Alma Mater Studiorum - Università di Bologna

-

Corso di Laurea in Informatica per il Management

Insegnamento: Basi di Dati

Progetto ESQL

Anno Accademico 2023/2024

Indice

Fase 1: Raccolta / Analisi dei requisiti

Testo completo delle specifiche sui dati

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 gli 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). E' 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 delle operazioni

- Registrazione di un docente, o studente, sulla piattaforma
- Autenticazione di un docente, o studente, sulla piattaforma
- Visualizzazione dei test disponibili da parte di un docente, o studente
- Visualizzazione dei quesiti presenti in ogni test da parte di un docente
- Effettuazione del test (quindi visualizzazione dei quesiti) da parte di uno studente
- Inserimento di una nuova tabella di esercizio, con relativi metadati, da parte di un docente
- Inserimento di una riga per una tabella di esercizio definita dal docente
- Creazione di nuovo test da parte del docente
- Creazione di un nuovo quesito, riferito ad un test, con le relative risposte da parte di un docente
- Possibilità del docente di abilitare / disabilitare la visualizzazione delle risposte per uno specifico test
- Invio di un messaggio da parte di un docente a tutti gli studenti
- Invio di un messaggio da parte di uno studente ad un docente per uno specifico test
- Possibilità dello studente di rispondere ai quesiti
- Possibilità dello studente di visualizzare l'esito della risposta nel caso di domande di codice
- Possibilità dello studente di visualizzare l'esito del test se il docente setta "visualizzaRisposta"

Tavola media dei volumi

Concetto	Tipo	Volume
Utente	E	303
Studente	E	300
Docente	E	3
InvioDocente	R	20
RicezioneDocente	R	15000
Messaggio	E	15000+ 6000+15000+20=36020
InvioStudente	R	5*10*300=15000 (5 mess a studente per ogni test)
RicezioneStudente	R	(300*20) = 6000
Realizzazione	R	3000
Completamento	E	10 test * 300 studenti = 3000
Composizione	R	10 risposte * 3000 completamenti = 30000
Risposta	E	10 risposte * 3000 completamenti = 30000
Risposta a quesiti di codice	E	15000
Risposta a domande chiuse	E	15000
Riferimento	R	10 test * 36020 mess = 360200
Creazione	R	10
Test	E	10
Ideazione	R	30
Tabella di esercizio	E	20 tab * 10 test = 200
Formazione	R	10 test * 300 studenti = 3000
Inclusione	R	10 test * 10 domande = 100

Concetto	Tipo	Volume
Quesito	E	10 test * 10 domande = 100
Costituzione	R	2 tab a quesito * 100 quesiti = 200
Quesito a risposta chiusa	E	5 domande * 10 test = 50
Quesito di codice	E	5 domande * 10 test = 50
Inserimento	R	50 domande * 4 opzioni = 200
Opzione risposta	E	50 domande * 4 opzioni = 200
Verifica	R	5 risposta * 50 quesiti = 250
Controllo	R	5 risposta * 50 quesiti = 250
Disposizione	R	5 attributi * 200 tab = 2500
Attributo	E	5 attributi * 200 tab = 2500
Vincolo di integrità	R	2 vincoli * 5 attributi * 200 tabelle = 5000

