Progettazione e sviluppo di una base di dati relazionale per una applicazione di e-learning tramite test basati su quiz

Autori
Antonio Legnante N86003588
Matteo Marino N86003963

Docente
Silvio Barra

Gennaio 2022

Questa pagina è stata lasciata volutamente bianca

Indice

Capitolo 1

Analisi del problema e requisiti identificati

Si progetterà ed implementerà una base di dati relazionale adibita alla gestione di dati relativi ad un applicazione di E-learning tramite quiz. Dalla realtà di riferimento si evincono i seguenti tipi di entità fondamentali:

- **Utente** Sono le entità principali di interesse per l'applicazione. Si dividono in 2 categorie: insegnante e studente. Gli studenti sostengono i test e hanno accesso alle relative valutazioni, gli insegnante registrano i test e forniscono le valutazioni agli studenti. Riguardo gli utenti ci interessa memorizzare: nome, cognome, email e password.

- **Test** Le altre entità principali per l'applicazione, un test è un aggregato di domande a cui gli studenti sono tenuti a rispondere per sostenere le prove. Sono di interesse: il nome del test, la data di registrazione, la categoria, il numero di quiz, il punteggio minimo, massimo per risposta e il punteggio massimo globale.

- **Domanda** Le domande sono suddivise in Domande a risposta aperta e domande a risposta multipla. Le domande a risposta multipla sono correlate da una sequenza di risposte automatizzate di cui una risulta essere corretta. È di interesse il testo che descrive la domanda.

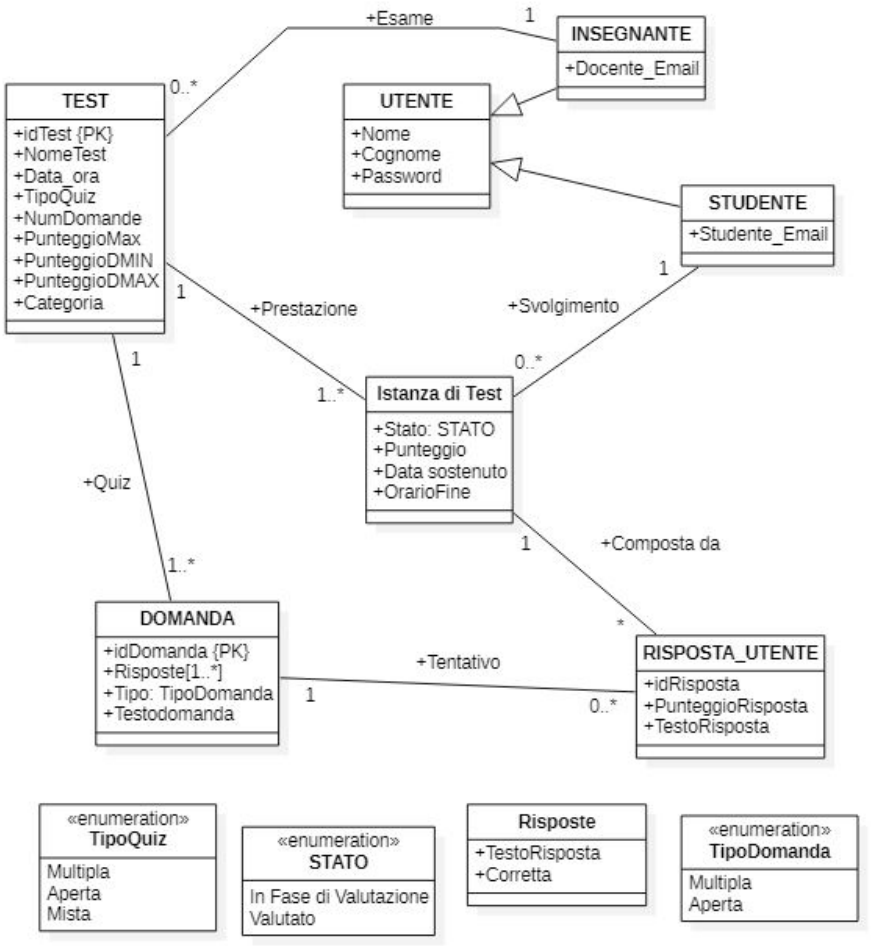
- **Istanza di test** Questa entità descrive i singoli test svolti dagli studenti. Sono inclusi i dati relativi al quando é stata sostenuta la prova, sul punteggio totalizzato e sullo stato della valutazione(e quindi conseguentemente indica se il punteggio totalizzato sia parziale o totale). Una istanza di test è un aggregato di risposteUtente fornite dallo studente.

- **rispostautente** Questa entità rappresenta la singola risposta di un utente fornita in una specifica istanza di test e ad una specifica domanda di cui memorizziamo il testo della risposta e il relativo punteggio acquisito

Capitolo 2

Progettazione concettuale

2.1 Class Diagram



Link per la visione dello schema concettuale

2.1.1 Alcune precisazioni sul class Diagram [Attributo Email]

Discutiamo brevemente delle caratteristiche della classe Utente:
Essa si presenta come una generalizzazione delle classi ben più specifiche di studente e insegnante, le quali hanno in comune tutti gli attributi ad eccezione del campo Email, nei quali presenta un dominio differente. Nello specifico l'email presenta un

formato prestabilito:
nomeUtente@Docenti.Universita.it per gli Insegnanti
nomeUtente@Studenti.Universita.it per gli Studenti

