



# 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

<b>1</b>	<b>Analisi del problema e requisiti identificati</b>	<b>5</b>
<b>2</b>	<b>Progettazione concettuale</b>	<b>6</b>
2.1	Class Diagram . . . . .	6
2.1.1	Alcune precisazioni sul class Diagram [Attributo Email] . . . .	6
2.1.2	Alcune precisazioni sul class Diagram [Attributo Risposte] . .	7
2.2	Analisi della ristrutturazione del Class Diagram . . . . .	7
2.2.1	Analisi delle ridondanze . . . . .	7
2.2.2	Analisi degli identificativi . . . . .	7
2.2.3	Rimozione degli attributi multipli . . . . .	7
2.2.4	Rimozione degli attributi composti . . . . .	8
2.2.5	Partizione/Accorpamento delle associazioni . . . . .	8
2.2.6	Rimozione delle gerarchie . . . . .	8
2.3	Class Diagram ristrutturato . . . . .	9
2.4	Dizionario delle classi . . . . .	10
2.5	Dizionario dei vincoli . . . . .	12
2.6	Dizionario delle associazioni . . . . .	14
2.7	Varianti del class Diagram . . . . .	15
2.7.1	Associazione tra rispostaUtente e risposta . . . . .	15
2.7.2	Associazione tra rispostaUtente e Studente a sostituzione dell'associazione RispostaStudente tra rispostaUtente e Istanza-DiTest . . . . .	15
<b>3</b>	<b>Progettazione Logica</b>	<b>16</b>
<b>4</b>	<b>Progettazione Fisica</b>	<b>17</b>
4.1	Definizione delle tabelle e relativi vincoli di dominio e vincoli intra-relazionali . . . . .	17
4.1.1	Utente . . . . .	17
4.1.2	Test . . . . .	18
4.1.3	Domanda . . . . .	18
4.1.4	Risposta . . . . .	18
4.1.5	IstanzaDiTest . . . . .	19
4.1.6	RispostaUtente . . . . .	19
4.2	Procedure automatizzate . . . . .	20
4.2.1	Calcolo Ruolo automatizzato . . . . .	20
4.2.2	Controllo registrazione test docenti . . . . .	20
4.2.3	Aggiornamento numDomande e MaxPunteggio INSERT . . .	21
4.2.4	Aggiornamento numDomande e MaxPunteggio DELETE . . .	21

4.2.5	Consistenza Inserimenti Domanda . . . . .	22
4.2.6	Consistenza Inserimenti Risposta . . . . .	22
4.2.7	Controllo svolgimento Test . . . . .	23
4.2.8	Consistenza OrarioFine . . . . .	23
4.2.9	Consistenza risposteUtente a quesiti Multipli . . . . .	24
4.2.10	Controllo rispostaUtente valida . . . . .	24
4.2.11	Controllo punteggio assegnato a risposta . . . . .	25
4.2.12	Calcolo punteggio risposte multiple . . . . .	26