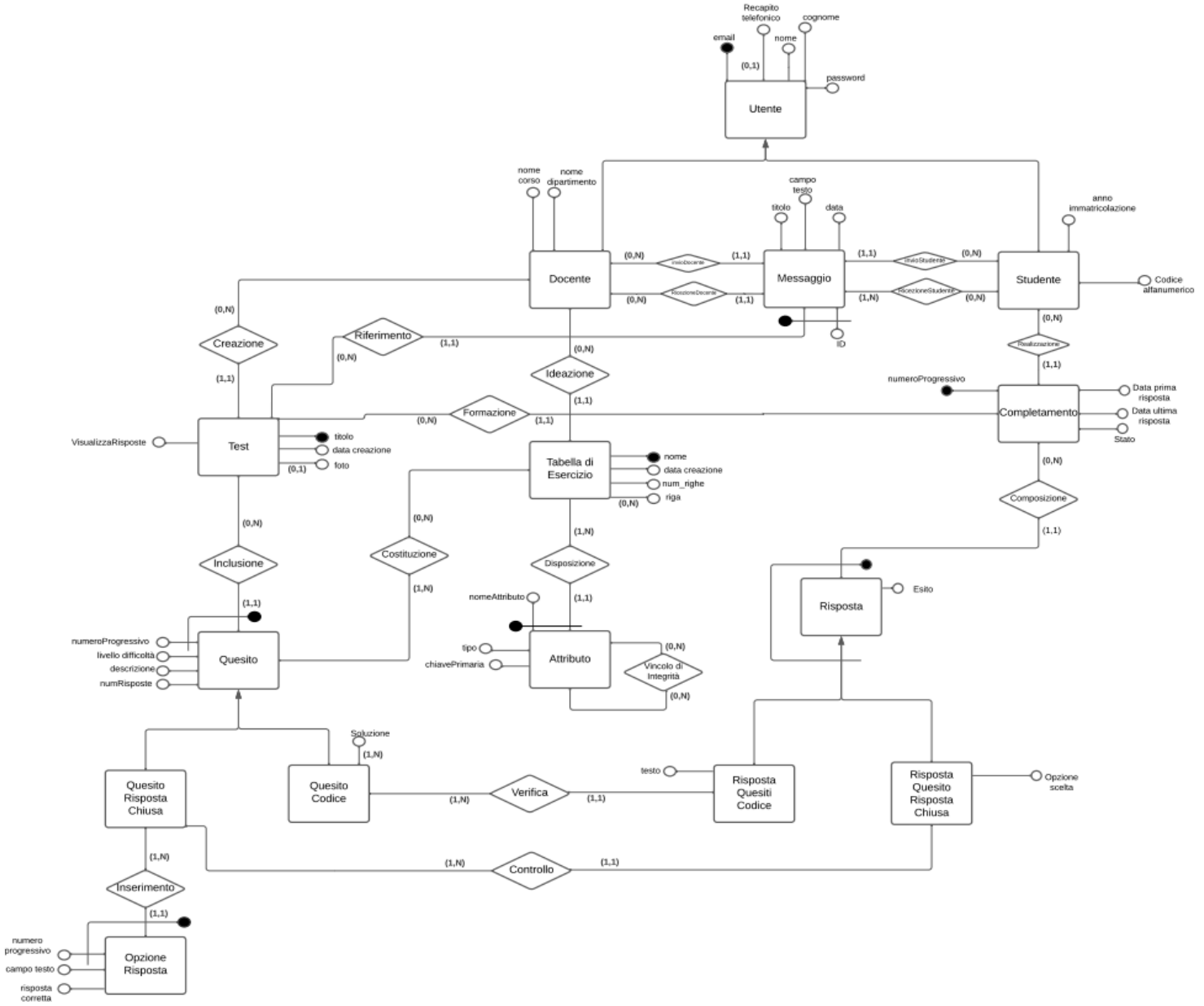
Glossario dei dati

Termine	Descrizione	Sinonimi	Collegamenti
Utente	persona che utilizza la piattaforma		Studente, Docente
Docente	insegnante del corso, crea test, tabelle di esercizio, quesiti, ed invia messaggi		Utente, Test, Messaggio, Tabella di Esercizio
Studente	studente che accede alla piattaforma, manda messaggi ed esegue test		Utente, Messaggio, Completamento
Tabella di Esercizio	tabella di dati necessaria a svolgere test		Docente, Quesito, Attributo
Attributo	attributo di un'entita del database		Tabella di Esercizio, Attributo
Messaggio	testo mandato e ricevuto da studenti e docenti		Docente, Studente, Test
Completamento	tiene traccia dello stato di completamento di un test da parte dello studente		Studente, Test, Risposta
Risposta	risposta posta ad un quesito di un test		Risposta a Quesito di Codice, Risposta a Quesito a risposta chiusa, Completamento
Quesito	domanda scritta dal docente a cui gli studenti devono rispondere		Test, Tabella di esercizio, quesito a risposta chiusa, quesito di codice
Quesito A Risposta Chiusa	domanda a scelta multipla con una o più risposte corrette		quesito, opzione risposta, risposta a quesito a risposta

			chiusa
Quesito di Codice	domanda che richiede un codice SQL come soluzione		quesito, risposta a quesito di codice, soluzione
Risposta a Quesito A Risposta Chiusa	risposta scelta dallo studente per un quesito a risposta chiusa	Risposta Quesito risposta chiusa	risposta, quesito a risposta chiusa
Risposta a Quesito di Codice	risposta scelta dallo studente per un quesito di tipo codice	Risposta Quesito Codice	risposta, quesito di codice
Test	insieme di quesiti da svolgere per uno studente		Docente, Messaggio, Quesito, Completamento
Soluzione	Codice da inserire per ottenere un determinato output		Quesito di codice
Opzione Risposta	Opzione del quesito a scelta		Quesito a risposta chiusa

Fase 2: Progettazione concettuale

Diagramma E-R



Dizionario delle Entità

Entità	Descrizione	Attributi	Identificatore
Docente	un docente registrato nel sistema.	Email, PasswordDocente, Nome, Cognome, recapitoTelefonicoDocente, NomeDipartimento, NomeCorso	Email
Studente	uno studente registrato nel sistema	Email, PasswordStudente, Nome, Cognome, RecapitoTelefonicoStudente, AnnoImmatricolazione, CodiceAlfanumerico	Email
Messaggio	messaggi inviati all'interno del sistema	Id, TitoloTest, TitoloMessaggio, CampoTesto, Data	Id, TitoloTest
RicezioneStudente	Associazione tra messaggio e studente destinatario	Id, TitoloTest, EmailStudenteDestinatario	Id, TitoloTest, EmailStudenteDestinatario
InvioStudente	Associazione tra messaggio e studente mittente	Id, TitoloTest, EmailStudenteMittente	Id, TitoloTest, EmailStudenteMittente
RicezioneDocente	Associazione tra messaggio e docente destinatario	Id, TitoloTest, EmailDocenteDestinatario	Id, TitoloTest, EmailDocenteDestinatario
InvioDocente	Associazione tra messaggio e docente mittente	Id, TitoloTest, EmailDocenteMittente	Id, TitoloTest, EmailDocenteMittente
Completamento	Stato di completamento di un test da parte di uno studente	NumeroProgressivo, Stato, TitoloTest, EmailStudente, DataPrimaRisposta, DataUltimaRisposta	NumeroProgressivo
RispostaQuesitoCodice	risposta ai quesiti di tipo codice	NumeroProgressivoCompletamento, TitoloTest, Testo, NumeroProgressivoQuesito, Esito	NumeroProgressivoCompletamento, TitoloTest, NumeroProgressivoQuesito