2.1.2 Alcune precisazioni sul class Diagram [Attributo Risposte]

La classe Domanda ha, tra gli attributi che la descrivono, il campo risposte. Esso si presenta come attributo multiplo e composto dai sottoattributi testo risposta e corretta. Di conseguenza rientra nella definizione di attributo complesso o alternativamente attributo entità. L'attributo è opzionale giacché non tutte le domande hanno un set predefinito di risposte automatizzata(solo quelle a risposta multipla)

2.2 Analisi della ristrutturazione del Class Diagram

2.2.1 Analisi delle ridondanze

Nel diagramma sono presenti gli attributi NumDomande, PunteggioMax, TipoQuiz che possono essere derivabili da altre entità o attributi. Tuttavia, considerando che si trattano di attributi qualificativi dei test a scelta e proiettandoci nell'ottica dell'applicativo, nella quale anche solo visualizzare un test comporterebbe il calcolo di ciascuno degli attributi sopra citati, motiva la decisione di mantenerli. Si descrivono nel dettaglio i seguenti:
- **NumDomande** tiene conto del numero di domande dei test, i quali potrebbero essere ricavati effettuando una count delle domande effettivamente inserite
- **TipoQuiz** indica a priori il tipo del test, ricavabili controllando per ogni test tutte le domande presenti per stabilire se si tratta di un test a risposta multipla, aperta o mista,
- **PunteggioMax** indica il punteggio massimo ottenibile ad un test, ricavabile moltiplicando il numero di domande per il punteggio massimo per singola domanda,

Si decide di introdurre anche i seguenti dati derivabili da risposta utente e domanda:
- **numeroRcorrette** indica il numero di risposte corrette fornite dallo studente in una data istanza di test,
- **numerorRsbagliate** indica il numero di risposte, sbagliate fornite dallo studente in una data istanza di test.

2.2.2 Analisi degli identificativi

Per tutte le classi si è scelto di impiegare chiavi artificiali in modo da evitare l'impiego di chiavi primarie composte, come ad esempio (Test, datasostenuto, Studente, numeroDomanda) per risposta utente. Per questioni di omogeneità l'uso di chiavi artificiali è stato esteso anche per le classi che non presentavano il suddetto problema. Tale scelta comporterà una riduzione dello sforzo computazionale richiesto nella ricerca di un determinata tupla.

2.2.3 Rimozione degli attributi multipli

Nel diagramma è presente un significativo attributo multiplo nella classe Domanda. Quest'ultimo descrive le risposte ad una domanda a risposta chiusa. Vista la natura

da attributo complesso, in quanto oltre al testo della risposta è necessario indicare anche se sia corretta o errata, si decide di convertirlo in una classe apposta

2.2.4 Rimozione degli attributi composti

Nel diagramma è presente l'attributo composto risposta nella classe domanda, la quale si è già discusso nella sottosezione precedente. Un altro attributo composto significativo è Data ora e data sostenuto rispettivamente in Test e istanzaDiTest, adibito alla memorizzazione sia della data in cui è stato registrato o svolto un test, sia dell'ora effettiva; visto lo scarso interesse, da parte dell'applicazione, nel effettuare query sui singoli valori atomici, quali ad esempio test svolti in un certa data è stato convenuto di manterli

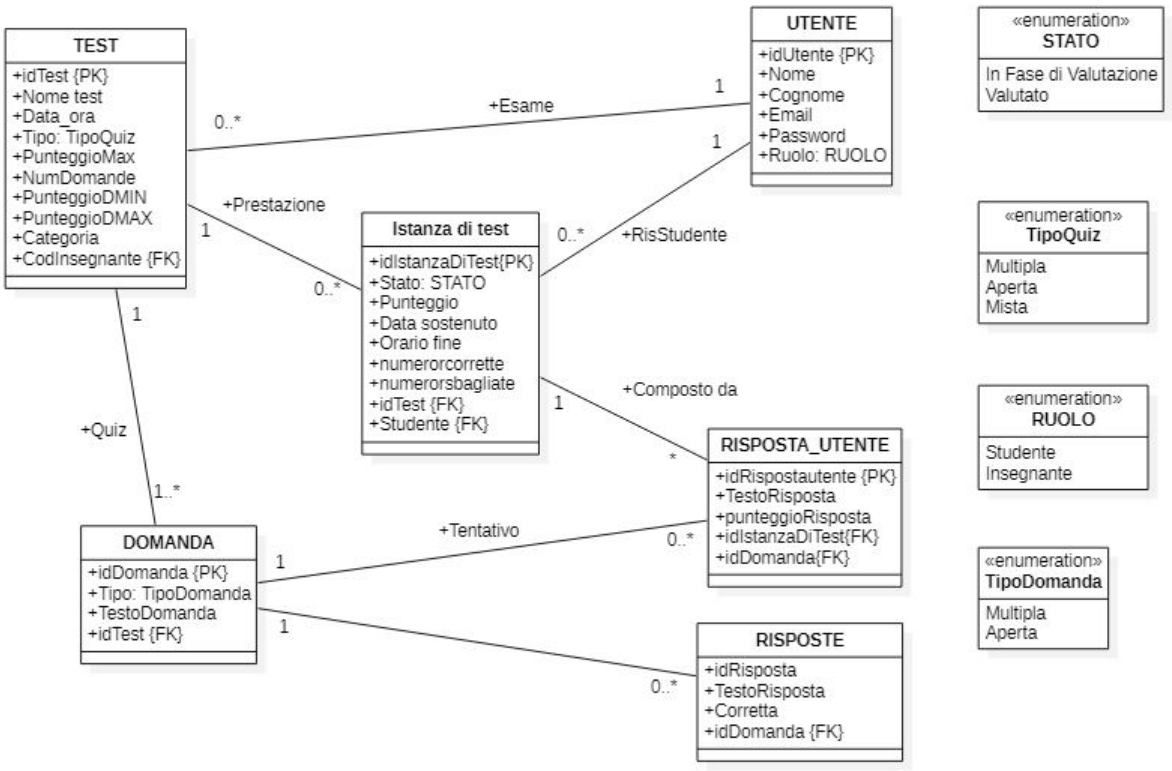
2.2.5 Partizione/Accorpamento delle associazioni

Non vi sono presenti associazione da accorpare nel diagramma.

