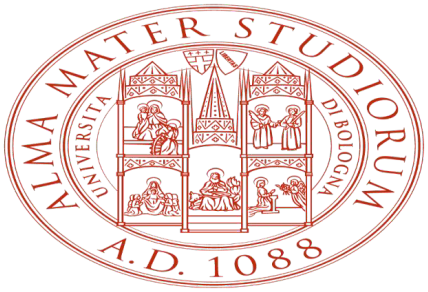
Relazione Progetto Base di dati
2023/2024



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Componeti:

Nome: Carlos Vasco Chironda

Matricola: 0000995

Data: 15/11/2024

RACCOLTA E ANALISI DEI REQUISITI

Premessa

Si vuole realizzare la piattaforma ESQL per supportare la didattica del corso di dati, e, nello specifico, del modulo di programmazione SQL. La piattaforma è liberamente ispirata al tool Moodle SQL utilizzato nelle esercitazioni in aula e contenuto nel tool Virtuale. La piattaforma consente ai docenti di creare delle tabelle del modello relazionale, e di popolarne il contenuto. Inoltre, consente di creare test con domande a risposta chiusa o con sketch di codice. La piattaforma consente la registrazione da parte di studenti; ogni studente può visionare i test disponibili, completare un test, visionare l'esito o inviare messaggi al docente.

SPECIFICA DELLA PIATTAFORMA

La piattaforma ESQL si basa sul database relazionale ESQLDB.

Tutti gli utenti della piattaforma dispongono di: email, nome, cognome, eventuale recapito telefonico. Gli utenti sono divisi in due tipologie: docenti e studenti. I primi dispongono anche di: nome del dipartimento di afferenza e nome del corso di cui sono titolari. I secondi dispongono anche di un campo anno di immatricolazione e di un codice alfanumerico di lunghezza pari a 16 caratteri. I docenti possono creare delle tabelle SQL (definite tabelle di esercizio, nel seguito): ogni tabella di esercizio dispone di nome, data di creazione, un campo num_righe. Inoltre, ogni tabella di esercizio dispone di un insieme di attributi: ogni attributo dispone di un nome, un tipo, e può far parte della chiave primaria della tabella di esercizio. Devono poter essere inseriti dai docenti anche i vincoli di integrità tra attributi di diverse tabelle di esercizio. In aggiunta, ogni docente può creare dei test: ogni test dispone di un titolo univoco, una data di creazione ed un'eventuale foto. Ogni test può includere una serie di quesiti: ogni quesito dispone di un numero progressivo (univoco, ma solo all'interno di uno specifico test), un livello di difficoltà (campo enum con valori: Basso, Medio, Alto), un campo descrizione, un campo numrisposte (ridondanza concettuale, vedere specifiche sotto) e fa riferimento ad una o più tabelle di esercizio tra quelle create dal docente. I quesiti possono appartenere esclusivamente a due categorie: quesiti a risposta chiusa o quesiti di codice. Nel primo caso, il quesito dispone di una serie di opzioni di risposta: ogni opzione dispone di una numerazione (univoca, ma solo all'interno di uno specifico quesito) ed un campo testo. Nel secondo caso, il quesito dispone di una o più soluzioni (sketch di codice SQL che implementano correttamente quanto richiesto dalla descrizione del quesito). Ogni test dispone di un campo booleano `VisualizzaRisposte`: se settato a True, le risposte dei quesiti diventano visibili agli studenti, altrimenti restano nascoste. Gli studenti possono svolgere un test, inserendo una o più risposte per ciascun quesito. Si vuole tenere traccia del completamento del test, ossia: data di inserimento della prima risposta (su scala temporale), data di inserimento dell'ultima risposta (su scala temporale), stato (campo enum con tre valori:

Aperto, InCompletamento, Concluso). Nel caso di quesiti a risposta chiusa, la risposta consiste nell'opzione scelta tra quelle disponibili. Nel caso di quesiti di codice, la risposta consiste in un campo testo (codice SQL che risolve l'esercizio). È prevista la possibilità per lo studente di sottomettere più risposte per lo stesso quesito, in istanti diversi di tempo, ma solo se il test non è stato Concluso. Ogni risposta dispone di un campo esito, che può valere True o False a seconda che la risposta fornita dallo studente coincida con quella inserita dal docente (nel caso di quesiti a risposta chiusa) o che la risposta fornita dallo studente produca lo stesso output di quella inserita dal docente (nel caso di quesiti di codice). Infine, è prevista la possibilità di inviare messaggi nella piattaforma. Ogni messaggio dispone di un titolo, un campo testo, una data di inserimento, e fa riferimento ad uno specifico test. Il messaggio può essere inviato da un docente: in tal caso, il messaggio viene ricevuto da tutti gli studenti. Viceversa, un messaggio può essere inviato da uno studente: in tal caso, il destinatario è uno specifico docente.

Lista Operazioni

Operazioni che riguardano tutti gli utenti

- Autenticazione/registrazione sulla piattaforma
- Visualizzazione dei test disponibili
- Visualizzazione dei quesiti presenti all'interno di ciascun test

Operazioni che riguardano SOLO i docenti

- Inserimento di una nuova tabella di esercizio, con relativi meta-dati
- Inserimento di una riga per una tabella di esercizio definita dal docente.
- Creazione di nuovo test
- Creazione di un nuovo quesito con le relative risposte
- Abilitare / disabilitare la visualizzazione delle risposte per uno specifico test
- Inserimento di un messaggio

Operazioni che riguardano SOLO gli studenti

- Inserimento di una nuova risposta ad un quesito
- Visualizzazione dell'esito della risposta
- Inserimento di un messaggio

Statistiche (visibili da tutti gli utenti)

- Visualizzare la classifica degli studenti, sulla base del numero di test completati (un test si considera completato se il suo stato è pari a "Concluso"). Nella classifica NON devono apparire i dati sensibili dello studente (nome, cognome, email) ma solo il codice alfanumerico.
- Visualizzare la classifica degli studenti, sulla base del numero di risposte corrette inserite rispetto al numero totale di risposte inserite. Nella classifica NON devono apparire i dati sensibili dello studente (nome, cognome, email) ma solo il codice alfanumerico.
- Visualizzare la classifica dei quesiti, in base al numero di risposte inserite dagli studenti.

GLOSSARIO DEI DATI

Si valuta che il testo sia abbastanza chiaro: non risultano usi di sinonimi complessi, da richiedere la creazione di una tabella apposita per i sinonimi (glossario dei dati).

Progettazione Concettuale (Diagramma E-R)

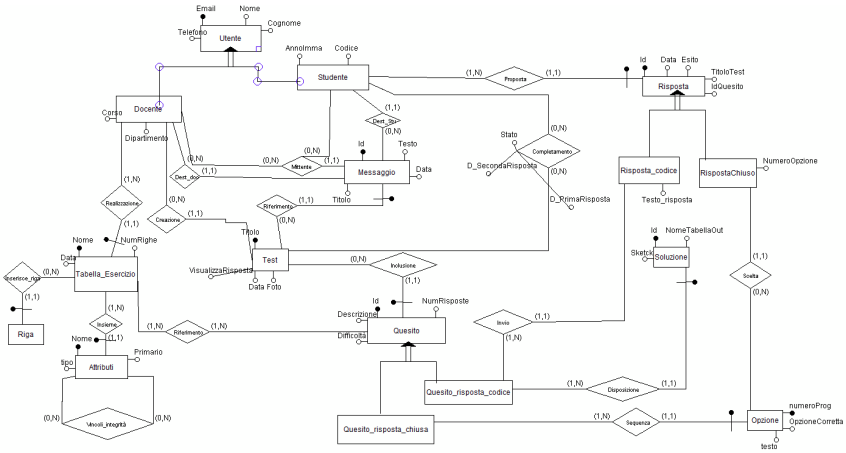


Figure 1: Diagramma Concettuale.