# Capitolo 1

## Analisi del problema e requisiti identificati

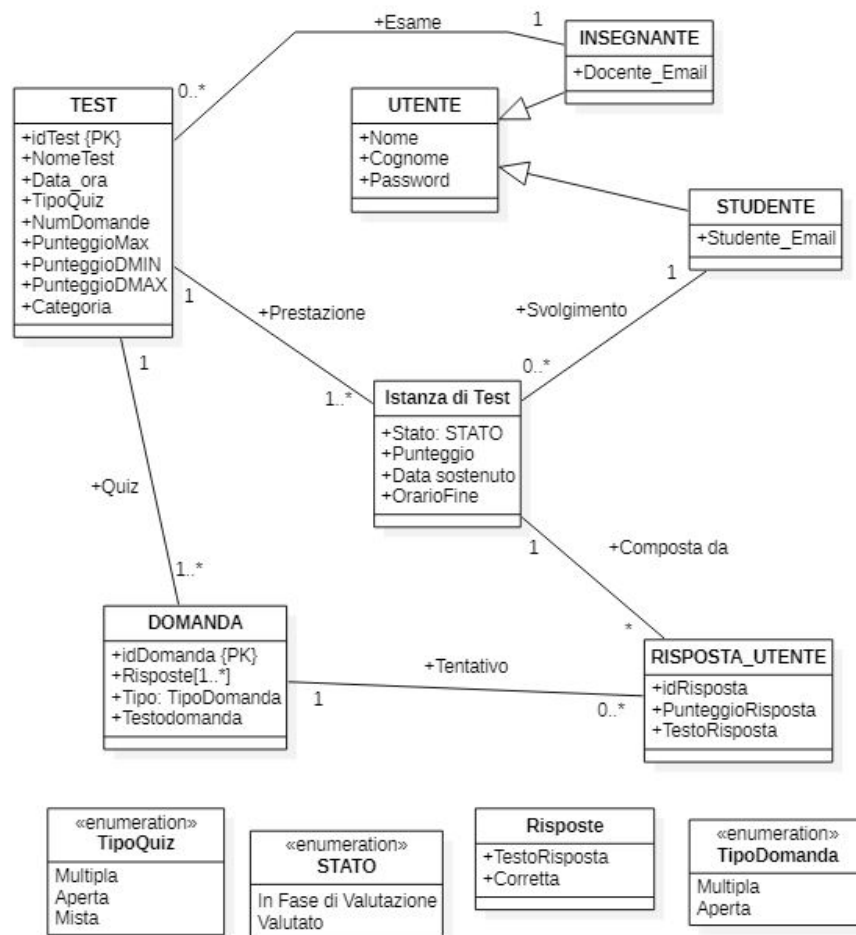
Si progetterà ed implementerà una base di dati relazionale adibita alla gestione di dati relativi ad un applicazione di E-learning tramite quiz. Dalla realtà di riferimento si evincono i seguenti tipi di entità fondamentali:

- **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 insegnanti 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 è 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 risposte Utente fornite dallo studente.
- **risposta utente** 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 mantenerli

#### 2.2.5 Partizione/Accorpamento delle associazioni

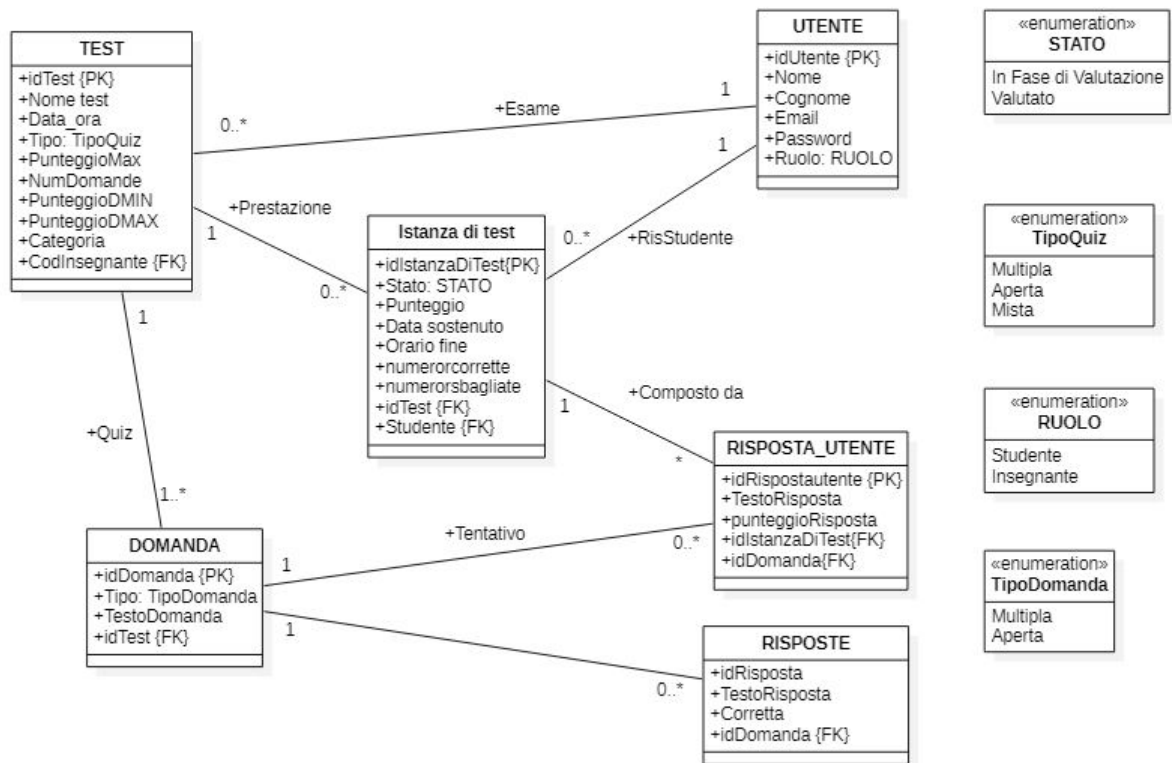
Non vi sono presenti associazione da accorpare nel diagramma.

