIMESTAMP,
FOREIGN KEY (nomeTest) REFERENCES test (nome),
FOREIGN KEY (usernameS) REFERENCES studente (username)
);
```

Simeone Vitale
N86003606

Quiz_multiplo

```
CREATE TABLE quiz_multiplo(
id SERIAL PRIMARY KEY,
nomeTest VARCHAR(25) NOT NULL,
domanda VARCHAR(200) NOT NULL,
punteggioEsatto FLOAT NOT NULL,
punteggioErrato FLOAT NOT NULL,
opzione1 VARCHAR(100) NOT NULL,
opzione2 VARCHAR(100) NOT NULL,
opzione3 VARCHAR(100) NOT NULL,
opzione4 VARCHAR(100) NOT NULL,
risposta_esatta INT NOT NULL,
FOREIGN KEY (nomeTest) REFERENCES test (nome)
);
```

Quiz_aperto

```
CREATE TABLE quiz_aperto(
id SERIAL PRIMARY KEY,
nomeTest VARCHAR(25) NOT NULL,
domanda VARCHAR(200) NOT NULL,
punteggioMinimo FLOAT NOT NULL,
punteggioMassimo FLOAT NOT NULL,
maxCaratteriRisposta INT NOT NULL,
FOREIGN KEY (nomeTest) REFERENCES test (nome)
);
```

Risposta_multiplo

```
CREATE TABLE risposta_multiplo(
id SERIAL PRIMARY KEY,
idQuiz INT NOT NULL,
idTestSvolto INT NOT NULL,
risposta INT NOT NULL,
votoAssegnato FLOAT,
rispostaGiusta BOOLEAN,
FOREIGN KEY (idQuiz) REFERENCES quiz_multiplo (id),
FOREIGN KEY (idTestSvolto) REFERENCES test_svolto (id)
);
```

Risposta_aperto

```
CREATE TABLE risposta_aperto(

id SERIAL PRIMARY KEY,

idQuiz INT NOT NULL,

idTestSvolto INT NOT NULL,

risposta VARCHAR(500) NOT NULL,

votoAssegnato FLOAT,

rispostaGiusta BOOLEAN,

FOREIGN KEY (idQuiz) REFERENCES quiz_multiplo (id),

FOREIGN KEY (idTestSvolto) REFERENCES test_svolto (id)

);
```

Vincoli

Controllo punteggio quiz multiplo

```
ALTER TABLE quiz_multiplo

ADD CONSTRAINT check_punteggoMax CHECK ( punteggioEsatto > 0 AND punteggioEsatto > punteggioErrato),

ADD CONSTRAINT check_punteggoMin CHECK ( punteggioErrato <= 0 AND punteggioErrato < punteggioEsatto);
```

Controllo dominio risposta esatta quiz multiplo

```
ALTER TABLE quiz_multiplo

ADD CONSTRAINT check_dominioRisposta CHECK ( risposta_esatta >= 1 AND risposta_esatta <= 4);
```

Controllo punteggio quiz aperto

```
ALTER TABLE quiz_aperto

ADD CONSTRAINT check_punteggioMax CHECK ( punteggioMassimo > 0 AND punteggioMassimo > punteggioMinimo),

ADD CONSTRAINT check_punteggoMin CHECK ( punteggioMinimo <= 0 AND punteggioMinimo < punteggioMassimo);
```

Controllo caratteri massimi inseribili da utente

```
ALTER TABLE quiz_aperto

ADD CONSTRAINT check_maxCharRisp CHECK ( maxCaratteriRisposta > 0 AND maxCaratteriRisposta <= 500);
```

Controllo dominio risposta multiplo

```
ALTER TABLE risposta_multiplo

ADD CONSTRAINT check_dominioRisposta CHECK (risposta >= 1 AND risposta <= 4);
```

Simeone Vitale
N86003606

Trigger

Check credenziali studente