DIZIONARIO DELLE ENTITÀ

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
UTENTE	Utente che interagisce con la piattaforma	Email, Nome, Telefono, Cognome	Email
DOCENTE	Tipo di utente che può creare tabelle, popolarle, creare test e quesiti, inviare messaggi	Dipartimento, Corso	Email
STUDENTE	Tipo di utente che può visionare e completare i test, visionare l'esito o inviare messaggi al docente	Codice, AnnoImmatricolazione	Email
TABELLA DI ESERCIZIO	Tabella SQL creata dal docente	Nome, Data, num_righe	Nome
ATTRIBUTO	Attributo della tabella SQL	Nome, Primaria, Tipo	Nome, Tabella di Esercizio
RIGA	Riga della tabella SQL	Valori	Valori, Tabella di Esercizio
QUESITO	Quesito posto dal docente	ID, Difficoltà, numRisposte, Descrizione	ID, Test
QUESITO A RISPOSTA CHIUSA	Quesito che prevede una serie di opzioni di risposta		ID
QUESITO DI CODICE	Quesito che prevede una o più soluzioni (sketch di codice SQL)		ID
OPZIONE	Opzione di risposta di un quesito a risposta chiusa	Numerazione, Testo, Opzione corretta	Numerazione, Quesito a Risposta Chiusa
SOLUZIONE	Soluzione (sketch di codice SQL) di un quesito di codice	ID, Sketch, Nome tabella output	ID
RISPOSTA	Risposta proposta da uno studente per un quesito	Data, Esito, Id	Id, STUDENTE
RISPOSTA CODICE	Risposta proposta da uno studente per un quesito di codice	Testo risposta	Id, STUDENTE
RISPOSTA CHIUSA	Risposta proposta da uno studente per un quesito a risposta chiusa	Data	Id, STUDENTE
TEST	Test creato dal docente che comprende un certo numero di quesiti	Titolo, Data, Foto, Visualizza Risposte	Titolo
MESSAGGIO	Messaggio scambiato tra docente e studente/i	Data, Titolo	Data, Titolo, TEST

Table 1: Dizionario delle entità utilizzate nel progetto.

DIZIONARIO DELLE RELAZIONI

RELAZIONE	DESCRIZIONE	COMPONENTI	ATTRIBUTI
REALIZZAZIONE	Realizzazione di una tabella da parte del docente	Docente, Tabella Esercizio	
CREAZIONE	Creazione di una Test da parte di un Docente	Docente, Test	
RIFERIMENTO	Un test può fare riferimento a più quesiti, e ogni quesito può essere incluso in più test	Test, Quesito	
INCLUSIONE	Un Test può includere dei quesiti chiusa.	Quesito, Test	
COMPLETAMENTE	Svolgimento di un test da parte di un studente	Studente, Test	Stato,DataPrimaRisposta, maRisposta
INSIEME	Insieme di attributi di una Tabella di Esercizio	Attributi, Tabella esercizio	
PROPOSTA	Proposta di una Risposta da parte di uno Studente	Studente, Risposta	
INSERISCE_RIGA	inserisce una riga ad una tabella esercizio	Riga, Tabella Esercizio	
INVIO	Invio di una risposta ad un quesito codice	RispostaCodice, QuesitoCodice	
SEQUENZA	Sequenza di opzione per un quesito chiusto	Opzione, QuesitoChiuso	
DISPOSIZIONE	Un quesito codice dispone di una soluzione	QuesitoCodice, Soluzione	
CITAZIONE	Un Messaggio fa citazione ad un test	Test, Messaggio	
MITTENTE	Mittente di un messaggio	Docente, Studente, Messaggio	
DES_DOCENTE	Destinatario Docente di un Messaggio	Docente,Messaggio	
DES_Studente	Destinatario Studente di un Messaggio	Studente,Messaggio	

Table 2: Dizionario delle relazioni utilizzate nel progetto.

TAVOLA DELLE BUSINESS RULES

ID	REGOLA DI VINCOLO
1	Il codice alfanumerico di Studente è di 16 caratteri.
2	Il livello di difficoltà del Quesito può essere 'Basso', 'Medio', 'Alto'.
3	Il campo VisualizzaRisposte è un campo booleano: se settato a true le risposte diventano visibili agli studenti, altrimenti restano nascoste.
4	Uno studente può inserire una o più risposte per ciascun quesito in istanti diversi di tempo se il Test non è 'Concluso'.
5	La Data di inserimento della prima Risposta deve essere anteriore o uguale alla Data di inserimento dell'ultima Risposta.
6	Lo stato di svolgimento di un test per uno studente può essere 'Aperto', 'InCompletamento', 'Concluso'.
7	Il campo esito di Risposta può essere true o false a seconda che la risposta inserita dallo studente coincida con quella inserita dal docente (in caso di quesiti a risposta chiusa) o produca lo stesso output di quella inserita dal docente (in caso di quesiti di codice).
8	Nel caso in cui il messaggio sia inviato dal docente, viene ricevuto da tutti gli studenti.
9	Nel caso in cui sia inviato da uno studente, il destinatario è uno specifico docente.
10	Il docente, nell'inserimento della risposta corretta per il quesito di codice che crea, non può inserire uno sketch di codice sbagliato (es. Non sono previsti problemi di sintassi o riferimenti a tabelle inesistenti).
11	I codici alfanumerici degli studenti non possono essere ripetuti.

Table 3: Tavola delle Business Rules.

PROGETTAZIONE LOGICA (Schema E-R Ristrutturato)

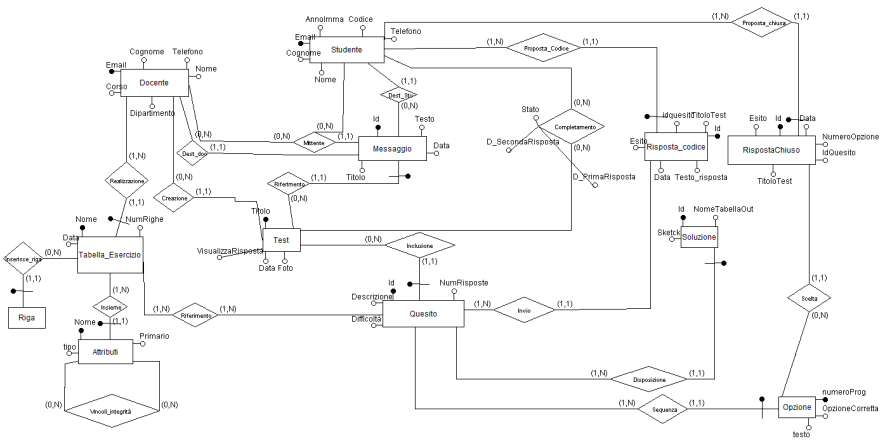


Figure 2: Diagramma Logico ristrutturato.

Lista delle tabelle con vincoli di chiave

- **Studente** (Email, Nome, Cognome, Codice, Telefono, AnnoImmatricolazione)
- **Docente** (Email, Nome, Cognome, Telefono, NomeDipartimento, NomeCorso)
- **Tabella_Esercizio** (Nome, Data, NumeroRighe, EmailDocente)
- **Attributi** (Nome, NomeTabella, Tipo, Primario)
- **Integrità_Referenziale** (Id, NomeAttributo1, NomeAttributo2, Tabella1, Tabella2)
- **Riferimento** (NomeTabella_Esercizio, IdQuesito, TitoloTest)
- **Quesito** (Id, TitoloTest, NumRisposta, Difficoltà, Descrizione, TipoQuesito)
- **Opzione** (Numerazione, IdQuesito, TitoloTest, OpzioneCorretta, Testo)
- **RispostaCodice** (Id, EmailStudente, Esito, Data, TestoRisposta, IdQuesito)
- **RispostaChiusa** (Id, EmailStudente, Data, Esito, NumeroOpzione, NumerazioneOpzione)