2.2.6 Rimozione delle gerarchie

Nel diagramma è presente la gerarchia che riguarda la classe **Utente** che si specializza nelle classi **Studente** e **Docente**. Tale gerarchia viene eliminata inserendo nella classe Utente l'attributo *ruolo*, associato ad una enumerazione che specifica se l'utente è uno studente o un insegnante.

2.3 Class Diagram ristrutturato



Link per la visione dello schema concettuale ristrutturato

2.4 Dizionario delle classi

Classe	Descrizione	Attributi
Utente	Contiene i dati degli utenti, specificandone il ruolo	**Nome** (*String*): nome dell'utente. **Cognome** (*String*): cognome dell'utente. **Password** (*String*): password di accesso dell'utente. **Email** (*String*): indirizzo email dell'utente. **Ruolo** (*enumeration*): ruolo dell'utente.
Test	Descrive i dati relativi ai test	**idTest** (*Integer*): codice univoco per ogni test. **nomeTest** (*String*): Nome dato al test. **DataOra** (*Timestamp*): data e ora di registrazione del test. **tipoQuiz** (*enumeration*): Tipologia di quiz: Aperto, multiplo e mista. **punteggioMax** (*Float*): Massimo punteggio che può essere totalizzato da uno studente. **numDomande** (*Integer*): Numero di domande che costituiscono il test **PunteggioDMin** (*Float*): punteggio di ogni domanda in caso di risposta errata (può essere anche negativo) **PunteggioDMAX** (*Float*): punteggio di ogni domanda in caso di risposta esatta. **Categoria** (*String*): categoria del test. **Insegnante** (*Integer*): identificativo dell'insegnante che crea il test.
Domanda	Descrive i dati delle domande che sono presenti nei test	**idDomanda** (*Integer*): codice che identifica univocamente le domande. **TipoDomanda** (*Enumeration*): indica se la domanda è a risposta multipla o aperta. **TestoDomanda** (*String*): contiene il testo della domanda. **idTest** (*Integer*): codice del test in cui è presente la domanda.

Classe	Descrizione	Attributi
Risposta	Descrive le possibili risposte alle domande (solo quelle a risposta multipla)	**idRisposta** (*Integer*): codice identificativo di risposta. **TestoRisposta** (*String*): testo della possibile risposta. **Corretta** (*Boolean*): flag che determina se la possibile risposta è quella corretta. **idDomanda** (*Integer*): codice della domanda associata alla possibile risposta.
IstanzaDiTest	Descrive i risultati ottenuti da uno studente che sostiene un test	**idIstanzaDiTest** (*Integer*): codice identificativo di un istanza di test. **Stato** (*Enumeration*): stato di correzione del test. **Punteggio** (*Float*): punteggio ottenuto dallo studente nel test. **dataSostenuto** (*Timestamp*): Indica la data e l'ora in cui uno studente avvia un test. **OrarioFine** (*Time*): Indica l'ora in cui uno studente termina un test. **numeroRcorrette** (*Integer*): indica il numero di risposte corrette fornite dallo studente **numeroRsbagliate** (*Integer*): indica il numero di risposte sbagliate fornite dallo studente test. **idTest** (*Integer*): codice del test sostenuto. **Studente** (*Integer*): identificativo dello studente che sostiene il test.
RispostaUtente	Descrive le risposte fornite dagli studenti alle domande	**idRispostaUtente** (*Integer*): codice univoco di ogni risposta. **TestoRisposta** (*String*): testo della risposta dell'utente alla domanda. **punteggioRisposta** (*Float*): punteggio conseguito dallo studente riferito alla specifica risposta **idIstanzaDiTest** (*Integer*): identificativo dell'istanza di test, nella quale lo studente che fornisce la risposta. **idDomanda** (*Integer*): identificativo della domanda a cui lo studente fornisce la risposta.

2.5 Dizionario dei vincoli

Vincolo	Classi coinvolte	Descrizione
Email unica	Utente	Non possono esistere 2 utenti registrati con la stessa Email
Formato Email	Utente	Una Email è valida per la registrazione se e solo se il suo dominio è @Studenti.Universita.it per gli Studenti e @Docenti.Universita.it per gli Insegnanti
Consistenza Ruolo	Utente	Un insegnante non può essere registrato con l'email di uno Studente e viceversa uno Studente non può essere registrato con l'Email di un Insegnante
Registrazione Test	Utente, Test	Un test può essere registrato esclusivamente da un Insegnante
Consistenza NumDomande	Test, Domanda	Non vi possono essere più domande associate allo stesso test di quelle specificate da numDomande
Consistenza MaxPunteggio	Test	MaxPunteggio deve coincidere con la seguente espressione: numDomande * punteggioDMax
punteggioDMax positivo	Test	Il punteggio massimo per domanda totalizzabile dallo studente per un test deve essere Non negativo
punteggioDMin negativo	Test	Il punteggio minimo per domanda totalizzabile dallo studente per un test deve essere negativo
Risposta Automatizzata	Domanda, Risposta	Una risposta multipla può riferirsi solo ad una domanda a risposta Multipla
Sostenere il test	Utente, IstanzaDiTest	Un istanza di test può riferirsi solo ad uno studente
Viaggio nel tempo	Test, IstanzaDiTest	Un istanza di test non può essere sostenuto in una data antecedente alla creazione del test