Entità	Descrizione	Attributi	Identificatore
RispostaQuesitoRisposta Chiusa	risposta ai quesiti a risposta chiusa	NumeroProgressivoComplet amento, TitoloTest, OpzioneScelta, NumeroProgressivoQuesito, Esito	NumeroProgressivo Completamento, TitoloTest, NumeroProgressivo Quesito
Test	test creato da un docente	Titolo, DataCreazione, Foto, VisualizzaRsiposte, EmailDocente	Titolo
TabellaDiEsercizio	tabella creata da un docente	Nome, DataCReazione, num_righe, EmailDocente	Nome
Riga		Testo, NomeTabella	Testo, NomeTabella
Quesito	quesito all'interno di un test	NumeroProgressivo, TitoloTest, LivelloDifficolta, Descrizione, NumeroRisposte	NumeroProgressivo, TitoloTest
QuesitoCodice	quesito di tipo codice	NumeroProgressivo, TitoloTest	NumeroProgressivo, TitoloTest
QuesitoRispostaChiusa	quesito a risposta chiusa	NumeroProgressivo, TitoloTest	NumeroProgressivo, TitoloTest
OpzioneRisposta	opzione di risposta per i quesiti a risposta chiusa	NumeroProgressivoOpzione, TitoloTest, NumeroProgressivoQuesito, CampoTesto, RispostaCorretta	NumeroProgressivo Opzione, NumeroProgressivo Quesito, TitoloTest
Soluzione	risposta corretta ad un quesito di codice	NumeroProgressivo, TitoloTest, TestoSoluzione	NumeroProgressivo, TitoloTest, TestoSoluzione
Attributo	attributo di una tabella creata dal docente	NomeTabella, NomeAttributo, Tipo, chiavePrimaria	NomeTabella, NomeAttributo
VincoloDiIntegrità	relazione tra due o più attributi	NomeTabellaUno, NomeAttributoUno, NomeTabellaDue, NomeAttributoDue	NomeTabellaUno, NomeAttributoUno, NomeTabellaDue, NomeAttributoDue
Costituzione	Relazione tra test e tabella	TitoloTest, NumeroProgressivoQuesito, NomeTabella	TitoloTest, NumeroProgressivo Quesito, NomeTabella

Dizionario delle Relazioni

Relazione	Descrizione	Componenti	Attributi
InvioDocente	Messaggi inviati da docenti	Docente - Messaggio	
RicezioneDocente	Messaggi ricevuti dai docenti	Docente - Messaggio	
InvioStudente	Messaggi inviati dagli studenti	Studente - Messaggio	
RicezioneStudente	Messaggi ricevuti dagli studenti	Studente - Messaggio	
Realizzazione	Stato di realizzazione di un test da parte degli studenti	Studente - Completamento	
Ideazione	Ideazione di una tabella da parte di un docente	Docente - TabellaDiEsercizio	
Formazione	Legame tra lo stato di completamento e il test(da modificare)	Completamento - Test	
Riferimento	Ogni messaggio è legato a un test	Messaggio - Test	
Creazione	Docente crea un test	Docente - Test	
Composizione	Completamento contiene le risposte inserite dagli studenti	Completamento - Risposta	
Disposizione	Come ogni attributo si lega alla rispettiva tabella	TabellaDiEsercizio - Attributo	
Costituzione	Tabella riferita al quesito	TabellaDiEsercizio - Quesito	
Inclusione	Test formato da quesiti	Test - Quesito	
Verifica	Della risposta ai quesiti di codice	QuesitoCodice - RispostaQuesitiCodice	
Controllo	Della scelta nei	QuesitoRispostaChiusa -	