- **Soluzione** (Id; Sketch, IdQuesito, TitoloTest, NomeTabellaOutPut)
- **Riga** (NomeTabella, Valori)
- **Test** (Titolo, Foto, Data, VisualizzazioneRisposta, EmailDocente)
- **Completamento** (TitoloTest, EmailDocente, DataPrimaRisposta, DataUltimaRisposta, Stato)
- **Messaggio** (Titolo, Data, TitoloTesto, Testo, EmailStudenteMittente, EmailDocenteMittente, EmailDocenteDestinatario)
- **DestinatarioStudente** (EmailStudente, TitoloMessaggio, TitoloTest, Data)

Vincoli Inter-relazionali

- **Tabella_Esercizio.EmailDocente → Docente.Email**
- **Attributo.Nome → Tabella_Esercizio.Nome**
- **Integrità_Referenziale.NomeAttributo1 → Attributo.Nome**
- **Integrità_Referenziale.NomeAttributo2 → Attributo.Nome**
- **Integrità_Referenziale.Tabella1 → Tabella_Esercizio.Nome**
- **Integrità_Referenziale.Tabella2 → Tabella_Esercizio.Nome**
- **Riferimento.NomeTabella_Esercizio → Tabella_Esercizio.Nome**
- **Riferimento.IdQuesito → Quesito.Id**
- **Riferimento.TitoloTest → Quesito.TitoloTest**
- **Quesito.TitoloTest → Test.Titolo**
- **Opzione.IdQuesito → Quesito.Id**
- **Opzione.TitoloTest → Quesito.TitoloTest**
- **RispostaCodice.EmailStudente → Studente.Email**
- **RispostaCodice.IdQuesito → Quesito.Id**
- **RispostaChiusa.EmailStudente → Studente.Email**
- **RispostaChiusa.NumerazioneOpzione → Opzione.Numerazione**
- **Soluzione.IdQuesito → Quesito.Id**
- **Soluzione.TitoloTest → Quesito.TitoloTest**
- **Riga.NomeTabella → Tabella_Esercizio.Nome**

- **Test.EmailDocente → Docente.Email**
- **Completamento.TitoloTest → Test.Titolo**
- **Completamento.EmailDocente → Docente.Email**
- **Messaggio.TitoloTest → Test.Titolo**
- **Messaggio.EmailDocenteMittente → Docente.Email**
- **Messaggio.EmailDocenteDestinatario → Docente.Email**
- **Messaggio.EmailStudenteMittente → Studente.Email**
- **DestinatarioStudente.EmailStudente → Studente.Email**
- **DestinatarioStudente.TitoloMessaggio → Messaggio.Titolo**
- **DestinatarioStudente.TitoloTest → Test.Titolo**

Analisi delle Ridondanze

Pesatura delle Operazioni

- **Peso dell'operazione batch:** $W_B = 0.5$
- **Peso dell'operazione interattiva:** $W_I = 1$
- **Peso delle operazioni di scrittura:** $\alpha = 2$

Formula Generale

$$c(OT) = f(OT) \cdot w_T \cdot (\alpha \cdot NC_{write} + NC_{read})$$

Dove:

- $c(OT)$: Costo dell'operazione OT
- $f(OT)$: Frequenza dell'operazione OT
- w_T: Peso dell'operazione (batch o interattiva)
- α: Peso delle operazioni di scrittura
- NC_{write}: Numero di operazioni di scrittura
- NC_{read}: Numero di operazioni di lettura

Tavola dei Volumi

Concetto	Tipo	Volume
Quesito	Entità	20
Risposta	Entità	200
Utente	Entità	50
Appartenenza	Relazione	200

Table 4: Volumi delle entità e relazioni.

Tavola delle Operazioni

Operazione	Tipo	Frequenza
Operazione 1: Aggiungere una nuova risposta a un quesito esistente	Interattiva	10 volte/mese
Operazione 2: Rimuovere un quesito e tutte le sue risposte	Batch	2 volte/mese
Operazione 3: Visualizzare tutti gli utenti presenti nella piattaforma	Batch	1 volta/mese
Operazione 4: Contare il numero di risposte per ciascun quesito nella piattaforma	Interattiva	2 volte/mese

Table 5: Frequenze e tipologia delle operazioni.

Analisi delle Ridondanze

Calcolo senza ridondanze

Operazione 1: Aggiungere una nuova risposta ad un quesito esistente

Risposta codice:

- 1 accesso in lettura alla tabella `QUESITO` per identificare il quesito.
- 1 accesso in scrittura alla tabella `INVIO`.
- 1 accesso in scrittura alla tabella `Risposta_codice` per aggiungere la risposta.

$$C_1 = 10 \cdot 1 \cdot [2 \cdot 2 + 1] = 10 \cdot [5] = 50$$

Risposta Chiusa:

- 1 accesso in lettura alla tabella `QUESITO` per identificare il quesito.
- 1 accesso in scrittura alla tabella `Scelta` per inserire la scelta.

- 1 accesso in lettura alla tabella `Opzione` per visualizzare l'opzione del quesito.
- 1 accesso in scrittura alla tabella `Risposta_chiusa` per scrivere la risposta.

$$C_1 = 10 \cdot 1 \cdot [2 \cdot 2 + (1 + 1)] = 10 \cdot [6] = 60$$

Operazione 2: Rimuovere un quesito e tutte le risposte

Risposta codice:

- 1 accesso in lettura alla tabella `QUESITO` per recuperare il quesito.
- 1 accesso in scrittura per eliminare il quesito.
- 10 accessi in scrittura alla tabella `Risposta_Codice` per eliminare le 10 risposte associate.
- 10 accessi alla tabella `INVIO` per eliminare le 10 relazioni tra quesito e invio.

$$C_2 = 2 \cdot 0.5 \cdot [2 \cdot (1 + 10 + 10) + 1] = 43$$

Risposta Chiusa:

- 1 accesso in lettura alla tabella `QUESITO` per recuperare il quesito.
- 1 accesso in scrittura per eliminare il quesito.
- 10 accessi in scrittura alla tabella `Risposta_Chiusa` per eliminare le 10 risposte associate.
- 10 accessi in scrittura alla tabella `Scelta` per eliminare le 10 relazioni tra quesito e scelta.
- 10 accessi alla tabella `Opzione` per eliminare le 10 opzioni associate.
- 1 accesso in `Sequenza` per eliminare la relazione tra sequenza e opzione.

$$C_2 = 2 \cdot 0.5 \cdot [2 \cdot (1 + 1 + 10 \cdot 3) + 1] = 65$$

Operazione 3: Visualizzare tutti gli utenti presenti nella piattaforma

- 1 accesso in lettura per ogni utente nella tabella `STUDENTE` (25 studenti).
- 1 accesso in lettura per ogni utente nella tabella `DOCENTE` (25 docenti).

$$C_3 = 1 \cdot 0.5 \cdot [2 \cdot 0 + 50] = 25$$

Operazione 4: Contare il numero di risposte per ciascun quesito

Risposta codice:

- 1 accesso in lettura alla tabella `QUESITO` per ogni quesito (20 quesiti).
- 1 accesso in lettura alla relazione `INVIO` per ciascun quesito (200 accessi).

$$C_4 = 2 \cdot 1 \cdot [2 \cdot 0 + (20 + 200)] = 440$$

Risposta Chiusa:

- 1 accesso in lettura alla tabella `QUESITO` per ogni quesito (20 quesiti).
- 1 accesso alla relazione `Scelta` per ciascuna risposta.
- 1 accesso alla tabella `Opzione` per ciascuna risposta.