Vincolo	Classi coinvolte	Descrizione
IstanzaDiTest terminato	IstanzaDiTest	Un Istanza di test risultata terminata solo quando viene aggiornato il valore OrarioFine da NULL all'orario di termine della prova
Consistenza OrarioFine	IstanzaDiTest	Un Istanza di test non può terminare prima della orario di inizio dello stesso
Lunghezza risposta	RispostaUtente	Una risposta ha una lunghezza massima di 256 caratteri
Risposta a quesiti multipli	RispostaUtente, Domanda, Risposta	Una risposta ad un quesito multiplo deve coincidere con una risposta automatizza associata alla domanda specifica
Consistenza RisposteUtente	RispostaUtente	Per ogni istanza di test, è possibile fornire una sola risposta per domanda
Consistenza Domande RisposteUtente	RispostaUtente, Test, IstanzaDiTest	Il test associato all'istanza di test deve coincidere con il test associato alla domanda risposta dallo studente
Correzione risposta Utente	RispostaUtente, Test	Il punteggio associato alla singola domanda deve essere compreso tra il punteggioDMin e il punteggioDMax
Consistenza Risultato	IstanzaDiTest, rispostaUtente	Il risultato del test ottenuto dallo studente deve essere pari alla somma dei punteggi ottenuti sulle singole domande
Domanda corretta	IstanzaDiTest, rispostaUtente	Una rispostaUtente risulta corretta se ottiene il punteggio massimo previsto per quella domanda
Correzione Fantasma	rispostaUtente	Una correzione non può essere effettuata prima che lo studente abbia terminato il test

2.6 Dizionario delle associazioni

Associazione	Descrizione	Classi coinvolte
Esame	Descrive quale insegnante ha creato il test	**Utente** [*1*] ruolo: indica quale insegnante crea il test. **Test** [*0..**] ruolo: indica quale test è stato creato dall'insegnante.
Quiz	Descrive da quali domande è composto ogni test	**Test** [*1*] ruolo: indica a quale test fanno riferimento le domande. **Domanda** [*1..**] ruolo: indica da quali domande è composto ogni test.
Prestazione	Descrive l'esito di un istanza di test	**Test** [*1*] ruolo: indica a quale test è riferito l'esito. **IstanzaDiTest** [] ruolo: indica l'esito ottenuto nel test.
RisStudente	Descrive i risultati degli studenti	**Utente** [*1*] ruolo: indica a quale studente fanno riferimento gli esiti. **IstanzaDiTest** [*0..**] ruolo: indica i risultati ottenuti dagli studenti.
RispostaStudente	Indica le risposte fornite dagli studenti alle domande in una determinata istanza di test	**IstanzaDiTest** [*1*] ruolo: indica quale studente fornisce la risposta. **RispostaUtente** [*0..**] ruolo: indica le risposte fornite dagli studenti.
Tentativo	Indica le risposte fornite dagli studenti alle domande	**RispostaUtente** [*0..**] ruolo: indica le risposte fornite alle domande. **Domanda** [*1*] ruolo: indica le domande al quale fanno riferimento le risposte.
PossibiliSoluzioni	Indica le possibili soluzioni alle domande (solo domande a risposta multipla)	**Risposte** [*0..**] ruolo: indica le possibili risposte alle domande. **Domanda** [*1*] ruolo: indica a quale domanda riferiscono le risposte.

2.7 Varianti del class Diagram

Vengono presentanti in questa sezione alcune variazioni del class Diagram ragionate:

2.7.1 Associazione tra rispostaUtente e risposta

Una possibile variazione consisterebbe nel implementare un associazione aggiuntiva tra risposta e rispostaUtente in maniera tale da referenziare le risposte automatizzate. Questa soluzione è stata scartata per il seguente motivo:
Aumento significativo di valori NULL: Poiché non tutte le domande sono a risposta multipla, le risposteUtente ad una domanda a risposta aperta comporterebbero inevitabilmente la chiave esterna ad divenire un attributo NULL. Se pensiamo a un test di 10 domande a risposta aperta svolto da 10 studenti al giorno, si registrerebbero 100 valori NULL; d'altro canto la stessa situazione si verificherebbe anche nel caso in cui il test fosse costituito da domande a risposta multipla, giacché, in questo caso sarebbe l'attributo testoRisposta ad essere NULL

2.7.2 Associazione tra rispostaUtente e Studente a sostituzione dell'associazione RispostaStudente tra rispostaUtente e IstanzaDiTest

Un altra possibile variazione consisterebbe nel sostituire l'associazione tra rispostaStudente con una nuova associazione, nella quale la risposta utente referenzia lo studente(Utente) che l'ha fornita. In questo caso, uno studente ha svolto un test, nel momento in cui ha risposto ad almeno una domanda del suddetto test e di conseguenza una procedure si sarebbe occupata di inserire, nel caso in cui non ci fosse, una tupla in IstanzaDiTest. Questa soluzione è stata scartata per i seguenti motivi:
Rottura logica tra rispostaUtente e IstanzaDiTest: ad essere precisi: IstanzaDiTest, RispostaUtente e Utente dovrebbero costituire un associazione ternaria che, in linea generale, per fini computazionali si preferisce scartare. Ciò nonostante, eliminare un associazione diretta tra istanzaDiTest e rispostaUtente, risulterebbe errato da un punto di vista logico
Modellizzazione più generale: le risposteUtente vengono memorizzate nel database in un unica transizione, pertanto, se uno studente non consegnasse le risposte, non verrebbe memorizzato la sua partecipazione, oggetto di cui potremmo essere interessati a fini statistici. Un altro aspetto è legato alla possibilità di sostenere un test più volte, per quanto sia difficile.