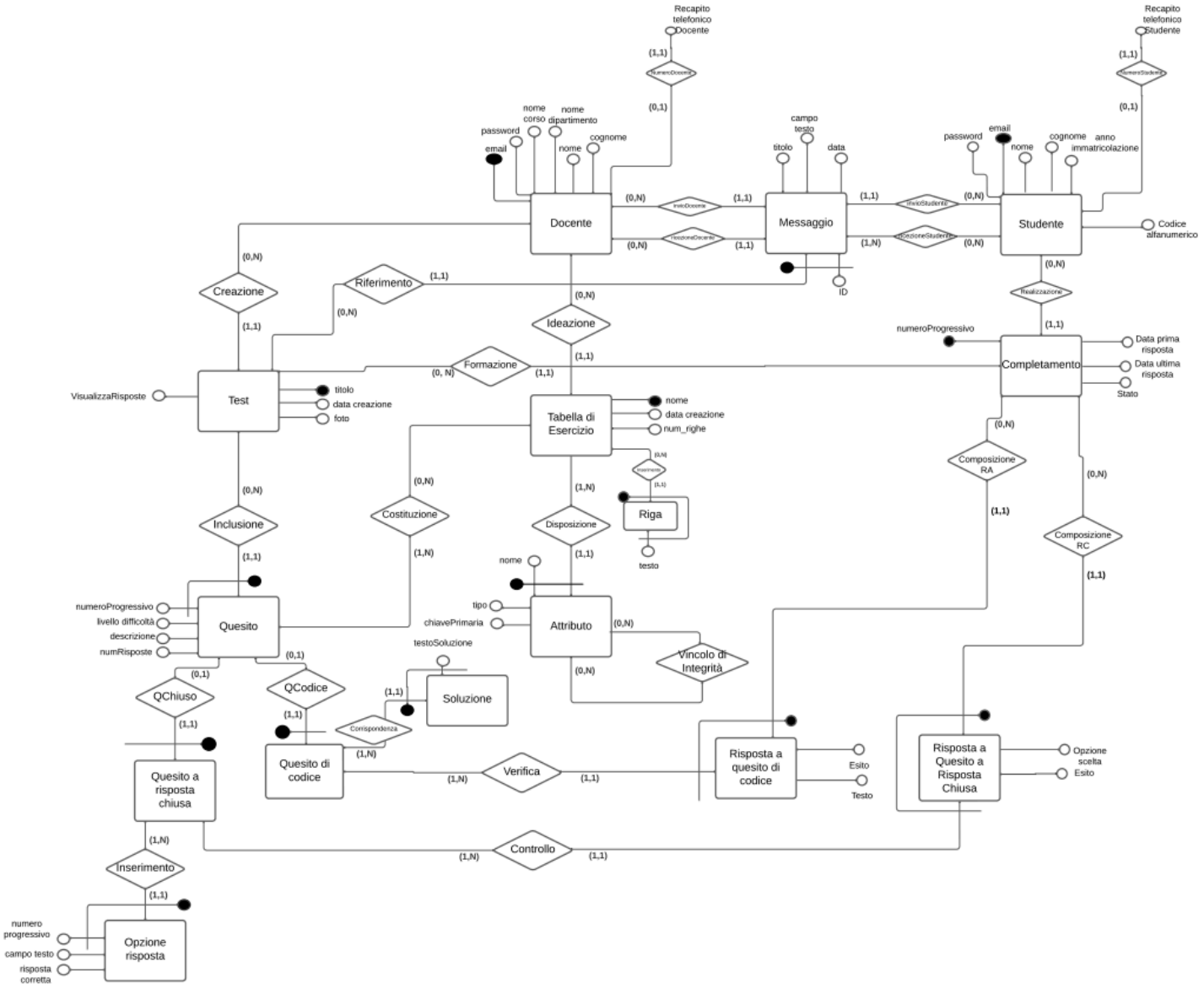
	quesiti a risposta chiusa	RispostaQuesitoRispostaChiusa	
Inserimento	opzione di risposta al rispettivo quesito a risposta chiusa	QuesitoRispostaChiusa - OpzioneRisposta	

Tavola delle Business Rules

Gestione dello stato del test: il test può avere uno stato di Aperto, InCompletamento o Concluso, a seconda dello stato di completamento delle risposte degli studenti.
Visualizzazione delle risposte: se il campo VisualizzaRisposte di un test è impostato su True, le risposte dei quesiti diventano visibili agli studenti.
Gestione delle risposte degli studenti: gli studenti possono inviare risposte per i quesiti, e ogni risposta può avere un esito (True o False) in base alla corrispondenza con la risposta corretta
Nelle classifiche riguardanti gli studenti, non devono mai apparire i dati sensibili di questi, ma solamento il codice alfanumerico
Il "Codice Alfanumerico" dello studente deve essere lungo esattamente 16 caratteri
Finché lo stato del test non è concluso, lo studente può rispondere ad un quesito più volte, e la risposta effettiva da considerare sarà l’ultima che ha dato
Ogni studente può inviare un messaggio ad un solo docente per volta, mentre ogni messaggio inserito da un docente viene recapitato a tutti gli studenti

Fase 3: Progettazione Logica

Ristrutturazione dello schema concettuale



Analisi delle ridondanze

Valutazione della ridondanza #num_risposte relativa ad un quesito.