Per ogni Quesito di tipo chiuso:

- Ci sono 20 quesiti e ognuno ha 10 risposte.
- Ogni risposta di tipo chiuso richiede l'accesso a Scelta e Opzione. Quindi:
- Accessi alla tabella Quesito: 20.
- Accessi alla relazione Scelta: 200.
- Accessi alla tabella Opzione: 200.

$$C_4 = 2 \cdot 1 \cdot [2 \cdot 0 + (20 + 200 + 200)] = 840$$

Totale senza ridondanze: 1514

Calcoli con ridondanza

Operazione 1: Aggiungere una nuova risposta a un quesito esistente

Risposta codice:

- 1 accesso in lettura alla tabella `QUESITO`.
- 1 accesso in scrittura per inserire la nuova riga nella tabella `Risposta_Codice`.
- 1 accesso in scrittura alla tabella `QUESITO` per aggiornare il campo #num-Risposte.
- 1 accesso in scrittura alla relazione `INVIO`.

$$C_1 = 10 \cdot 1 \cdot [2 \cdot (1 + 1 + 1) + 1] = 10 \cdot [6 + 1] = 70$$

Risposta chiusa:

- 1 accesso in lettura alla tabella `QUESITO`.
- 1 accesso in scrittura per inserire la nuova riga nella tabella `Risposta_Chiusa`.
- 1 accesso in scrittura alla tabella `QUESITO` per aggiornare il campo #num-Risposte.
- 1 accesso in scrittura alla tabella `Scelta` per inserire la scelta.
- 1 accesso in lettura alla tabella `Opzione` per visualizzare l'opzione.
- 1 accesso in lettura alla tabella `Sequenza`.

$$C_1 = 10 \cdot 1 \cdot [2 \cdot (1 + 1 + 1) + 3] = 10 \cdot [6 + 1] = 90$$

Operazione 2: Rimuovere un quesito e tutte le sue risposte

Risposta codice:

- 1 accesso in lettura alla tabella `QUESITO`.
- 1 accesso in scrittura per eliminare il quesito dalla tabella `QUESITO`.
- 10 accessi in scrittura per eliminare le risposte associate dalla tabella `Risposta_Codice`.
- 10 accessi in scrittura per eliminare le relazioni dalla tabella `INVIO`.

$$C_2 = 2 \cdot 0.5 \cdot [2 \cdot (10 + 10 + 1) + 1] = 43$$

Risposta chiusa:

- 1 accesso in lettura alla tabella `QUESITO`.
- 1 accesso in scrittura per eliminare il quesito dalla tabella `QUESITO`.
- 10 accessi in scrittura per eliminare le risposte associate dalla tabella `Risposta_Chiusa`.
- 10 accessi in scrittura per eliminare le relazioni dalla tabella `Scelta`.
- 1 accesso in scrittura alla tabella `Opzione`.
- 1 accesso in scrittura alla tabella `Sequenza`.

$$C_2 = 2 \cdot 0.5 \cdot [2 \cdot (10 + 10 + 1 + 1 + 1) + 1] = 47$$

Operazione 3: Visualizzare tutti gli utenti presenti nella piattaforma

- 1 accesso in lettura per ogni utente nella tabella `STUDENTE` (25 studenti).
- 1 accesso in lettura per ogni utente nella tabella `DOCENTE` (25 docenti).

$$C_3 = 1 \cdot 0.5 \cdot [2 \cdot 0 + 50] = 25$$

Operazione 4: Contare il numero di risposte per ciascun quesito

Risposta codice:

- 1 accesso in lettura alla tabella `QUESITO` per leggere il campo #numRisposte, per ogni quesito (20 quesiti).

$$C_4 = 2 \cdot 1 \cdot [2 \cdot 0 + 20] = 40$$

Risposta chiusa:

- 1 accesso in lettura alla tabella `QUESITO` per leggere il campo #numRisposte, per ogni quesito (20 quesiti).

$$C_4 = 2 \cdot 1 \cdot [2 \cdot 0 + 20] = 40$$

Totale con ridondanze: 355

Speed-Up e Conclusioni

$$Speed - Up : \frac{c(S)}{c(S_{rid})} = \frac{1514}{355} = 4.26$$

$$m(S_{rid}) = X + 4\,Byte \cdot 20 = X + 80\,Byte$$

Conclusione: Le operazioni senza ridondanza richiedono un costo significativamente maggiore rispetto a quelle con ridondanza, con un rapporto di 4.26. Ciò significa che le operazioni con ridondanza sono più efficienti rispetto a quelle senza. Considerando lo speed-up e l'overhead generato dalla ridondanza, si ritiene accettabile un overhead di soli 80 byte, che consente un aumento di velocità superiore al doppio.

Considerando tutti i fattori e i valori ottenuti, si conclude che il campo #numRisposte va mantenuto.

Normalizzazione

Non si ritiene necessaria.