Capitolo 3

Progettazione Logica

Di seguito viene mostrato lo schema logico

Utente	(idUtente, Nome, Cognome, Email, Password, Ruolo)
Test	(idTest, nomeTest, Data$_Ora$, $tipoQuiz$, $MaxPunteggio$, $NumDomande$, $Catego$ **Insegnante** $\mapsto$ **Utente(idUtente)**
Domanda	(idDomanda, tipoDomanda, testoDomanda, idTest) **idTest** $\mapsto$ **Test(idTest)**
Risposta	(idRisposta, testoRisposta, Corretta, idDomanda) **idDomanda** $\mapsto$ **Domanda(idDomanda)**
IstanzaDiTest	(idIstanzaDiTest, Stato, Risultato, DataSostenuto, OrarioFine, numeroRcorrette, numeroRSbagliate, idTest, Studente) **idTest** $\mapsto$ **Test(idTest)** **Studente** $\mapsto$ **Utente(idUtente)**
RispostaUtente	(idrispostaUtente, testoRisposta, punteggioRisposta,idIstanzaDiTest, idDomanda) **idIstanzaDiTest** $\mapsto$ **idIstanzaDiTest(idIstanzaDiTest)** **idDomanda** $\mapsto$ **Domanda(idDomanda)**

Capitolo 4

Progettazione Fisica

Qui di seguito verranno proposti le implementazione effettive delle tabelle, dei domini e delle procedure atte ad implementare i vincoli discussi nei capitoli precedenti. Il DBMS scelto è PostegreSql Versione 14

4.1 Definizione delle tabelle e relativi vincoli di dominio e vincoli intrarelazionali

Link per la visione delle tabelle e dei vincoli intrarelazionali

4.1.1 Utente

Dominio FormatoEmail

```
CREATE DOMAIN FormatoEmail AS VARCHAR(64)
CHECK (VALUE LIKE '%@Studenti.Universita.it' OR VALUE LIKE '%Docenti.Universita.it'
    );
```

Enumerazione Ruolo

```
CREATE DOMAIN ruolo AS VARCHAR(16)
CHECK (VALUE = 'Studente' OR VALUE = 'Insegnante');
```

Tabella Utente

```
CREATE TABLE Utente
(
    idUtente BIGSERIAL NOT NULL,
    Nome VARCHAR(32) NOT NULL,
    Cognome VARCHAR(32) NOT NULL,
    Email FormatoEmail NOT NULL,
    Password VARCHAR(32) NOT NULL,
    Ruolo ruolo,
    PRIMARY KEY(idUtente)
);
```

Unicità Email

```
ALTER TABLE Utente ADD UNIQUE(Email);
```

Consistenza Ruolo

```
ALTER TABLE Utente ADD CONSTRAINT ConsistenzaRuolo CHECK((Ruolo = 'Studente' AND
    Email LIKE '%@Studenti.Universita.it') OR (Ruolo = 'Insegnante' AND Email LIKE
    '%@Docenti.Universita.it'));
```

4.1.2 Test

Enumerazione TipoQuiz

```
CREATE DOMAIN tipoQuiz VARCHAR(16)
CHECK (VALUE = 'Multipla' OR VALUE = 'Aperta' OR VALUE = 'Mista');
```

Dominio punteggioDmax

```
CREATE DOMAIN punteggioDmax AS FLOAT
CHECK (VALUE > 0 OR VALUE = 0);
```

Dominio punteggioDmin

```
CREATE DOMAIN punteggioDmin AS FLOAT
CHECK (VALUE < 0 OR VALUE = 0);
```

Tabella Test

```
CREATE TABLE Test
(
    idTest BIGSERIAL NOT NULL,
  nomeTest VARCHAR(32),
    Data_Ora TIMESTAMP NOT NULL,
  TipoQuiz tipoQuiz NOT NULL,
    MaxPunteggio FLOAT DEFAULT 0,
    NumDomande INTEGER DEFAULT 0,
    PunteggioDMax punteggioDmax NOT NULL DEFAULT 0,
    PunteggioDMin punteggioDmin DEFAULT 0,
    Categoria VARCHAR(32) NOT NULL,
    Insegnante BIGSERIAL NOT NULL,
    PRIMARY KEY (idTest),
    FOREIGN KEY (Insegnante) REFERENCES Utente (idUtente)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

4.1.3 Domanda

Enumerazione tipoDomanda

```
CREATE DOMAIN tipoDomanda AS VARCHAR(16)
CHECK (VALUE = 'Multipla' OR VALUE = 'Aperta');
```

Tabella Domanda