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

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

<b>Vincolo</b>	<b>Classi coinvolte</b>	<b>Descrizione</b>
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 automatizzata associata alla domanda specifica
Consistenza RisposteUtente	RispostaUtente	Per ogni istanza di test, è possibile fornire una sola risposta per domanda
Consistenza Domande RisposteUtente	RispostaUtente, IstanzaDiTest	Test, 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	<b>Utente</b> [1] ruolo: indica quale insegnante crea il test. <b>Test</b> [0..*] ruolo: indica quale test è stato creato dall'insegnante.
Quiz	Descrive da quali domande è composto ogni test	<b>Test</b> [1] ruolo: indica a quale test fanno riferimento le domande. <b>Domanda</b> [1..*] ruolo: indica da quali domande è composto ogni test.
Prestazione	Descrive l'esito di un istanza di test	<b>Test</b> [1] ruolo: indica a quale test è riferito l'esito. <b>IstanzaDiTest</b> [] ruolo: indica l'esito ottenuto nel test.
RisStudente	Descrive i risultati degli studenti	<b>Utente</b> [1] ruolo: indica a quale studente fanno riferimento gli esiti. <b>IstanzaDiTest</b> [0..*] ruolo: indica i risultati ottenuti dagli studenti.
RispostaStudente	Indica le risposte fornite dagli studenti alle domande in una determinata istanza di test	<b>IstanzaDiTest</b> [1] ruolo: indica quale studente fornisce la risposta. <b>RispostaUtente</b> [0..*] ruolo: indica le risposte fornite dagli studenti.
Tentativo	Indica le risposte fornite dagli studenti alle domande	<b>RispostaUtente</b> [0..*] ruolo: indica le risposte fornite alle domande. <b>Domanda</b> [1] ruolo: indica le domande al quale fanno riferimento le risposte.
PossibiliSoluzioni	Indica le possibili soluzioni alle domande (solo domande a risposta multipla)	<b>Risposte</b> [0..*] ruolo: indica le possibili risposte alle domande. <b>Domanda</b> [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 referencia 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 procedura 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

<b>Utente</b>	( <u>idUtente</u> , Nome, Cognome, Email, Password, Ruolo)
<b>Test</b>	( <u>idTest</u> , nomeTest, DataOra, tipoQuiz, MaxPunteggio, NumDomande, Categoria, PunteggioDMin, PunteggioDMax, <u>Insegnante</u> ) <b>Insegnante</b> $\mapsto$ <b>Utente</b> (idUtente)
<b>Domanda</b>	( <u>idDomanda</u> , tipoDomanda, testoDomanda, <u>idTest</u> ) <b>idTest</b> $\mapsto$ <b>Test</b> (idTest)
<b>Risposta</b>	(idRisposta, testoRisposta, Corretta, <u>idDomanda</u> ) <b>idDomanda</b> $\mapsto$ <b>Domanda</b> (idDomanda)
<b>IstanzaDiTest</b>	( <u>idIstanzaDiTest</u> , Stato, Risultato, DataSostenuto, OrarioFine, numeroRcorrette, numeroRSbagliate, <u>idTest</u> , <u>Studente</u> ) <b>idTest</b> $\mapsto$ <b>Test</b> (idTest) <b>Studente</b> $\mapsto$ <b>Utente</b> (idUtente)
<b>RispostaUtente</b>	( <u>idrispostaUtente</u> , testoRisposta, punteggioRisposta, <u>idIstanzaDiTest</u> , <u>idDomanda</u> ) <b>idIstanzaDiTest</b> $\mapsto$ <b>idIstanzaDiTest</b> (idIstanzaDiTest) <b>idDomanda</b> $\mapsto$ <b>Domanda</b> (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 PostgreSQL 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