Appendice: Script SQL

```
-- Tabella Studente
CREATE TABLE Studente (
    Email VARCHAR(255) PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Cognome VARCHAR(100) NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Codice CHAR(16) UNIQUE NOT NULL,
    Telefono VARCHAR(20),
    AnnoImmatricolazione INT
);

-- Tabella Docente
CREATE TABLE Docente (
    Email VARCHAR(255) PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Cognome VARCHAR(100) NOT NULL,
    Telefono VARCHAR(20),
    Password VARCHAR(255) NOT NULL,
    NomeDipartimento VARCHAR(100),
    NomeCorso VARCHAR(100)
);
-- Tabella Tabella_Esercizio
CREATE TABLE Tabella_Esercizio (
     Nome VARCHAR(100) PRIMARY KEY,
    Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    NumeroRighe INT DEFAULT 0,
    EmailDocente VARCHAR(255),
    FOREIGN KEY (EmailDocente) REFERENCES Docente(Email) ON DELETE CASCADE
);




-- Tabella Attributi
CREATE TABLE Attributi (
     Nome VARCHAR(100),
     NomeTabella VARCHAR(100),
     Tipo VARCHAR(50),
     Primario BOOLEAN,
    PRIMARY KEY (Nome, NomeTabella),
    FOREIGN KEY (NomeTabella) REFERENCES Tabella_Esercizio(Nome) ON DELETE CASCADE
);
-- Tabella Integrità_Referenziale
CREATE TABLE Integrita_Referenziale (
```

```
    Id INT AUTO_INCREMENT PRIMARY KEY,
    NomeAttributo1 VARCHAR(100),
    NomeAttributo2 VARCHAR(100),
    Tabella1 VARCHAR(100),
    Tabella2 VARCHAR(100),
    FOREIGN KEY (NomeAttributo1, Tabella1) REFERENCES
    Attributi(Nome, NomeTabella) ON DELETE CASCADE,
    FOREIGN KEY (NomeAttributo2, Tabella2)
    REFERENCES Attributi(Nome, NomeTabella) ON DELETE CASCADE
);
CREATE TABLE RIGA (
    NomeTabella VARCHAR(100),
    Valori TEXT,
    PRIMARY KEY(NomeTabella, Valori(255)),
    FOREIGN KEY (NomeTabella) REFERENCES Tabella_Esercizio(Nome) ON DELETE CASCADE
);
-- Tabella TestValutaRisposta.html
CREATE TABLE Test (
     Titolo VARCHAR(255) PRIMARY KEY,
    Foto MEDIUMBLOB,
     Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    VisualizzazioneRisposta BOOLEAN DEFAULT FALSE,
     EmailDocente VARCHAR(255),
    FOREIGN KEY (EmailDocente) REFERENCES Docente(Email) ON DELETE CASCADE
);

-- Tabella Quesito
CREATE TABLE Quesito (
     Id INT AUTO_INCREMENT PRIMARY KEY,
    TitoloTest VARCHAR(255),
    NumRisposta INT,
    Difficoltà ENUM('Basso', 'Medio', 'Alto'),
    Descrizione TEXT,
     TipoQuesito ENUM('RispostaChiusa', 'Codice'),
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE
);

-- Tabella Riferimento
CREATE TABLE Riferimento (
     NomeTabellaEsercizio VARCHAR(100),
    IdQuesito INT,
    TitoloTest VARCHAR(255),
    PRIMARY KEY (NomeTabellaEsercizio, IdQuesito, TitoloTest),
    FOREIGN KEY (NomeTabellaEsercizio) REFERENCES Tabella_Esercizio(Nome) ON DELETE CASCADE,
    FOREIGN KEY (IdQuesito) REFERENCES Quesito(Id) ON DELETE CASCADE,
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE
```

```
);


-- Tabella Soluzione
CREATE TABLE Soluzione (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Sketch TEXT,
    IdQuesito INT,
    TitoloTest VARCHAR(255),
    NomeTabellaOutput VARCHAR(100),
    FOREIGN KEY (IdQuesito) REFERENCES Quesito(Id) ON DELETE CASCADE,
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE
);
-- Tabella Opzione
CREATE TABLE Opzione (
    Numerazione INT,
     IdQuesito INT,
     TitoloTest VARCHAR(255),
    OpzioneCorretta BOOLEAN,
    Testo TEXT,
    PRIMARY KEY (Numerazione, IdQuesito, TitoloTest),
    FOREIGN KEY (IdQuesito) REFERENCES Quesito(Id) ON DELETE CASCADE,
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE
);
-- Tabella RispostaCodice
CREATE TABLE RispostaCodice (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    EmailStudente VARCHAR(255),
    Esito BOOLEAN,
    Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TestoRisposta TEXT,
    IdQuesito INT,
    TitoloTest VARCHAR(255),
    FOREIGN KEY (EmailStudente) REFERENCES Studente(Email) ON DELETE CASCADE,
    FOREIGN KEY (IdQuesito) REFERENCES Quesito(Id) ON DELETE CASCADE,
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE
);
-- Tabella RispostaChiusa
CREATE TABLE RispostaChiusa (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    EmailStudente VARCHAR(255),
    Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
     Esito BOOLEAN,
     NumeroOpzione INT,
    IdQuesito INT,
    TitoloTest VARCHAR(255),
```

```
    FOREIGN KEY (EmailStudente) REFERENCES Studente(Email) ON DELETE CASCADE,
    -- FOREIGN KEY (TitoloTest) REFERENCES TestValutaRisposta.html(Titolo) ON DELETE CASCADE
    FOREIGN KEY (NumeroOpzione, IdQuesito, TitoloTest) REFERENCES
    Opzione(Numerazione, IdQuesito, TitoloTest) ON DELETE CASCADE
);



-- Tabella Messaggio
CREATE TABLE Messaggio (
    Titolo VARCHAR(255),
     Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TitoloTest VARCHAR(255),
     Testo TEXT,
    EmailStudenteMittente VARCHAR(255),
    EmailDocenteMittente VARCHAR(255),
    EmailDocenteDestinatario VARCHAR(255),
    PRIMARY KEY (Titolo, Data, TitoloTest),
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE,
    FOREIGN KEY (EmailStudenteMittente) REFERENCES Studente(Email) ON DELETE CASCADE,
    FOREIGN KEY (EmailDocenteMittente) REFERENCES Docente(Email) ON DELETE CASCADE,
    FOREIGN KEY (EmailDocenteDestinatario) REFERENCES Docente(Email) ON DELETE CASCADE
);

-- Tabella Destinatario
CREATE TABLE Destinatario (
    EmailStudente VARCHAR(255) NOT NULL,
    TitoloMessaggio VARCHAR(255) NOT NULL,
    TitoloTest VARCHAR(255) NOT NULL,
    Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (EmailStudente, TitoloMessaggio, TitoloTest, Data),
    FOREIGN KEY (EmailStudente) REFERENCES Studente(Email) ON DELETE CASCADE,
    FOREIGN KEY (TitoloMessaggio) REFERENCES Messaggio(Titolo) ON DELETE CASCADE,
    FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE
);

CREATE TABLE Completamento (
    TitoloTest VARCHAR(255),
    EmailStudente  VARCHAR(255),
    DataPrimaRisposta TIMESTAMP,
    DataUltimaRisposta TIMESTAMP,
    Stato ENUM('Aperto', 'InCompletamento', 'Concluso') DEFAULT 'Aperto',
     PRIMARY KEY (TitoloTest, EmailStudente),
     FOREIGN KEY (TitoloTest) REFERENCES Test(Titolo) ON DELETE CASCADE,
    FOREIGN KEY (EmailStudente) REFERENCES Studente(Email) ON DELETE CASCADE
);
```

```
--- Le procedure-----
DELIMITER //

CREATE PROCEDURE RegistraUtente(
    IN email VARCHAR(255),
    IN nome VARCHAR(100),
    IN cognome VARCHAR(100),
    IN password VARCHAR(255), -- Password aggiunta
    IN codice CHAR(16),
    IN telefono VARCHAR(20),
    IN annoImmatricolazione INT,
    IN tipoUtente ENUM('Studente', 'Docente'),
    IN nomeDipartimento VARCHAR(100),
    IN nomeCorso VARCHAR(100)
)
BEGIN
    IF tipoUtente = 'Studente' THEN
        INSERT INTO Studente (Email, Nome, Cognome, Password, Codice, Telefono,
        AnnoImmatricolazione)
        VALUES (email, nome, cognome, password, codice, telefono, annoImmatricolazione);
    ELSEIF tipoUtente = 'Docente' THEN
        INSERT INTO Docente (Email, Nome, Cognome, Password, Telefono, NomeDipartimento, Nom
        VALUES (email, nome, cognome, password, telefono, nomeDipartimento, nomeCorso);
    END IF;
END //

DELIMITER ;


-- Procedure per la creazione di una tabella