```
CREATE OR REPLACE FUNCTION checkCredenzialiS() RETURNS TRIGGER

LANGUAGE PLPGSQL

AS $checkCredenzialiS$

DECLARE

username studente.username%TYPE;
pwd studente.password%TYPE;
nome studente.nome%TYPE;
cognome studente.cognome%TYPE;

BEGIN

username := NEW.username;
pwd := NEW.password;
nome := NEW.nome;
cognome := NEW.cognome;



IF( LENGTH(username) < 8 OR LENGTH(pwd) < 8) THEN
        IF(LENGTH(username) < 8) THEN
                RAISE EXCEPTION 'Username non valido'
                USING HINT = 'Username deve essere formato da almeno 8 caratteri';
        END IF;

        IF(LENGTH(pwd) < 8) THEN
                RAISE EXCEPTION 'Password non valida'
                USING HINT = 'La password deve essere formata da almeno 8 caratteri';
        END IF;

END IF;
IF(LENGTH(nome) < 2 OR LENGTH(cognome) < 2) THEN
        RAISE EXCEPTION 'Nome o cognome non valido'
```

```
        USING HINT = 'Prova a inserire un nome e cognome vero!';
END IF;
RETURN NEW;
END $checkCredenzialiS$;

CREATE TRIGGER checkCredenziali AFTER INSERT OR UPDATE ON STUDENTE
FOR EACH ROW EXECUTE FUNCTION checkCredenzialiS();
```

Check credenziali insegnante

```
CREATE OR REPLACE FUNCTION checkCredenzialiI()
RETURNS TRIGGER
LANGUAGE plpgsql AS $checkCredenzialiI$

DECLARE

username insegnante.username%TYPE;
pwd insegnante.password%TYPE;
nome insegnante.nome%TYPE;
cognome insegnante.cognome%TYPE;

BEGIN

username := NEW.username;
pwd := NEW.password;
nome := NEW.nome;
cognome := NEW.cognome;

IF( LENGTH(username) < 8 OR LENGTH(pwd) < 8) THEN
        IF(LENGTH(username) < 8) THEN
                RAISE EXCEPTION 'Username non valido'
                USING HINT = 'Username deve essere formato da almeno 8 caratteri';
        END IF;

        IF(LENGTH(pwd) < 8) THEN
```

```
            RAISE EXCEPTION 'Password non valida'
            USING HINT = 'La password deve essere formata da almeno 8 caratteri';
        END IF;

END IF;
IF(LENGTH(nome) < 2 OR LENGTH(cognome) < 2) THEN
        RAISE EXCEPTION 'Nome o cognome non valido'
        USING HINT = 'Prova a inserire un nome e cognome vero!';
END IF;
RETURN NEW;
END $checkCredenzialiI$;

CREATE TRIGGER checkCredenziali AFTER INSERT OR UPDATE ON INSEGNANTE
FOR EACH ROW EXECUTE FUNCTION checkCredenzialiI();
```

Check username già registrato

```
CREATE OR REPLACE FUNCTION checkUsername() RETURNS TRIGGER LANGUAGE plpgsql AS $checkCredenziali$

DECLARE

username VARCHAR(25);
contAccountS INTEGER;
contAccountI INTEGER;
BEGIN


SELECT COUNT(*) INTO contAccountI
FROM insegnante
WHERE insegnante.username = NEW.username;

IF contAccountI > 0 THEN
        RAISE EXCEPTION 'Esiste già un insegnante associato a questo username!'
        USING HINT = 'Si prega di utilizzare un altro username';
END IF;

SELECT COUNT(*) INTO contAccountS
```

```
FROM studente
WHERE studente.username = NEW.username;

IF contAccountS > 0 THEN
        RAISE EXCEPTION 'Esiste già uno studente associato a questo username!'
        USING HINT = 'Si prega di utilizzare un altro username';
END IF;

RETURN NEW;

END $checkCredenziali$;