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

##### Enumerazione Ruolo

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

##### Tabella Utente

```
1 CREATE TABLE Utente
2 (
3     idUtente BIGSERIAL NOT NULL,
4     Nome VARCHAR(32) NOT NULL,
5     Cognome VARCHAR(32) NOT NULL,
6     Email FormatoEmail NOT NULL,
7     Password VARCHAR(32) NOT NULL,
8     Ruolo ruolo,
9     PRIMARY KEY(idUtente)
10 );
```

##### Unicità Email

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

##### Consistenza Ruolo

```
1 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

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

#### Dominio punteggioDmax

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

#### Dominio punteggioDmin

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

#### Tabella Test

```
1 CREATE TABLE Test
2 (
3     idTest BIGSERIAL NOT NULL,
4     nomeTest VARCHAR(32),
5     Data_Ora TIMESTAMP NOT NULL,
6     TipoQuiz tipoQuiz NOT NULL,
7     MaxPunteggio FLOAT DEFAULT 0,
8     NumDomande INTEGER DEFAULT 0,
9     PunteggioDMax punteggioDmax NOT NULL DEFAULT 0,
10    PunteggioDMin punteggioDmin DEFAULT 0,
11    Categoria VARCHAR(32) NOT NULL,
12    Insegnante BIGSERIAL NOT NULL,
13    PRIMARY KEY (idTest),
14    FOREIGN KEY (Insegnante) REFERENCES Utente (idUtente)
15    ON DELETE CASCADE ON UPDATE CASCADE
16 );
```

### 4.1.3 Domanda

#### Enumerazione tipoDomanda

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

#### Tabella Domanda

```
1 CREATE TABLE Domanda
2 (
3     idDomanda BIGSERIAL NOT NULL,
4     TipoDomanda tipoDomanda NOT NULL,
5     TestoDomanda VARCHAR(256) NOT NULL,
6     idTest BIGSERIAL NOT NULL,
7     PRIMARY KEY (idDomanda),
8     FOREIGN KEY (idTest) REFERENCES test (idTest)
9     ON DELETE CASCADE ON UPDATE CASCADE
10 );
```

### 4.1.4 Risposta

#### Tabella Risposta

```
1 CREATE TABLE Risposta (
2     idRisposta BIGSERIAL NOT NULL,
3     idDomanda BIGSERIAL NOT NULL,
4     TestoRisposta VARCHAR(512),
5     Corretta BOOLEAN,
6     PRIMARY KEY (idRisposta),
7     FOREIGN KEY (idDomanda) REFERENCES Domanda (idDomanda)
8     ON DELETE CASCADE ON UPDATE CASCADE
9 );
```

### 4.1.5 IstanzaDiTest

#### Enumerazione StatoValutazione

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

#### Tabella IstanzaDiTest

```
1 CREATE TABLE IstanzaDiTest
2 (
3     idIstanzaDiTest BIGSERIAL NOT NULL,
4     Stato StatoValutazione,
5     Risultato FLOAT DEFAULT 0,
6     idTest BIGSERIAL NOT NULL,
7     Studente BIGSERIAL NOT NULL,
8     dataSostenuto TIMESTAMP NOT NULL,
9     OrarioFine TIME,
10    NumeroRCorrette INTEGER DEFAULT 0,
11    NumeroErrate INTEGER DEFAULT 0,
12    PRIMARY KEY (idIstanzaDiTest),
13    FOREIGN KEY (idTest) REFERENCES Test (idTest),
14    ON DELETE CASCADE ON UPDATE CASCADE,
15    FOREIGN KEY (Studente) REFERENCES Utente (idUtente)
16    ON DELETE CASCADE ON UPDATE CASCADE
17 );
```

#### ConsistenzaOrarioFine

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

### 4.1.6 RispostaUtente

#### Tabella RispostaUtente