Operazione 1: Aggiungere una nuova risposta ad un quesito esistente (10 volte/mese, interattiva)

Con #num-risposte:
C(Op1) = 10 * 1 * (2 * 6 + 0) = 120

Senza #num-risposte:
C(Op1)=10*1*(2*3 + 0) = 60

Operazione 2: Rimuovere un quesito e tutte le risposte ottenute (2 volte/mese, batch)
Stimando 10 risposte che possono essere o di un quesito a risposta chiusa o di un quesito di codice (non considerando l'entità "Opzione Risposta")

2 relazioni con test e tabella + 1 quesito + 1 relazione + 1 quesito specifico + 10 relazioni + 10 entità risposta + 10 relazioni

Con #num-risposte:
C(Op2) = 2 * 0.5 * (2 * 35 + 0) = 70

Senza #num-risposte:
C(Op2) = 2 * 0.5 * (2 * 35 + 0) = 70

Operazione 3: Visualizzare tutti gli utenti presenti nella pattaforme (1 volta/mese, batch)
Non avendo l'entità Utente si considerano 50 utenti divisi tra Docenti e Studenti

Con #num-risposte:
C(Op3)=1 * 0.5* (2 * 0 + 50) = 25

Senza#num-risposte:
C(Op3)=1 * 0.5* (2 * 0 + 50) = 25

Operazione 4: Contare il numero di risposte per ciascun quesito presente nella piattaforma (2 volte/mese, interattiva)

Con #num-risposte:
C(Op4) = 2 * 1 * (0 + 20) = 40

Senza #num-risposte:
C(Op4) = 2 * 1 * (0 + 20 * 20) = 800
10 risposte + 10 relazioni per ogni quesito (20 quesiti)

Costi totali con ridondanza: 255

Costi totali senza ridondanza: 955

Dato che nell’operazione 4 riuscire a determinare quante risposte ogni quesito possiede diventa un’operazione molto costosa,mantenere il campo numRisposte conviene.

Lista delle tabelle con vincoli di chiavi

DOCENTE (Email, Nome, Cognome, PasswordDocente, RecapitoTelefonicoDocente, NomeDipartimento, NomeCorso)
STUDENTE (Email, PasswordStudente, Nome, Cognome, RecapitoTelefonicoStudente, Anno immatricolazione, CodiceAlfanumerico)
MESSAGGIO (Id, TitoloTest, TitoloMessaggio, Campo testo, Data)
RICEZIONESTUDENTE(Id, TitoloTest, EmailStudenteDestinatario)
INVIOSTUDENTE (Id, TitoloTest, EmailStudenteMittente)
RICEZIONEDOCENTE (Id, TitoloTest, EmailDocenteDestinatario)
INVIODOCENTE (Id, TitoloTest, EmailDocenteMittente)
TEST (Titolo, DataCreazione, Foto, VisualizzaRisposte, EmailDocente)
COMPLETAMENTO (NumeroProgressivo, TitoloTest, EmailStudente, Stato, DataPrimaRisposta, DataUltimaRisposta)
TABELLADIESERCIZIO (Nome, DataCreazione, num_righe, EmailDocente)
RIGA (Testo, NomeTabella)
ATTRIBUTO (NomeTabella, NomeAttributo, Tipo, ChiavePrimaria)
VINCOLODIINTEGRITA (NomeTabellaUno, NomeAttributoUno, NomeTabellaDue, NomeAttributoDue)
QUESITO (TitoloTest, NumeroProgressivo, LivelloDifficoltà, Descrizione, NumeroRisposte)
QUESITORISPOSTACHIUSA(TitoloTest, NumeroProgressivo)
QUESITOCODICE(TitoloTest, NumeroProgressivo)
SOLUZIONE(TitoloTest,NumeroProgressivo,TestoSoluzione)
RISPOSTAQUESITORISPOSTACHIUSA (NumeroProgressivoCompletamento, TitoloTest, NumeroProgressivoQuesito, OpzioneScelta, Esito)
RISPOSTAQUESITOCODICE (NumeroProgressivoCompletamento, TitoloTest, NumeroProgressivoQuesito, Testo, Esito)
OPZIONERISPOSTA (TitoloTest, NumeroProgressivoQuesito, NumeroProgressivoOpzione, CampoTesto, RispostaCorretta)
COSTITUZIONE (TitoloTest, NumeroProgressivoQuesito, NomeTabella)