CREATE TRIGGER check_userName BEFORE INSERT OR UPDATE ON studente
FOR EACH ROW EXECUTE FUNCTION checkusername();
CREATE TRIGGER check_userName BEFORE INSERT OR UPDATE ON insegnante
FOR EACH ROW EXECUTE FUNCTION checkusername();
```

Verifica numero caratteri risposta quiz aperto

```
CREATE FUNCTION checkRispostaAperta() RETURNS TRIGGER LANGUAGE plpgsql AS $checkRispostaAperta$
DECLARE
maxCharRisp quiz_aperto.maxcaratteririsposta%TYPE;
BEGIN

SELECT maxcaratteririsposta INTO maxCharRisp
FROM quiz_aperto
WHERE id = NEW.idQuiz;

IF (LENGTH(NEW.risposta) > maxCharRisp) THEN
        RAISE EXCEPTION 'La risposta inserita supera il limite di % caratteri',maxCharRisp
        USING HINT = 'Riprova inserendo una risposta più breve';
END IF;
RETURN NEW;
END $checkRispostaAperta$;

CREATE TRIGGER checkRisposta AFTER INSERT OR UPDATE ON risposta_aperto
FOR EACH ROW EXECUTE FUNCTION checkRispostaAperta();
```

Verifica voto assegnato risposta multipla

```
CREATE FUNCTION check_votoAssegnato() RETURNS TRIGGER LANGUAGE plpgsql AS $check_votoAssegnato$

DECLARE
votoMin quiz_aperto.punteggioMinimo%TYPE;
votoMax quiz_aperto.punteggioMassimo%TYPE;

BEGIN

SELECT punteggioMinimo,punteggioMassimo INTO votoMin,votoMax
FROM quiz_aperto
WHERE id = NEW.idQuiz;

IF NEW.votoAssegnato > votoMax OR NEW.votoAssegnato < votoMin THEN
        RAISE EXCEPTION 'Il punteggio inserito non rispetta il range inserito in fase di creazione del quiz (%<=x<=%)',votoMin,votoMax
        USING HINT = 'Riprova inserendo un punteggio conforme';
END IF;
RETURN NEW;
END $check_votoAssegnato$;

CREATE TRIGGER check_votoAssegnato AFTER INSERT OR UPDATE ON risposta_aperto
FOR EACH ROW EXECUTE FUNCTION check_votoAssegnato();
```

Simeone Vitale
N86003606

Verifica numero quiz associati a un test

```
CREATE FUNCTION check_numQuiz() RETURNS TRIGGER LANGUAGE plpgsql AS $check_numQuiz$
DECLARE
numQuizMultiplo INTEGER;
numQuizAperto INTEGER;
totQuiz INTEGER;
BEGIN

SELECT COUNT(*) INTO numQuizMultiplo
FROM quiz_multiplo
WHERE nomeTest = NEW.nomeTest;

IF (numQuizMultiplo >= 5) THEN
        RAISE EXCEPTION 'Sono presenti già 5 quiz a risposta multipla, impossibile inserire ulteriori quiz';
END IF;

SELECT COUNT(*) INTO numQuizAperto
FROM quiz_aperto
WHERE nomeTest = NEW.nomeTest;

IF (numQuizAperto >= 5) THEN
        RAISE EXCEPTION 'Sono presenti già 5 quiz a risposta aperta, impossibile inserire ulteriori quiz';
END IF;
totQuiz := numQuizMultiplo + numQuizAperto;
IF (totQuiz >= 5) THEN
        RAISE EXCEPTION 'Sono presenti già 5 quiz, impossibile inserire ulteriori quiz';
END IF;
RETURN NEW;
END $check_numQuiz$;

CREATE TRIGGER check_numQuiz BEFORE INSERT OR UPDATE ON quiz_aperto
FOR EACH ROW EXECUTE FUNCTION check_numQuiz();