```
1 CREATE TABLE RispostaUtente (
2     idRispostaUtente BIGSERIAL NOT NULL,
3     TestoRisposta VARCHAR(256),
4     punteggioRisposta FLOAT,
5     idIstanzaDiTest BIGSERIAL NOT NULL,
6     idDomanda BIGSERIAL NOT NULL,
7     PRIMARY KEY (idRispostaUtente),
8     FOREIGN KEY (idIstanzaDiTest) REFERENCES IstanzaDiTest (idIstanzaDiTest)
9     ON DELETE CASCADE ON UPDATE CASCADE,
10    FOREIGN KEY (idDomanda) REFERENCES Domanda (idDomanda)
11    ON DELETE CASCADE ON UPDATE CASCADE
12 );
```

#### RispostaUtenteUnique

```
1 ALTER TABLE RispostaUtente ADD UNIQUE(idistanzaditest, iddomanda);
```

## 4.2 Procedure automatizzate

Sono state individuate 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

```
1 --calcolo automatico del ruolo dell'utente a partire dall'email
2 CREATE FUNCTION CalcolaRuolo()
3 RETURNS TRIGGER
4 AS $body$
5 BEGIN
6     IF NEW.Email LIKE '%@Studenti.Universita.it' THEN
7         UPDATE Utente
8             SET Ruolo = 'Studente'
9             WHERE idUtente = NEW.idUtente;
10    ELSE
11        UPDATE Utente
12            SET Ruolo = 'Insegnante'
13            WHERE idUtente = NEW.idUtente;
14    END IF;
15    RETURN NEW;
16 END;
17 $body$ LANGUAGE plpgsql;
18
19 CREATE TRIGGER ruolo
20 AFTER INSERT ON Utente
21 FOR EACH ROW
22 EXECUTE PROCEDURE CalcolaRuolo();
```

### 4.2.2 Controllo registrazione test docenti

```
1 --un test pu essere registrato solamente da un insegnante
2 CREATE FUNCTION ControlloDocenti()
3 RETURNS TRIGGER
4 AS $body$
5 DECLARE
6     _Ruolo VARCHAR(10);
7 BEGIN
8     SELECT Ruolo INTO _Ruolo
9     FROM Utente
10    WHERE NEW.Insegnante = idUtente;
11
12    IF _Ruolo = 'Studente' THEN
13        ROLLBACK;
14    END IF;
15    RETURN NEW;
16 END;
17 $body$ LANGUAGE plpgsql;
18
19 CREATE TRIGGER CheckDocenti
20 BEFORE INSERT ON Test
21 FOR EACH ROW
22 EXECUTE PROCEDURE ControlloDocenti();
```

### 4.2.3 Aggiornamento numDomande e MaxPunteggio INSERT

```
1  --Aggiorna il numDomande e il punteggio totale ogni volta che si inserisce una
   nuova domanda
2
3  CREATE FUNCTION AggiornaDatiTestInsert()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      PunteggioMassimoD Test.punteggiomax%TYPE;
8  BEGIN
9
10     SELECT punteggiomax INTO PunteggioMassimoD
11     FROM Test
12     WHERE idTest = NEW.idTest;
13
14     UPDATE Test
15     SET numdomande = numdomande + 1, maxpunteggio = maxpunteggio +
        PunteggioMassimoD
16     WHERE idTest = NEW.idTest;
17
18     RETURN NEW;
19 END;
20 $body$ LANGUAGE plpgsql;
21
22 CREATE TRIGGER AggiornadatiTestInsert
23 AFTER INSERT ON Domanda
24 FOR EACH ROW
25 EXECUTE PROCEDURE AggiornaDatiTestInsert();
```

### 4.2.4 Aggiornamento numDomande e MaxPunteggio DELETE