```
CREATE TABLE Domanda
(
    idDomanda BIGSERIAL NOT NULL,
    TipoDomanda tipoDomanda NOT NULL,
    TestoDomanda    VARCHAR(256)  NOT NULL,
    idTest    BIGSERIAL  NOT NULL,
    PRIMARY KEY (idDomanda),
    FOREIGN KEY (idTest) REFERENCES test (idTest)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

4.1.4 Risposta

Tabella Risposta

```
CREATE TABLE Risposta (
  idRisposta BIGSERIAL NOT NULL,
    idDomanda  BIGSERIAL    NOT NULL,
    TestoRisposta  VARCHAR(512),
    Corretta    BOOLEAN,
    PRIMARY KEY (idRisposta),
    FOREIGN KEY (idDomanda) REFERENCES Domanda (idDomanda)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

4.1.5 IstanzaDiTest

Enumerazione StatoValutazione

```
CREATE DOMAIN StatoValutazione VARCHAR(32)
CHECK (VALUE = 'In fase di valutazione' OR VALUE = 'Valutato') DEFAULT 'In fase di
    valutazione';
```

Tabella IstanzaDiTest

```
CREATE TABLE IstanzaDiTest
(
   idIstanzaDiTest BIGSERIAL NOT NULL,
   Stato StatoValutazione,
   Risultato FLOAT DEFAULT 0,
   idTest BIGSERIAL NOT NULL,
   Studente BIGSERIAL NOT NULL,
   dataSostenuto TIMESTAMP NOT NULL,
   OrarioFine TIME,
   NumeroRCorrette INTEGER DEFAULT 0,
   NumeroRErrate INTEGER DEFAULT 0,
   PRIMARY KEY (idIstanzaDiTest),
   FOREIGN KEY (idTest) REFERENCES Test (idTest),
   ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (Studente) REFERENCES Utente (idUtente)
   ON DELETE CASCADE ON UPDATE CASCADE
);
```

ConsistenzaOrarioFine

```
ALTER TABLE IstanzaDiTest ADD CONSTRAINT ConsistenzaOrarioFine CHECK (datasostenuto
    ::TIME < OrarioFine);
```

4.1.6 RispostaUtente

Tabella RispostaUtente

```
CREATE TABLE RispostaUtente (
    idRispostaUtente  BIGSERIAL   NOT NULL,
    TestoRisposta    VARCHAR(256),
  punteggioRisposta FLOAT,
    idIstanzaDiTest BIGSERIAL NOT NULL,
    idDomanda BIGSERIAL NOT NULL,
    PRIMARY KEY (idRispostaUtente),
    FOREIGN KEY (idIstanzaDiTest) REFERENCES IstanzaDiTest (idIstanzaDiTest)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (idDomanda) REFERENCES Domanda (idDomanda)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

RispostaUtenteUnique

```
ALTER TABLE Rispostautente ADD UNIQUE(idistanzaditest, iddomanda);
```

4.2 Procedure automatizzate

Sono state individuare le seguenti tipologie di procedura atte alla gestione delle ridondanze, implementazione di vincoli e implementazione della logica funzionale

Link per la visione delle procedure individuate

4.2.1 Calcolo Ruolo automatizzato

```
--calcolo automatico del ruolo dell'utente a partire dall'email
CREATE FUNCTION CalcolaRuolo()
RETURNS TRIGGER
AS $body$
BEGIN
    IF NEW.Email LIKE '%@Studenti.Universita.it' THEN
      UPDATE Utente
        SET Ruolo = 'Studente'
    WHERE idUtente = NEW.idUtente;
  ELSE
      UPDATE Utente
        SET Ruolo = 'Insegnante'
    WHERE idUtente = NEW.idUtente;
  END IF;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER ruolo
AFTER INSERT ON Utente
FOR EACH ROW
EXECUTE PROCEDURE CalcolaRuolo();
```

4.2.2 Controllo registrazione test docenti

```
--un test pu  essere registrato solamente da un insegnante
CREATE FUNCTION ControlloDocenti()
RETURNS TRIGGER
AS $body$
DECLARE
    _Ruolo VARCHAR(10);
BEGIN
    SELECT Ruolo INTO _Ruolo
  FROM Utente
  WHERE NEW.Insegnante = idUtente;

  IF _Ruolo = 'Studente' THEN
      ROLLBACK;
  END IF;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER CheckDocenti
BEFORE INSERT ON Test
FOR EACH ROW
EXECUTE PROCEDURE ControlloDocenti();
```

4.2.3 Aggiornamento numDomande e MaxPunteggio INSERT

```
--Aggiorna il numDomande e il punteggio totale ogni volta che si inserisce una
    nuova domanda

CREATE FUNCTION AggiornaDatiTestInsert()
RETURNS TRIGGER
AS $body$
DECLARE
    PunteggioMassimoD Test.punteggiodmax%TYPE;
BEGIN

    SELECT punteggiodmax INTO PunteggioMassimoD
  FROM Test
  WHERE idTest = NEW.idTest;

    UPDATE Test
      SET numdomande = numdomande + 1, maxpunteggio = maxpunteggio +
    PunteggioMassimoD
  WHERE idTest = NEW.idTest;

  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER AggiornadatiTestInsert
AFTER INSERT ON Domanda
FOR EACH ROW
EXECUTE PROCEDURE AggiornaDatiTestInsert();
```