Lista dei vincoli inter-relazionali

TEST.EmailDocente→DOCENTE.Email
MESSAGGIO.TitoloTest→TEST.titolo
RICEZIONESTUDENTE.Id→MESSAGGIO.Id
RICEZIONESTUDENTE.TitoloTest→TEST.titolo
RICEZIONESTUDENTE.EmailStudenteDestinatario→STUDENTE.Email
INVIOSTUDENTE.Id→MESSAGGIO.Id
INVIOSTUDENTE.TitoloTest→TEST.titolo
INVIOSTUDENTE.EmailStudenteMittente→STUDENTE.Email
RICEZIONEDOCENTE.Id→MESSAGGIO.Id
RICEZIONEDOCENTE.TitoloTest→TEST.titolo
RICEZIONEDOCENTE.EmailDocenteDestinatario→DOCENTE.Email
INVIODOCENTE.Id→MESSAGGIO.Id
INVIODOCENTE.TitoloTest→TEST.titolo
INVIODOCENTE.EmailDocenteMittente→DOCENTE.Email
COMPLETAMENTO.TitoloTest→TEST.titolo
COMPLETAMENTO.EmailStudente→STUDENTE.Email
TABELLADIESERCIZIO.EmailDocente→DOCENTE.Email
RIGA.NomeTabella→TABELLADIESERCIZIO.Nome
ATTRIBUTO.NomeTabella→TABELLADIESERCIZIO.Nome
VINCOLODIINTEGRITA.NomeTabellaUno→TABELLADIESERCIZIO.Nome
VINCOLODIINTEGRITA.NomeAttributoUno→ ATTRIBUTO.NomeAttributo
VINCOLODIINTEGRITA.NomeTabellaDue→TABELLADIESERCIZIO.Nome
VINCOLODIINTEGRITA.NomeAttributoDue→ ATTRIBUTO.NomeAttributo
QUESITO.TitoloTest→TEST.Titolo
QUESITORISPOSTACHIUSA.TitoloTest→QUESITO.TitoloTest
QUESITORISPOSTACHIUSA.NumeroProgressivo→QUESITO.NumeroProgressivo
QUESITOCODICE.TitoloTest→QUESITO.TitoloTest
QUESITOCODICE.NumeroProgressivo→QUESITO.NumeroProgressivo
SOLUZIONE.TitoloTest→QUESITOCODICE.TitoloTest
SOLUZIONE.NumeroProgressivo→QUESITOCODICE.NumeroProgressivo
RISPOSTAQUESITORISPOSTACHIUSA.NumeroProgressivoCompletamento→COMPLETAMENTO.NumeroProgressivo
RISPOSTAQUESITORISPOSTACHIUSA.NumeroProgressivoQuesito→QUESITORISPOSTACHIUSA.NumeroProgressivo
RISPOSTAQUESITORISPOSTACHIUSA.TitoloTest→QUESITORISPOSTACHIUSA.TitoloTest
RISPOSTAQUESITOCODICE.NumeroProgressivoCompletamento→COMPLETAMENTO.NumeroProgressivo
RISPOSTAQUESITOCODICE.NumeroProgressivoQuesito→QUESITOCODICE.NumeroProgressivo
RISPOSTAQUESITOCODICE.TitoloTest→QUESITOCODICE.TitoloTest
OPZIONERISPOSTA.TitoloTest→QUESITORISPOSTACHIUSA.TitoloTest

OPZIONERISPOSTA.NumeroProgressivoQuesito→QUESITORISPOSTACHIUSA.NumeroProgressivo
COSTITUZIONE.TitoloTest→QUESITO.TitoloTest
COSTITUZIONE.NumeroProgressivoQuesito→QUESITO.NumeroProgressivo
COSTITUZIONE.NomeTabella→TABELLADIESERCIZIO.Nome

Fase 4: Descrizione ad Alto Livello delle Funzionalità dell'Applicazione

La piattaforma ESQL ospita due tipologie di utenti: docenti e studenti.
Nella schermata iniziale si può eseguire l'accesso e la registrazione. Per la registrazione dello studente sono richiesti: nome, cognome, email, password, recapito telefonico, codice alfanumerico, e anno di immatricolazione, mentre per la registrazione del docente bisogna inserire: nome, cognome, email, password, recapito telefonico, nome del dipartimento, e nome del corso. Una volta che lo studente o il docente è registrato, è sufficiente inserire email e password dopo aver specificato il ruolo per eseguire l'accesso alla piattaforma.

I **Docenti** hanno accesso a diverse funzionalità, accessibili grazie alla barra di navigazione in alto alla pagina. Qui, oltre al tasto per effettuare il logout, sono presenti quattro bottoni, Gestione Test, Gestione Tabelle, Messaggi, e Visualizza Statistiche, ognuno dei quali reindirizza alla pagina desiderata.
In **Gestione Test** saranno visibili i test creati sulla piattaforma e i loro dati. Su ognuno di loro è possibile eseguire tre azioni: cancellarlo, modificarlo, e visualizzarne le tabelle associate. Inoltre, è possibile creare un nuovo test, scegliendone nome ed eventuale foto associata. Successivamente, nella sezione modifica è possibile attivare la visualizzazione delle risposte dei test, così come eliminare quesiti ed aggiungerne di nuovi.
Per quest'ultima fase è necessario scegliere il tipo (se a scelta o se di codice), il numero di risposte corrette, la descrizione, e il livello di difficoltà. Nella schermata successiva bisognerà collegare le tabelle associate, e in quella ancora seguente inserire le opzioni risposta (se si tratta di un quesito a risposta chiusa) o le soluzioni (se si tratta di un quesito di codice.. Infine, se si tratta di un quesito a scelta, si dovrà selezionare quale è la risposta corretta tra le varie inserite.
Nella sezione **Gestione Tabelle** è possibile visualizzare tutte le tabelle che sono state create dal docente. Inoltre, si possono creare nuove tabelle inserendo il codice SQL corretto, oltre che aggiungere ed eliminare righe, ed infine eliminare tabelle già esistenti.
Nella pagina **Messaggi** sono visualizzabili i messaggi inviati dal docente e quelli ricevuti, oltre alla possibilità di mandarne di nuovi cliccando sul tasto "Invia Nuovo Messaggio". In questa sezione bisognerà inserire il test associato al messaggio (in quanto ogni messaggio deve essere relativo ad un test), l'oggetto del messaggio, ed infine il testo.
L'ultima pagina disponibile per il docente è **Visualizza Statistiche**, in cui si possono vedere tre statistiche sull'applicazione ESQL: la classifica degli studenti (visualizzando il codice alfanumerico) sulla base del numero di test completati; la classifica degli studenti sulla base del numero di risposte corrette in rapporto a quelle date; la classifica dei quesiti a cui sono state date più risposte.