DELIMITER $$
-- Creazione tabella esercizio

DELIMITER //

CREATE PROCEDURE CreaTabellaEsercizio(
    IN p_Nome VARCHAR(100),
    IN p_EmailDocente VARCHAR(255)
)
BEGIN
    DECLARE numero_righe INT DEFAULT 0;

    -- Controlla se la tabella di esercizio esiste già
    IF EXISTS (SELECT 1 FROM Tabella_Esercizio WHERE Nome = p_Nome) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La tabella di esercizio esiste già.';
```

```
    ELSE
        -- Crea una nuova tabella di esercizio
        INSERT INTO Tabella_Esercizio (Nome, EmailDocente, NumeroRighe )
        VALUES (p_Nome, p_EmailDocente, numero_righe);
    END IF;
END //

DELIMITER ;
 -- creazione attributi tabella esercizi
DELIMITER //

CREATE PROCEDURE AggiungiAttributo(
    IN p_NomeAttributo VARCHAR(100),
    IN p_NomeTabella VARCHAR(100),
    IN p_Tipo VARCHAR(50),
    IN p_Primario BOOLEAN
)
BEGIN
    -- Controlla se l'attributo già esiste per la tabella specificata
    IF EXISTS (SELECT 1 FROM Attributi
    WHERE Nome = p_NomeAttributo AND NomeTabella = p_NomeTabella) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'L\'attributo esiste già nella tabella sp
    ELSE
        -- Aggiungi l'attributo alla tabella
        INSERT INTO Attributi (Nome, NomeTabella, Tipo, Primario)
        VALUES (p_NomeAttributo, p_NomeTabella, p_Tipo, p_Primario);

        -- Se l'attributo è primario, incrementa il numero delle righe della tabella eserciz
        IF p_Primario THEN
            UPDATE Tabella_Esercizio
            SET NumeroRighe = NumeroRighe + 1
            WHERE Nome = p_NomeTabella;
        END IF;
    END IF;
END //

DELIMITER ;


DELIMITER //
-- procedure per l'aggiunta dei vincoli di integrità nella tabella creata
CREATE PROCEDURE AggiungiVincoloIntegrita(
    IN p_NomeAttributo1 VARCHAR(100),
    IN p_NomeAttributo2 VARCHAR(100),
    IN p_Tabella1 VARCHAR(100),
    IN p_Tabella2 VARCHAR(100)
```

```
)
BEGIN
    -- Controlla se i vincoli di integrità già esistono
    IF EXISTS (SELECT 1 FROM Integrita_Referenziale
               WHERE NomeAttributo1 = p_NomeAttributo1 AND Tabella1 = p_Tabella1
                 AND NomeAttributo2 = p_NomeAttributo2 AND Tabella2 = p_Tabella2) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il vincolo di integrità esiste già.';
    ELSE
        -- Aggiungi il vincolo di integrità referenziale
        INSERT INTO Integrita_Referenziale (NomeAttributo1, NomeAttributo2, Tabella1, Tabell
        VALUES (p_NomeAttributo1, p_NomeAttributo2, p_Tabella1, p_Tabella2);
    END IF;
END //

DELIMITER ;

DELIMITER //
CREATE PROCEDURE CreaEsercizio (
    IN p_nome VARCHAR(100),
    IN p_emailDocente VARCHAR(255)
)
BEGIN
    DECLARE esiste INT;

    -- Controlla se esiste già un esercizio con lo stesso nome
    SELECT COUNT(*) INTO esiste FROM Tabella_Esercizio WHERE Nome = p_nome;

    IF esiste > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Esercizio già esistente.';
    ELSE
        INSERT INTO Tabella_Esercizio (Nome, EmailDocente)
        VALUES (p_nome, p_emailDocente);
    END IF;
END //



DELIMITER ;
DELIMITER //
CREATE PROCEDURE CreaTest(
    IN titolo VARCHAR(255),
    IN foto BLOB,
    IN emailDocente VARCHAR(255),
    OUT result INT
)
BEGIN
```

```
    -- Gestore per errori di duplicazione
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        BEGIN
            SET result = 0; -- Indica errore di duplicazione o altro errore SQL
            ROLLBACK;
        END;

    START TRANSACTION;

    -- Inserisci il nuovo test senza controlli
    IF foto IS NULL THEN
        INSERT INTO Test (Titolo, Foto, Data, VisualizzazioneRisposta, EmailDocente)
        VALUES (titolo, NULL, CURRENT_TIMESTAMP, FALSE, emailDocente);
    ELSE
        INSERT INTO Test (Titolo, Foto, Data, VisualizzazioneRisposta, EmailDocente)
        VALUES (titolo, foto, CURRENT_TIMESTAMP, FALSE, emailDocente);
    END IF;

    -- Se l'inserimento è riuscito, imposta il risultato a 1
    SET result = 1;
    COMMIT;
END //
DELIMITER ;


DELIMITER ;



-- Procedura per l'aggiunta di un quesito al test
DELIMITER //
CREATE PROCEDURE AggiungiQuesito(
    IN titoloTest VARCHAR(255),
    IN descrizione TEXT,
    IN difficolta ENUM('Basso', 'Medio', 'Alto'),
    IN tipoQuesito ENUM('RispostaChiusa', 'Codice'),
    OUT result INT
)
BEGIN
    DECLARE testExists INT;
    DECLARE numRisposta INT DEFAULT 0;

    -- Verifica che il test esista
    SELECT COUNT(*) INTO testExists FROM Test WHERE Titolo = titoloTest;

    IF testExists > 0 THEN
```

```
        -- Calcola il numero progressivo del quesito
        SET numRisposta = (SELECT IFNULL(MAX(NumRisposta), 0) + 1 FROM Quesito WHERE
        TitoloTest = titoloTest);

        -- Inserisce il quesito con il numero progressivo calcolato
        INSERT INTO Quesito (TitoloTest, NumRisposta, Difficoltà, Descrizione,
        TipoQuesito)
        VALUES (titoloTest, numRisposta, difficolta, descrizione, tipoQuesito);

        SET result = 1;  -- Quesito aggiunto con successo
    ELSE
        SET result = 0;  -- TestValutaRisposta.html non trovato
    END IF;
END //

DELIMITER ;

-- Procedura che crea riferimento tra il test ed il quesito

DELIMITER //

CREATE PROCEDURE AggiungiRiferimento(
    IN idQuesito INT,
    IN titoloTest VARCHAR(255),
    IN nomeTabellaEsercizio VARCHAR(100),
    OUT result INT
)
BEGIN
    DECLARE quesitoExists INT;
    DECLARE tabellaExists INT;

    -- Verifica che il quesito e la tabella di esercizio esistano
    SELECT COUNT(*) INTO quesitoExists FROM Quesito
    WHERE Id = idQuesito AND TitoloTest = titoloTest;
    SELECT COUNT(*) INTO tabellaExists FROM Tabella_Esercizio
    WHERE Nome = nomeTabellaEsercizio;

    IF quesitoExists > 0 AND tabellaExists > 0 THEN
        -- Inserisce il riferimento
        INSERT INTO Riferimento (NomeTabellaEsercizio,
        IdQuesito, TitoloTest)
        VALUES (nomeTabellaEsercizio, idQuesito, titoloTest);

        SET result = 1;  -- Riferimento aggiunto con successo
    ELSE
        SET result = 0;  -- Quesito o tabella di esercizio non trovati
```

```
    END IF;
END //

DELIMITER ;

-- Procedure per l'inserimento della soluzione per un test
DELIMITER //

CREATE PROCEDURE AggiungiSoluzione(
    IN sketch VARCHAR(255),
    IN idQuesito INT,
    IN titoloTest VARCHAR(255),
    IN nomeTabellaOutput VARCHAR(100)
)
BEGIN
    INSERT INTO Soluzione (Sketch, IdQuesito, TitoloTest, NomeTabellaOutput)
    VALUES (sketch, idQuesito, titoloTest, nomeTabellaOutput);
END //

DELIMITER ;


DELIMITER //
CREATE PROCEDURE InserisciRispostaCodice(
    IN emailStudente VARCHAR(255),
    IN idQuesito INT,
    IN titoloTest VARCHAR(255),
    IN testoRisposta TEXT
)
BEGIN
    -- Inserisce la risposta nel database, con Esito NULL
    INSERT INTO RispostaCodice (EmailStudente, Esito, Data,
    TestoRisposta, IdQuesito, TitoloTest)
    VALUES (emailStudente, NULL, CURRENT_TIMESTAMP, testoRisposta,
    idQuesito, titoloTest);
END //
DELIMITER ;
;

-- Procedura per inserire una risposta a quesito chiuso
DELIMITER //

CREATE PROCEDURE InserisciRispostaChiusa(
    IN emailStudente VARCHAR(255),
    IN dataRisposta TIMESTAMP,
    IN esito BOOLEAN,
```

```
    IN numeroOpzione INT,
    IN idQuesito INT,