```
1  --Aggiorna il numDomande e il punteggio totale ogni volta che si cancella una
   domanda
2
3  CREATE FUNCTION AggiornaDatiTestDelete()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      PunteggioMassimoD Test.punteggiomax%TYPE;
8  BEGIN
9
10     SELECT punteggiomax INTO PunteggioMassimoD
11     FROM Test
12     WHERE idTest = OLD.idTest;
13
14     UPDATE Test
15     SET numdomande = numdomande - 1, maxpunteggio = maxpunteggio -
        PunteggioMassimoD
16     WHERE idTest = OLD.idTest;
17     RETURN NEW;
18 END;
19 $body$ LANGUAGE plpgsql;
20
21 CREATE TRIGGER AggiornaDatiTestDelete
22 AFTER DELETE ON Domanda
23 FOR EACH ROW
24 EXECUTE PROCEDURE AggiornaDatiTestDelete();
```

## 4.2.5 Consistenza Inserimenti Domanda

```
1  --Verifica che gli inserimenti di domande siano consistenti con la tipologia di
   quiz:
2  --Una domanda aperta non pu  referenziare un test a risposta multipla
3  --Una domanda chiusa non pu  referenziare un test a risposta aperta
4
5  CREATE FUNCTION CheckTest()
6  RETURNS TRIGGER
7  AS $body$
8  DECLARE
9      _tipoQuiz Test.tipoQuiz%TYPE;
10 BEGIN
11     SELECT TipoQuiz INTO _tipoQuiz
12     FROM Test
13     WHERE NEW.idTest = idTest;
14
15     IF _tipoQuiz = 'Aperta' AND NEW.TipoDomanda = 'Multipla' THEN
16         ROLLBACK;
17     ELSE
18         IF _tipoQuiz = 'Multipla' AND NEW.TipoDomanda = 'Aperta' THEN
19             ROLLBACK;
20         END IF;
21     END IF;
22     RETURN NEW;
23 END;
24 $body$ LANGUAGE plpgsql;
25
26 CREATE TRIGGER CheckTestD
27 BEFORE INSERT ON Domanda
28 FOR EACH ROW
29 EXECUTE PROCEDURE CheckTest();
```

## 4.2.6 Consistenza Inserimenti Risposta

```
1  --Verifica che solamente una domanda a risposta multipla abbia associato un set di
   risposte automatizzate
2
3  CREATE FUNCTION ControllorRisposteAutomatizzate()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      _TipoDomanda VARCHAR(10);
8  BEGIN
9      SELECT TipoDomanda INTO _TipoDomanda
10     FROM Domanda
11     WHERE NEW.idDomanda = idDomanda;
12
13     IF _TipoDomanda = 'Aperta' THEN
14         ROLLBACK;
15     END IF;
16     RETURN NEW;
17 END;
18 $body$ LANGUAGE plpgsql;
19
20 CREATE TRIGGER IntegritaRisposteAutomatizzate
21 BEFORE INSERT ON Risposta
22 FOR EACH ROW
23 EXECUTE PROCEDURE ControllorRisposteAutomatizzate();
```

## 4.2.7 Controllo svolgimento Test

```
1  --Si assicura che un istanza di test possa essere aperta esclusivamente da uno
   studente
2
3  CREATE FUNCTION ControlloRisposteStudenti()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      _Ruolo VARCHAR(10);
8  BEGIN
9      SELECT Ruolo INTO _Ruolo
10     FROM Utente
11     WHERE NEW.Studente = idUtente;
12
13     IF _Ruolo = 'Insegnante' THEN
14         ROLLBACK;
15     END IF;
16     RETURN NEW;
17 END;
18 $body$ LANGUAGE plpgsql;
19
20 CREATE TRIGGER CheckRisposteUtente
21 BEFORE INSERT ON IstanzaDiTest
22 FOR EACH ROW
23 EXECUTE PROCEDURE ControlloRisposteStudenti();
```

## 4.2.8 Consistenza OrarioFine