La barra di navigazione nella parte relativa agli **Studenti** possiede tre pagine: Test Disponibili, Messaggi, e Visualizza Statistiche, oltre al tasto di logout come in precedenza.
Nella sezione **Test Disponibili** lo studente ha a disposizione i vari Test creati dai docenti, di conseguenza può sceglierne uno ed effettuarlo. Una volta cliccato il testo "Effettua Test" compariranno le domande una alla volta, e lo studente avrà modo di dare una risposta e

muoversi con i tasti avanti e indietro nel test. Arrivato all'ultima domanda, cliccando su "Salva Test", quest'ultimo verrà salvato e avverrà un reindirizzamento alla pagina principale.
Nella pagina **Messaggi** sono visualizzabili i messaggi inviati dallo studente e quelli ricevuti analogamente a quanto descritto nella sezione Docente. Per mandare nuovi messaggi bisognerà inserire il test associato al messaggio, il docente destinatario, l'oggetto del messaggio, ed infine il testo.
L'ultima pagina, come per il docente, è **Visualizza Statistiche**, in cui si possono vedere le tre statistiche dell'applicazione ESQL: la classifica degli studenti (sempre visualizzando il codice alfanumerico) sulla base del numero di test completati; la classifica degli studenti sulla base del numero di risposte corrette in rapporto a quelle date; la classifica dei quesiti a cui sono state date più risposte.