    IN titoloTest VARCHAR(255)
)
BEGIN
    INSERT INTO RispostaChiusa (EmailStudente, Data, Esito, NumeroOpzione, IdQuesito, Titolo
    VALUES (emailStudente, dataRisposta, esito, numeroOpzione, idQuesito, titoloTest);
END //

DELIMITER ;


    /* Stored Procedure per Visualizzare i TestValutaRisposta.html Disponibili per un Utente
DELIMITER //

CREATE PROCEDURE VisualizzaTestDisponibili()
BEGIN
    SELECT Titolo, Data, EmailDocente, VisualizzazioneRisposta
    FROM Test;
END //

DELIMITER //

CREATE PROCEDURE AggiungiOpzione(
    IN idQuesito INT,
    IN titoloTest VARCHAR(255),
    IN opzioneCorretta BOOLEAN,
    IN testo TEXT
)
BEGIN
    -- Inserisce la nuova opzione calcolando la numerazione in modo incrementale
    INSERT INTO Opzione (Numerazione, IdQuesito, TitoloTest, OpzioneCorretta, Testo)
    SELECT IFNULL(MAX(Numerazione), 0) + 1, idQuesito, titoloTest, opzioneCorretta, testo
    FROM Opzione
    WHERE IdQuesito = idQuesito AND TitoloTest = titoloTest;
END //

DELIMITER ;
-- trigger che prende tutti i testi di tipo quesito
DELIMITER //

CREATE PROCEDURE VisualizzaQuesitoChiuso()
BEGIN
    SELECT Quesito.Id, Quesito.TitoloTest, Quesito.Descrizione
    FROM Quesito
    WHERE TipoQuesito = 'RispostaChiusa';
```

```
END //

CREATE PROCEDURE VisualizzaQuesitoCodice()
BEGIN
    SELECT Quesito.Id, Quesito.TitoloTest, Quesito.Descrizione
    FROM Quesito
    WHERE TipoQuesito = 'Codice';
END //

DELIMITER ;

-- Procedure che consenti di inserire un messagio tra un studenti e un docente----------
DELIMITER //
CREATE PROCEDURE InviaMessaggioStudente(
    IN titolo VARCHAR(255),
    IN titoloTest VARCHAR(255),
    IN testo TEXT,
    IN emailStudenteMittente VARCHAR(255),
    IN emailDocenteDestinatario VARCHAR(255)
)
BEGIN
    -- Verifica che lo studente e il docente esistano
    IF (SELECT COUNT(*) FROM Studente WHERE Email = emailStudenteMittente) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Studente non valido.';
    END IF;
    IF (SELECT COUNT(*) FROM Docente WHERE Email = emailDocenteDestinatario) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Docente non valido.';
    END IF;

    -- Inserisce il messaggio
    INSERT INTO Messaggio (Titolo, Data, TitoloTest, Testo, EmailStudenteMittente, EmailDoce
    VALUES (titolo, CURRENT_TIMESTAMP, titoloTest, testo, emailStudenteMittente, emailDocent

    -- Aggiunge il docente come destinatario
    INSERT INTO Destinatario (EmailStudente, TitoloMessaggio, TitoloTest, Data)
    VALUES (emailStudenteMittente, titolo, titoloTest, CURRENT_TIMESTAMP);
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE InviaMessaggioDocente(
    IN titolo VARCHAR(255),
    IN titoloTest VARCHAR(255),
    IN testo TEXT,
    IN emailDocenteMittente VARCHAR(255)
```

```
)
BEGIN
    -- Verifica che il docente esista
    IF (SELECT COUNT(*) FROM Docente WHERE Email = emailDocenteMittente) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Docente non valido.';
    END IF;

    -- Inserisce il messaggio
    INSERT INTO Messaggio (Titolo, Data, TitoloTest, Testo, EmailDocenteMittente, EmailDocen
    VALUES (titolo, CURRENT_TIMESTAMP, titoloTest, testo, emailDocenteMittente, NULL);

    -- Aggiunge ogni studente come destinatario del messaggio
    INSERT INTO Destinatario (EmailStudente, TitoloMessaggio, TitoloTest, Data)
    SELECT Email, titolo, titoloTest, CURRENT_TIMESTAMP FROM Studente;
END //
DELIMITER ;




-- Visualizza risposta studente di un determinato test -----
DELIMITER //
CREATE PROCEDURE VisualizzaRisposteStudente(
    IN emailStudente VARCHAR(255),
    IN titoloTest VARCHAR(255)
)
BEGIN
    SELECT R.Id, R.Data, R.Esito, R.TestoRisposta, Q.TitoloTest, Q.Id
    FROM RispostaCodice R
             JOIN Quesito Q ON R.IdQuesito = Q.Id
    WHERE R.EmailStudente = emailStudente AND Q.TitoloTest = titoloTest;
END //
DELIMITER ;

-- Questa procedura visualizza tutte le opzioni di risposta per un quesito di tipo "Risposta
DELIMITER //
CREATE PROCEDURE VisualizzaOpzioniQuesito(
    IN idQuesito INT,
    IN titoloTest VARCHAR(255)
)
BEGIN
    SELECT Numerazione, Testo, OpzioneCorretta
    FROM Opzione
    WHERE IdQuesito = idQuesito AND TitoloTest = titoloTest;
END //
```

```
DELIMITER ;


 -- Cambia stato di un test quando deve essere aggiornato manualmente --
DELIMITER //
CREATE PROCEDURE AggiornaStatoCompletamentoTest(
    IN emailStudente VARCHAR(255),
    IN titoloTest VARCHAR(255),
    IN stato ENUM('InCorso', 'Completato', 'NonIniziato')
)
BEGIN
    UPDATE Completamento
    SET Stato = stato
    WHERE EmailStudente = emailStudente AND TitoloTest = titoloTest;
END //
DELIMITER ;


  -- Procedure per la visualizzazione di tutte le tabelle
DELIMITER //
CREATE PROCEDURE VisualizzaTabelle()
BEGIN
    SELECT Nome, Data FROM Tabella_Esercizio;
END //
DELIMITER ;
  drop procedure CalcolaEsitoTest;
DELIMITER //

CREATE PROCEDURE CalcolaEsitoTest(
    IN emailStudente VARCHAR(255),
    IN titoloTest VARCHAR(255)
)
BEGIN

    SELECT
        r.IdQuesito,
        r.NumeroOpzione AS OpzioneScelta,
        o.OpzioneCorretta,
        CASE WHEN o.OpzioneCorretta = 1 THEN 'Corretto' ELSE 'Errato' END AS Esito,
        o.Testo AS TestoRisposta
    FROM RispostaChiusa r
             JOIN Opzione o ON r.NumeroOpzione = o.Numerazione
        AND r.IdQuesito = o.IdQuesito
        AND r.TitoloTest = o.TitoloTest
    WHERE r.EmailStudente = emailStudente
      AND r.TitoloTest = titoloTest;
```

```
END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE CalcolaEsitiRisposte()
BEGIN
    -- Variabili per iterare sui risultati
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_id INT;
    DECLARE v_idQuesito INT;
    DECLARE v_titoloTest VARCHAR(255);
    DECLARE v_testoRisposta TEXT;
    DECLARE v_sketch TEXT;
    DECLARE v_esito BOOLEAN;

    -- Cursore per iterare su tutte le righe di RispostaCodice
    DECLARE cur CURSOR FOR
        SELECT Id, IdQuesito, TitoloTest, TestoRisposta
        FROM RispostaCodice;

    -- Gestore di uscita dal ciclo
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Apriamo il cursore
    OPEN cur;

    -- Loop per ciascuna risposta di RispostaCodice
    read_loop: LOOP
        -- Leggiamo i dati dal cursore
        FETCH cur INTO v_id, v_idQuesito, v_titoloTest, v_testoRisposta;

        -- Verifichiamo se abbiamo finito
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Recuperiamo la soluzione corrispondente
        SELECT Sketch INTO v_sketch
        FROM Soluzione
        WHERE IdQuesito = v_idQuesito AND TitoloTest = v_titoloTest
        LIMIT 1;

        -- Confrontiamo la risposta con la soluzione
```

```
        SET v_esito = (v_testoRisposta = v_sketch);

        -- Aggiorniamo l'esito nella tabella RispostaCodice
        UPDATE RispostaCodice
        SET Esito = v_esito
        WHERE Id = v_id;
    END LOOP;

    -- Chiudiamo il cursore
    CLOSE cur;
END //

DELIMITER ;

-- ------------------------------------------------------------------ Trigger ---------------
DELIMITER //
-- Trigger per inserimento su RispostaCodice
CREATE TRIGGER Trigger_Completamento_InserisciCodice
    AFTER INSERT ON RispostaCodice
    FOR EACH ROW
BEGIN
    -- Inserisce una nuova riga in Completamento se non esiste già per questo studente e tes