CREATE TRIGGER check_numQuiz BEFORE INSERT OR UPDATE ON quiz_multiplo
FOR EACH ROW EXECUTE FUNCTION check_numQuiz();
```

Simeone Vitale
N86003606

Automazioni

Correzione quiz a risposta multipla

```
CREATE FUNCTION correggi_quizMultiplo() RETURNS TRIGGER LANGUAGE plpgsql AS $correggi_quizMultiplo$

DECLARE
votoGiusta quiz_multiplo.punteggioEsatto%TYPE;
votoErrato quiz_multiplo.punteggioErrato%TYPE;
indiceRispostaGiusta quiz_multiplo.risposta_esatta%TYPE;
BEGIN

SELECT punteggioEsatto,punteggioErrato,risposta_esatta INTO votoGiusta,votoErrato,indiceRispostaGiusta
FROM quiz_multiplo
WHERE id = NEW.idQuiz;

IF NEW.risposta = indiceRispostaGiusta THEN
        NEW.votoAssegnato := votoGiusta;
        NEW.rispostaGiusta := TRUE;
ELSE
        NEW.votoAssegnato := votoErrato;
        NEW.rispostaGiusta := FALSE;
END IF;
RETURN NEW;
END $correggi_quizMultiplo$;

CREATE TRIGGER correggi_quizMultiplo BEFORE INSERT OR UPDATE ON risposta_multiplo
FOR EACH ROW EXECUTE FUNCTION correggi_quizMultiplo();
```

Simeone Vitale
N86003606

Imposta data consegna test

```
CREATE FUNCTION check_dataConsegna() RETURNS TRIGGER LANGUAGE plpgsql AS $check_dataConsegna$

BEGIN

NEW.dataConsegna := current_timestamp(0);

RETURN NEW;

END $check_dataConsegna$;

CREATE TRIGGER assegnaDataConsegna  BEFORE INSERT ON test_svolto

FOR EACH ROW EXECUTE FUNCTION check_dataConsegna();
```

Simeone Vitale
N86003606

View

Visualizza tutte le risposte esatte date ai quiz a risposta multipla

```
CREATE VIEW get_risposteQuizMultipliCorretti AS (
        SELECT ts.nometest,q.domanda,r_m.votoAssegnato,s.nome,s.cognome
        FROM
        risposta_multiplo as r_m,quiz_multiplo as q,test as t,test_svolto as ts,studente as s
        WHERE r_m.idQuiz = q.id AND q.nomeTest = t.nome  AND r_m.idtestsvolto = ts.id AND ts.usernameS = s.username
AND r_m.votoAssegnato = q.punteggioesatto
);
```

Visualizza tutte le risposte esatte date ai quiz a risposta aperta

```
CREATE VIEW get_risposteQuizApertiCorretti AS (
 SELECT test_svolto.nomeTest,
        risposta_aperto.id,
    risposta_aperto.idquiz,
    risposta_aperto.idtestsvolto,
    risposta_aperto.risposta,
    risposta_aperto.votoassegnato,
    risposta_aperto.rispostagiusta
   FROM risposta_aperto,test_svolto
  WHERE risposta_aperto.rispostagiusta = true AND test_svolto.id = risposta_aperto.idtestsvolto

);
```

Simeone Vitale
N86003606

Visualizza il numero di studenti che hanno svolto un test

```
CREATE VIEW get_contTestSvolti AS(

        SELECT test.nome,COUNT(*) as Test_Svolti
        FROM test_svolto,test
        WHERE test_svolto.nomeTest = test.nome
        GROUP BY test.nome
);
```

Numero di test svolti per ogni studente

```
CREATE VIEW get_countTestSvoltiPerStudente AS(

        SELECT s.username,s.nome,s.cognome,COUNT(*) as numero_test_svolti
        FROM studente as s,test_svolto as ts,test as t
        WHERE s.username = ts.usernameS AND t.nome = ts.nomeTest
        GROUP BY s.username

);
```

Simeone Vitale
N86003606