4.2.4 Aggiornamento numDomande e MaxPunteggio DELETE

```
--Aggiorna il numDomande e il punteggio totale ogni volta che si cancella una
    domanda

CREATE FUNCTION AggiornaDatiTestDelete()
RETURNS TRIGGER
AS $body$
DECLARE
    PunteggioMassimoD Test.punteggiodmax%TYPE;
BEGIN

    SELECT punteggiodmax INTO PunteggioMassimoD
  FROM Test
  WHERE idTest = OLD.idTest;

    UPDATE Test
      SET numdomande = numdomande - 1, maxpunteggio = maxpunteggio -
    PunteggioMassimoD
  WHERE idTest = OLD.idTest;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER AggiornaDatiTestDelete
AFTER DELETE ON Domanda
FOR EACH ROW
EXECUTE PROCEDURE AggiornaDatiTestDelete();
```

4.2.5 Consistenza Inserimenti Domanda

```
--Verifica che gli inserimenti di domande siano consistenti con la tipologia di quiz:
--Una domanda aperta non può referenziare un test a risposta multipla
--Una domanda chiusa non può referenziare un test a risposta aperta

CREATE FUNCTION CheckTest()
RETURNS TRIGGER
AS $body$
DECLARE
    _tipoQuiz Test.tipoQuiz%TYPE;
BEGIN
    SELECT TipoQuiz INTO _tipoQuiz
  FROM Test
  WHERE NEW.idTest = idTest;

  IF _tipoQuiz = 'Aperta' AND NEW.TipoDomanda = 'Multipla' THEN
      ROLLBACK;
  ELSE
      IF _tipoQuiz = 'Multipla' AND NEW.TipoDomanda = 'Aperta' THEN
          ROLLBACK;
      END IF;
  END IF;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER CheckTestD
BEFORE INSERT ON Domanda
FOR EACH ROW
EXECUTE PROCEDURE CheckTest();
```

4.2.6 Consistenza Inserimenti Risposta

```
--Verifica che solamente una domanda a risposta multipla abbia associato un set di risposte automatizzate

CREATE FUNCTION ControlloRisposteAutomatizzate()
RETURNS TRIGGER
AS $body$
DECLARE
    _TipoDomanda VARCHAR(10);
BEGIN
    SELECT TipoDomanda INTO _TipoDomanda
  FROM Domanda
  WHERE NEW.idDomanda = idDomanda;

  IF _TipoDomanda = 'Aperta' THEN
      ROLLBACK;
  END IF;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER IntegritaRisposteAutomatizzate
BEFORE INSERT ON Risposta
FOR EACH ROW
EXECUTE PROCEDURE ControlloRisposteAutomatizzate();
```

4.2.7 Controllo svolgimento Test

```
--Si assicura che un istanza di test possa essere aperta esclusivamente da uno
    studente

CREATE FUNCTION ControlloRisposteStudenti()
RETURNS TRIGGER
AS $body$
DECLARE
    _Ruolo VARCHAR(10);
BEGIN
    SELECT Ruolo INTO _Ruolo
  FROM Utente
  WHERE NEW.Studente = idUtente;

  IF _Ruolo = 'Insegnante' THEN
      ROLLBACK;
  END IF;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER CheckRisposteUtente
BEFORE INSERT ON IstanzaDiTest
FOR EACH ROW
EXECUTE PROCEDURE ControlloRisposteStudenti();
```

4.2.8 Consistenza OrarioFine

```
--Un test non pu  essere sostenuto in una data antecedente alla sua pubblicazione

CREATE FUNCTION CheckDataIstanzaTest()
RETURNS TRIGGER
AS $body$
DECLARE
    _dataS Test.data_ora%TYPE;
BEGIN
    SELECT data_ora INTO _dataS
  FROM Test
  WHERE NEW.idTest = idTest;

  IF _dataS > NEW.DataSostenuto THEN
      ROLLBACK;
  END IF;
  RETURN NEW;
END;
$body$ LANGUAGE plpgsql;

CREATE TRIGGER CheckDataIstanza
BEFORE INSERT ON IstanzaDiTest
FOR EACH ROW
EXECUTE PROCEDURE CheckDataIstanzaTest();
```

4.2.9 Consistenza risposteUtente a quesiti Multipli

```
--Una rispostaUtente fornita ad una domanda a risposta multipla deve rientrare in
    una delle risposte possibili

CREATE FUNCTION CheckConsistenzaRisposteUtente()
RETURNS TRIGGER
AS $body$
DECLARE
    risp CURSOR FOR
      SELECT testorisposta
    FROM risposta
    WHERE NEW.idDomanda = idDomanda;

  _risp Risposta.testoRisposta%TYPE;
  _flag INTEGER := 0;
  _tipoDomanda VARCHAR(10);
BEGIN
    SELECT TipoDomanda INTO _tipoDomanda
  FROM Domanda
  WHERE NEW.idDomanda = idDomanda;

  IF _tipoDomanda = 'Multipla' THEN
        OPEN risp;
          LOOP
            EXIT WHEN NOT FOUND OR _flag = 1;
          FETCH risp INTO _risp;
          IF _risp = NEW.TestoRisposta THEN
              _flag = 1;
          END IF;
        END LOOP;
      CLOSE risp;

      IF _flag = 0 THEN
          ROLLBACK;
    END IF;
  END IF;

  RETURN NEW;
END;

$body$ LANGUAGE plpgsql;
CREATE TRIGGER CheckRisposteUtente
BEFORE INSERT ON RispostaUtente
FOR EACH ROW
EXECUTE PROCEDURE CheckConsistenzaRisposteUtente();
```