```
1  --Un test non pu essere sostenuto in una data antecedente alla sua pubblicazione
2
3  CREATE FUNCTION CheckDataIstanzaTest()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      _dataS Test.data_ora%TYPE;
8  BEGIN
9      SELECT data_ora INTO _dataS
10     FROM Test
11     WHERE NEW.idTest = idTest;
12
13     IF _dataS > NEW.DataSostenuto THEN
14         ROLLBACK;
15     END IF;
16     RETURN NEW;
17 END;
18 $body$ LANGUAGE plpgsql;
19
20 CREATE TRIGGER CheckDataIstanza
21 BEFORE INSERT ON IstanzaDiTest
22 FOR EACH ROW
23 EXECUTE PROCEDURE CheckDataIstanzaTest();
```

### 4.2.9 Consistenza risposteUtente a quesiti Multipli

```
1  --Una rispostaUtente fornita ad una domanda a risposta multipla deve rientrare in
   una delle risposte possibili
2
3  CREATE FUNCTION CheckConsistenzaRisposteUtente()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      risp CURSOR FOR
8          SELECT testorisposta
9          FROM risposta
10         WHERE NEW.idDomanda = idDomanda;
11
12     _risp Risposta.testoRisposta%TYPE;
13     _flag INTEGER := 0;
14     _tipoDomanda VARCHAR(10);
15 BEGIN
16     SELECT TipoDomanda INTO _tipoDomanda
17     FROM Domanda
18     WHERE NEW.idDomanda = idDomanda;
19
20     IF _tipoDomanda = 'Multipla' THEN
21         OPEN risp;
22         LOOP
23             EXIT WHEN NOT FOUND OR _flag = 1;
24             FETCH risp INTO _risp;
25             IF _risp = NEW.TestoRisposta THEN
26                 _flag = 1;
27             END IF;
28         END LOOP;
29         CLOSE risp;
30
31         IF _flag = 0 THEN
32             ROLLBACK;
33         END IF;
34     END IF;
35
36     RETURN NEW;
37 END;
38
39 $body$ LANGUAGE plpgsql;
40 CREATE TRIGGER CheckRisposteUtente
41 BEFORE INSERT ON RispostaUtente
42 FOR EACH ROW
43 EXECUTE PROCEDURE CheckConsistenzaRisposteUtente();
```

### 4.2.10 Controllo rispostaUtente valida

```
1  --si assicura che l'attributo iddomanda sia consistente con idistanzaDiTest di test
   , cio  riferenziano lo stesso test
2
3  CREATE FUNCTION ConsistenzaDomandeRispostaUtente()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      testDomanda Test.idTest%TYPE;
8      testIstanzaDiTest Test.idTest%TYPE;
9  BEGIN
10     SELECT idTest INTO testDomanda
11     FROM Domanda
12     WHERE idDomanda = NEW.idDomanda;
13
14     SELECT idTest INTO testIstanzaDiTest
15     FROM IstanzaDiTest
16     WHERE idistanzaDiTest = NEW.idIstanzaDiTest;
17
18     IF testDomanda <> testIstanzaDiTest THEN
19         ROLLBACK;
20     END IF;
21     RETURN NEW;
```



```

22 END;
23
24 $body$ LANGUAGE plpgsql;
25
26 CREATE TRIGGER ConsistenzaRisposteUtente
27 BEFORE INSERT ON rispostaUtente
28 FOR EACH ROW
29 EXECUTE PROCEDURE ConsistenzaDomandeRispostaUtente();

```

#### 4.2.11 Controllo punteggio assegnato a risposta