    IF (SELECT COUNT(*) FROM Completamento
        WHERE TitoloTest = NEW.TitoloTest AND EmailStudente = NEW.EmailStudente) = 0 THEN
        INSERT INTO Completamento (TitoloTest, EmailStudente, DataPrimaRisposta, DataUltimaR
        VALUES (NEW.TitoloTest, NEW.EmailStudente, NEW.Data, NEW.Data, 'Aperto');
    ELSE
        -- Aggiorna DataUltimaRisposta se già esiste
        UPDATE Completamento
        SET DataUltimaRisposta = NEW.Data
        WHERE TitoloTest = NEW.TitoloTest AND EmailStudente = NEW.EmailStudente;
    END IF;
END //

-- Trigger per inserimento su RispostaChiusa
CREATE TRIGGER Trigger_Completamento_InserisciChiusa
    AFTER INSERT ON RispostaChiusa
    FOR EACH ROW
BEGIN
    IF (SELECT COUNT(*) FROM Completamento
        WHERE TitoloTest = NEW.TitoloTest AND EmailStudente = NEW.EmailStudente) = 0 THEN
        INSERT INTO Completamento (TitoloTest, EmailStudente, DataPrimaRisposta, DataUltimaR
        VALUES (NEW.TitoloTest, NEW.EmailStudente, NEW.Data, NEW.Data, 'Aperto');
    ELSE
        UPDATE Completamento
        SET DataUltimaRisposta = NEW.Data
```

```
        WHERE TitoloTest = NEW.TitoloTest AND EmailStudente = NEW.EmailStudente;
    END IF;
END //

DELIMITER ;
```

```
-- Questo trigger imposta lo stato del test su "InCompletamento" per uno studente quando vie
DELIMITER //
CREATE TRIGGER Trigger_InCompletamento
    AFTER INSERT ON RispostaChiusa
    FOR EACH ROW
BEGIN
    -- Controlla se il test per questo studente è in stato "NonIniziato" nella tabella ‘Comp
    IF EXISTS (
        SELECT 1 FROM Completamento
        WHERE TitoloTest = NEW.TitoloTest
          AND EmailStudente = NEW.EmailStudente
          AND Stato = 'Aperto'
    ) THEN
        -- Aggiorna lo stato e imposta la data della prima risposta
        UPDATE Completamento
        SET Stato = 'InCompletamento',
            DataPrimaRisposta = NEW.Data
        WHERE TitoloTest = NEW.TitoloTest
          AND EmailStudente = NEW.EmailStudente;
    END IF;
END //

DELIMITER ;

-- Questo trigger imposta lo stato del test su "InCompletamento" per uno studente quando vie
DELIMITER //

CREATE TRIGGER Trigger_InCompletamentoCodice
    AFTER INSERT ON RispostaCodice
    FOR EACH ROW
BEGIN
    -- Controlla se il test per questo studente è in stato "NonIniziato" nella tabella ‘Comp
    IF EXISTS (
        SELECT 1 FROM Completamento
```

```
        WHERE TitoloTest = NEW.TitoloTest
          AND EmailStudente = NEW.EmailStudente
          AND Stato = 'Aperto'
    ) THEN
        -- Aggiorna lo stato e imposta la data della prima risposta
        UPDATE Completamento
        SET Stato = 'InCompletamento',
            DataPrimaRisposta = NEW.Data
        WHERE TitoloTest = NEW.TitoloTest
          AND EmailStudente = NEW.EmailStudente;
    END IF;
END //

DELIMITER ;

-- trigger per settare lo stato di un test come concluso, qaundo il docente cambia la visual
DELIMITER //

CREATE TRIGGER Trigger_Concluso_TuttiStudenti
    AFTER UPDATE ON Test
    FOR EACH ROW
BEGIN
    IF NEW.VisualizzazioneRisposta = TRUE THEN
        UPDATE Completamento
        SET Stato = 'Concluso'
            -- DataUltimaRisposta = CURRENT_TIMESTAMP
        WHERE TitoloTest = NEW.Titolo;
    END IF;
END //

DELIMITER ;




-- trigger che mette come concluso tutti i testi che hanno le risposte correte--
DELIMITER //

CREATE TRIGGER Trigger_ConcludiTest
    AFTER INSERT ON RispostaCodice
    FOR EACH ROW
BEGIN
    DECLARE risposte_totali INT;
    DECLARE risposte_date INT;
    DECLARE risposte_corrette INT;
```

```
    -- Conta tutti i quesiti nel test corrente
    SELECT COUNT(*) INTO risposte_totali
    FROM Quesito
    WHERE TitoloTest = NEW.TitoloTest;

    -- Conta tutte le risposte fornite dallo studente (sia chiuse che di codice)
    SELECT COUNT(DISTINCT IdQuesito) INTO risposte_date
    FROM (
             SELECT IdQuesito FROM RispostaCodice
             WHERE EmailStudente = NEW.EmailStudente AND TitoloTest = NEW.TitoloTest
             UNION ALL
             SELECT IdQuesito FROM RispostaChiusa
             WHERE EmailStudente = NEW.EmailStudente AND TitoloTest = NEW.TitoloTest
         ) AS RisposteDate;

    -- Conta tutte le risposte corrette dallo studente
    SELECT COUNT(DISTINCT IdQuesito) INTO risposte_corrette
    FROM (
             SELECT IdQuesito FROM RispostaCodice
             WHERE EmailStudente = NEW.EmailStudente AND TitoloTest = NEW.TitoloTest AND Esi
             UNION ALL
             SELECT IdQuesito FROM RispostaChiusa
             WHERE EmailStudente = NEW.EmailStudente AND TitoloTest = NEW.TitoloTest AND Esi
         ) AS RisposteCorrette;

    -- Concludi il test solo se:
    -- (i) il numero di risposte date equivale al numero di quesiti nel test
    -- (ii) il numero di risposte corrette equivale al numero di quesiti nel test
    IF risposte_totali = risposte_date AND risposte_totali = risposte_corrette THEN
        UPDATE Completamento
        SET Stato = 'Concluso', DataUltimaRisposta = NEW.Data
        WHERE TitoloTest = NEW.TitoloTest AND EmailStudente = NEW.EmailStudente;
    END IF;
END //

DELIMITER ;




-- Trigger per Incrementare numrisposte al Aggiunta di una Nuova Risposta---------------

CREATE TRIGGER IncrementaNumRisposte
    AFTER INSERT ON RispostaChiusa
```

```
    FOR EACH ROW
BEGIN
    UPDATE Quesito
    SET NumRisposta = NumRisposta + 1
    WHERE Id = NEW.IdQuesito AND TitoloTest = NEW.TitoloTest;
END;
CREATE TRIGGER IncrementaNumRisposteCodice
    AFTER INSERT ON RispostaCodice
    FOR EACH ROW
BEGIN
    UPDATE Quesito
    SET NumRisposta = NumRisposta + 1
    WHERE Id = NEW.IdQuesito AND TitoloTest = NEW.TitoloTest;
END;


-- -------------------------------------------------Statisctiche----------------
-- Questa vista conta il numero di test completati per ciascuno studente.
CREATE VIEW Classifica_Test_Completati AS
SELECT Codice,
       COUNT(*) AS NumeroTestCompletati
FROM Completamento, Studente
WHERE Stato = 'Concluso' and Email=EmailStudente
GROUP BY Codice
ORDER BY NumeroTestCompletati DESC;


-- Questa vista mostra la percentuale di risposte corrette inserite da ogni studente.
CREATE VIEW Classifica_Risposte_Corrette AS
SELECT Codice,
       (SUM(CASE WHEN Esito = TRUE THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS PercentualeCorret
FROM (
         SELECT Codice, Esito FROM RispostaChiusa, Studente
                  where EmailStudente= Email
         UNION ALL
         SELECT Codice, Esito FROM RispostaCodice,Studente
                              where Email=EmailStudente
     ) AS Risposte
GROUP BY Codice
ORDER BY PercentualeCorrette DESC;

-- Questa vista conta il numero di risposte fornite per ciascun quesito, indipendentemente c

CREATE VIEW Classifica_Quesiti AS
SELECT IdQuesito,
       TitoloTest,
```

```
       COUNT(*) AS NumeroRisposte
FROM (
         SELECT IdQuesito, TitoloTest FROM RispostaChiusa
         UNION ALL
         SELECT IdQuesito, TitoloTest FROM RispostaCodice
     ) AS Risposte
GROUP BY IdQuesito, TitoloTest
ORDER BY NumeroRisposte DESC;


DELIMITER //

CREATE PROCEDURE OttieniClassificaTestCompletati()
BEGIN
    SELECT * FROM Classifica_Test_Completati;
END //

DELIMITER ;
```