4.2.10 Controllo rispostaUtente valida

```
--si assicura che l'attributo iddomanda sia consistente con idistanzaDiTest di test
    , cio  referenziano lo stesso test

CREATE FUNCTION ConsistenzaDomandeRispostaUtente()
RETURNS TRIGGER
AS $body$
DECLARE
    testDomanda Test.idTest%TYPE;
  testIstanzaDiTest Test.idTest%TYPE;
BEGIN
    SELECT idTest INTO testDomanda
  FROM Domanda
  WHERE idDomanda = NEW.idDomanda;

  SELECT idTest INTO testIstanzaDiTest
  FROM IstanzaDiTest
  WHERE idistanzaDiTest = NEW.idIstanzaDiTest;

  IF testDomanda <> testIstanzaDiTest THEN
      ROLLBACK;
    END IF;
  RETURN NEW;
```

```
END;

$body$ LANGUAGE plpgsql;

CREATE TRIGGER ConsistenzaRisposteUtente
BEFORE INSERT ON rispostaUtente
FOR EACH ROW
EXECUTE PROCEDURE ConsistenzaDomandeRispostaUtente();
```

4.2.11 Controllo punteggio assegnato a risposta

```
--Si assicura che il voto assegnanto ad una risposta rientri tra il minimo e
    massimo consentito e aggiorna risultati

CREATE FUNCTION CorrezioneDomandaRispostaAperta()
RETURNS TRIGGER
AS $body$
DECLARE
    tipoD Domanda.tipodomanda%TYPE;
  punteggioMin Test.punteggioDmin%TYPE;
  punteggioMax Test.punteggioDmax%TYPE;
  OrarioF IstanzaDiTest.OrarioFine%TYPE;
BEGIN
    SELECT tipoDomanda INTO tipoD
  FROM Domanda
  WHERE idDomanda = NEW.idDomanda;

  SELECT punteggioDmin, punteggioDmax INTO punteggioMin, PunteggioMax
  FROM Test T, Domanda D
  WHERE T.idTest = D.idTest AND D.idDomanda = NEW.idDomanda;

  SELECT OrarioFine INTO OrarioF
  FROM IstanzaDiTest
  WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;

  IF OrarioF IS NULL THEN
      ROLLBACK;
  END IF;

  IF tipoD <> 'Multipla' THEN
      IF NEW.punteggiorisposta > PunteggioMax OR NEW.punteggiorisposta <
    punteggioMin THEN
          ROLLBACK;
      ELSE IF NEW.punteggiorisposta = PunteggioMax THEN
        UPDATE IstanzaDiTest
          SET numerorcorrette = numerorcorrette + 1
      WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
    ELSE
        UPDATE IstanzaDiTest
          SET numerorerrate = numerorerrate + 1
      WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
      END IF;
    END IF;
  END IF;

  UPDATE ISTANZADITEST
        SET risultato = risultato + NEW.punteggiorisposta
  WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;

  RETURN NEW;
END;

$body$ LANGUAGE plpgsql;

CREATE TRIGGER correzioneDAperte
BEFORE UPDATE OF punteggiorisposta ON rispostaUtente
FOR EACH ROW
EXECUTE PROCEDURE CorrezioneDomandaRispostaAperta();
```

4.2.12 Calcolo punteggio risposte multiple

```
--Calcola il punteggio totalizzato sui quesiti a risposta multipla

CREATE FUNCTION CalcoloPunteggioRispostaChiusa()
RETURNS TRIGGER
AS $body$
DECLARE
    Risposte CURSOR FOR
 SELECT U.TestoRisposta rispUtente, R.TestoRisposta rispcorretta, D.iddomanda dom
 FROM rispostautente U, domanda D, risposta R
 WHERE U.iddomanda = D.iddomanda AND D.iddomanda = R.iddomanda AND U.
    idistanzaDiTest = NEW.idIstanzaDiTest AND D.tipoDomanda = 'Multipla' AND R.
    Corretta = 'true';

  punteggioMin Test.punteggioDmin%TYPE;
  punteggioMax Test.punteggioDmax%TYPE;
  Quiz Test.tipoQuiz%TYPE;
  domandeCorrette INTEGER := 0;
  domandeErrate INTEGER := 0;

BEGIN

    SELECT punteggioDmin, punteggioDmax, TipoQuiz INTO punteggioMin, PunteggioMax,
    Quiz
  FROM Test
  WHERE idTest = NEW.idTest;

  FOR I IN Risposte
  LOOP
      IF I.rispUtente = I.rispCorretta THEN
        UPDATE RispostaUtente
      SET punteggioRisposta = punteggioMax
            WHERE idIstanzaDiTest = NEW.idIstanzaDiTest AND idDomanda = I.dom;
      domandeCorrette := domandeCorrette + 1;

      ELSE
        UPDATE RispostaUtente
      SET punteggioRisposta = punteggioMin
        WHERE idIstanzaDiTest = NEW.idIstanzaDiTest AND idDomanda = I.dom;
      domandeErrate := domandeErrate + 1;
    END IF;
  END LOOP;

  UPDATE ISTANZADITEST
      SET numerorcorrette = domandeCorrette, numerorerrate = domandeErrate
  WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;

  IF Quiz = 'Multipla' THEN
      UPDATE ISTANZADITEST
        SET Stato = 'Valutato'
    WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
  END IF;

  RETURN NEW;
END;

$body$ LANGUAGE plpgsql;

CREATE TRIGGER CompletamentoTest
AFTER UPDATE OF OrarioFine ON ISTANZADITEST
FOR EACH ROW
EXECUTE PROCEDURE CalcoloPunteggioRispostaChiusa();
```

Link per la visione del Popolamento del database