```

1  --Si assicura che il voto assegnando ad una risposta rientri tra il minimo e
   massimo consentito e aggiorna risultati
2
3  CREATE FUNCTION CorrezioneDomandaRispostaAperta()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      tipoD Domanda.tipodomanda%TYPE;
8      punteggioMin Test.punteggioDmin%TYPE;
9      punteggioMax Test.punteggioDmax%TYPE;
10     OrarioF IstanzaDiTest.OrarioFine%TYPE;
11 BEGIN
12     SELECT tipoDomanda INTO tipoD
13     FROM Domanda
14     WHERE idDomanda = NEW.idDomanda;
15
16     SELECT punteggioDmin, punteggioDmax INTO punteggioMin, PunteggioMax
17     FROM Test T, Domanda D
18     WHERE T.idTest = D.idTest AND D.idDomanda = NEW.idDomanda;
19
20     SELECT OrarioFine INTO OrarioF
21     FROM IstanzaDiTest
22     WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
23
24     IF OrarioF IS NULL THEN
25         ROLLBACK;
26     END IF;
27
28     IF tipoD <> 'Multipla' THEN
29         IF NEW.punteggiorisposta > PunteggioMax OR NEW.punteggiorisposta <
           punteggioMin THEN
30             ROLLBACK;
31         ELSE IF NEW.punteggiorisposta = PunteggioMax THEN
32             UPDATE IstanzaDiTest
33             SET numerorcorrette = numerorcorrette + 1
34             WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
35         ELSE
36             UPDATE IstanzaDiTest
37             SET numerorerrate = numerorerrate + 1
38             WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
39         END IF;
40     END IF;
41 END IF;
42
43 UPDATE ISTANZADITEST
44 SET risultato = risultato + NEW.punteggiorisposta
45 WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
46
47 RETURN NEW;
48 END;
49
50 $body$ LANGUAGE plpgsql;
51
52 CREATE TRIGGER correzioneDAperte
53 BEFORE UPDATE OF punteggiorisposta ON rispostaUtente
54 FOR EACH ROW
55 EXECUTE PROCEDURE CorrezioneDomandaRispostaAperta();

```

## 4.2.12 Calcolo punteggio risposte multiple

```
1  --Calcola il punteggio totalizzato sui quesiti a risposta multipla
2
3  CREATE FUNCTION CalcoloPunteggioRispostaChiusa()
4  RETURNS TRIGGER
5  AS $body$
6  DECLARE
7      Risposte CURSOR FOR
8      SELECT U.TestoRisposta rispUtente, R.TestoRisposta rispCorretta, D.iddomanda dom
9      FROM rispostautente U, domanda D, risposta R
10     WHERE U.iddomanda = D.iddomanda AND D.iddomanda = R.iddomanda AND U.
        idistanzaDiTest = NEW.idIstanzaDiTest AND D.tipoDomanda = 'Multipla' AND R.
        Corretta = 'true';
11
12     punteggioMin Test.punteggioDmin%TYPE;
13     punteggioMax Test.punteggioDmax%TYPE;
14     Quiz Test.tipoQuiz%TYPE;
15     domandeCorrette INTEGER := 0;
16     domandeErrate INTEGER := 0;
17
18 BEGIN
19
20     SELECT punteggioDmin, punteggioDmax, TipoQuiz INTO punteggioMin, PunteggioMax,
        Quiz
21     FROM Test
22     WHERE idTest = NEW.idTest;
23
24     FOR I IN Risposte
25     LOOP
26         IF I.rispUtente = I.rispCorretta THEN
27             UPDATE RispostaUtente
28             SET punteggioRisposta = punteggioMax
29             WHERE idIstanzaDiTest = NEW.idIstanzaDiTest AND idDomanda = I.dom;
30             domandeCorrette := domandeCorrette + 1;
31
32         ELSE
33             UPDATE RispostaUtente
34             SET punteggioRisposta = punteggioMin
35             WHERE idIstanzaDiTest = NEW.idIstanzaDiTest AND idDomanda = I.dom;
36             domandeErrate := domandeErrate + 1;
37         END IF;
38     END LOOP;
39
40     UPDATE ISTANZADITEST
41     SET numerorcorrette = domandeCorrette, numerorerrate = domandeErrate
42     WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
43
44     IF Quiz = 'Multipla' THEN
45         UPDATE ISTANZADITEST
46         SET Stato = 'Valutato'
47         WHERE idIstanzaDiTest = NEW.idIstanzaDiTest;
48     END IF;
49
50     RETURN NEW;
51 END;
52
53 $body$ LANGUAGE plpgsql;
54
55 CREATE TRIGGER CompletamentoTest
56 AFTER UPDATE OF OrarioFine ON ISTANZADITEST
57 FOR EACH ROW
58 EXECUTE PROCEDURE CalcoloPunteggioRispostaChiusa();
```

[Link per la visione del Popolamento del database](#)