Fase 5: Codice SQL completo dello schema della base di dati

Tabelle

```
DROP DATABASE IF EXISTS ESQL;
CREATE DATABASE IF NOT EXISTS ESQL;
USE ESQL;

CREATE TABLE DOCENTE (
    Email VARCHAR(100) PRIMARY KEY,
    PasswordDocente VARCHAR(20) NOT NULL,
    Nome VARCHAR (50) NOT NULL,
    Cognome VARCHAR (50) NOT NULL,
    RecapitoTelefonicoDocente INT,
    NomeDipartimento VARCHAR(100),
    NomeCorso VARCHAR(100)

) ENGINE = INNODB;

CREATE TABLE STUDENTE (
    Email VARCHAR(100) PRIMARY KEY,
    PasswordStudente VARCHAR(20) NOT NULL,
    Nome VARCHAR (50) NOT NULL,
    Cognome VARCHAR (50) NOT NULL,
    RecapitoTelefonicoStudente INT,
    AnnoImmatricolazione INT,
    CodiceAlfaNumerico CHAR(16)

) ENGINE = INNODB;

CREATE TABLE TEST (
    Titolo VARCHAR(100) PRIMARY KEY,
    DataCreazione DATETIME,
    Foto VARCHAR(255),
    VisualizzaRisposte BOOLEAN,
    EmailDocente VARCHAR(100) NOT NULL,

    FOREIGN KEY(EmailDocente) REFERENCES DOCENTE(Email) ON DELETE
CASCADE
```

```
) ENGINE = INNODB;

CREATE TABLE MESSAGGIO (
    Id INT auto_increment,
    TitoloTest VARCHAR(100) NOT NULL,
    TitoloMessaggio VARCHAR(100),
    CampoTesto VARCHAR(500),
    Data DATETIME,

    PRIMARY KEY(Id, TitoloTest),

    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE RICEZIONESTUDENTE (
    Id INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    EmailStudenteDestinatario VARCHAR(100) NOT NULL,

    PRIMARY KEY(Id, TitoloTest, EmailStudenteDestinatario),

    FOREIGN KEY(Id) REFERENCES MESSAGGIO(Id) ON DELETE CASCADE,
    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE,
    FOREIGN KEY(EmailStudenteDestinatario) REFERENCES STUDENTE(Email)
ON DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE INVIOSTUDENTE (
    Id INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    EmailStudenteMittente VARCHAR(100) NOT NULL,

    PRIMARY KEY(Id, TitoloTest, EmailStudenteMittente),

    FOREIGN KEY(Id) REFERENCES MESSAGGIO(Id) ON DELETE CASCADE,
    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE,
    FOREIGN KEY(EmailStudenteMittente) REFERENCES STUDENTE(Email) ON
DELETE CASCADE

) ENGINE = INNODB;
```

```
CREATE TABLE RICEZIONEDOCENTE (
    Id INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    EmailDocenteDestinatario VARCHAR(100) NOT NULL,

    PRIMARY KEY(Id, TitoloTest, EmailDocenteDestinatario),

    FOREIGN KEY(Id) REFERENCES MESSAGGIO(Id) ON DELETE CASCADE,
    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE,
    FOREIGN KEY(EmailDocenteDestinatario) REFERENCES DOCENTE(Email) ON
DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE INVIODOCENTE (
    Id INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    EmailDocenteMittente VARCHAR(100) NOT NULL,

    PRIMARY KEY(Id, TitoloTest, EmailDocenteMittente),

    FOREIGN KEY(Id) REFERENCES MESSAGGIO(Id) ON DELETE CASCADE,
    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE,
    FOREIGN KEY(EmailDocenteMittente) REFERENCES DOCENTE(Email) ON
DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE COMPLETAMENTO (
    NumeroProgressivo INT auto_increment,
    Stato ENUM("Aperto","InCompletamento","Concluso") NOT NULL,
    TitoloTest VARCHAR(100)NOT NULL,
    EmailStudente VARCHAR(100)NOT NULL,
    DataPrimaRisposta DATETIME,
    DataUltimaRisposta DATETIME,

    PRIMARY KEY(NumeroProgressivo),

    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE,
    FOREIGN KEY(EmailStudente) REFERENCES STUDENTE(Email) ON DELETE
CASCADE

) ENGINE = INNODB;
```

```
CREATE TABLE TABELLADIESERCIZIO (
    Nome VARCHAR(20) PRIMARY KEY,
    DataCreazione DATETIME,
    num_righe INT,
    EmailDocente VARCHAR(100),
    FOREIGN KEY(EmailDocente) REFERENCES DOCENTE(Email) ON DELETE
CASCADE

) ENGINE = INNODB;

CREATE TABLE RIGA(
    Testo VARCHAR(255),
    NomeTabella VARCHAR(20),

    PRIMARY KEY(Testo,NomeTabella),
    FOREIGN KEY(NomeTabella) REFERENCES TABELLADIESERCIZIO(Nome) ON
DELETE CASCADE
);

CREATE TABLE ATTRIBUTO (
    NomeTabella VARCHAR(20) NOT NULL,
    NomeAttributo VARCHAR(100) NOT NULL,
    Tipo VARCHAR(30) NOT NULL,
    chiavePrimaria BOOLEAN,

    PRIMARY KEY(NomeAttributo, NomeTabella),

    FOREIGN KEY(NomeTabella) REFERENCES TABELLADIESERCIZIO(Nome) ON
DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE VINCOLODIINTEGRITA  (
    NomeTabellaUno VARCHAR(20) NOT NULL,
    NomeAttributoUno VARCHAR(100) NOT NULL,
    NomeTabellaDue VARCHAR(20) NOT NULL,
    NomeAttributoDue VARCHAR(100) NOT NULL,

    PRIMARY KEY(NomeTabellaUno, NomeAttributoUno, NomeTabellaDue,
NomeAttributoDue),
```

```
    FOREIGN KEY(NomeTabellaUno) REFERENCES TABELLADIESERCIZIO(Nome) ON
DELETE CASCADE,
    FOREIGN KEY(NomeAttributoUno) REFERENCES ATTRIBUTO(NomeAttributo)
ON DELETE CASCADE,
    FOREIGN KEY(NomeTabellaDue) REFERENCES TABELLADIESERCIZIO(Nome) ON
DELETE CASCADE,
    FOREIGN KEY(NomeAttributoDue) REFERENCES ATTRIBUTO(NomeAttributo)
ON DELETE CASCADE

)  ENGINE=INNODB;

CREATE TABLE QUESITO (
    NumeroProgressivo INT auto_increment,
    TitoloTest VARCHAR(100) NOT NULL,
    LivelloDifficolta ENUM("Basso","Medio","Alto"),
    Descrizione VARCHAR(255),
    NumeroRisposte INT,

    PRIMARY KEY(NumeroProgressivo,TitoloTest),

    FOREIGN KEY(TitoloTest) REFERENCES TEST(Titolo) ON DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE QUESITORISPOSTACHIUSA (
    NumeroProgressivo INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,

    PRIMARY KEY(NumeroProgressivo, TitoloTest),

    FOREIGN KEY(TitoloTest, NumeroProgressivo) REFERENCES
QUESITO(TitoloTest, NumeroProgressivo) ON DELETE CASCADE
) ENGINE = INNODB;

CREATE TABLE QUESITOCODICE (
    NumeroProgressivo INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,

    PRIMARY KEY(TitoloTest, NumeroProgressivo),

   FOREIGN KEY(TitoloTest,NumeroProgressivo) REFERENCES
QUESITO(TitoloTest,NumeroProgressivo) ON DELETE CASCADE
) ENGINE = INNODB;
```

```
CREATE TABLE SOLUZIONE (
    NumeroProgressivo INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    TestoSoluzione VARCHAR(500),

    PRIMARY KEY(TitoloTest, NumeroProgressivo, TestoSoluzione),

   FOREIGN KEY(TitoloTest, NumeroProgressivo) REFERENCES
QUESITOCODICE(TitoloTest,NumeroProgressivo) ON DELETE CASCADE
) ENGINE = INNODB;

CREATE TABLE RISPOSTAQUESITORISPOSTACHIUSA  (
    NumeroProgressivoCompletamento INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    OpzioneScelta VARCHAR(200),
    NumeroProgressivoQuesito INT,
    Esito BOOLEAN,

    PRIMARY KEY (NumeroProgressivoCompletamento, TitoloTest,
NumeroProgressivoQuesito),

    FOREIGN KEY(NumeroProgressivoCompletamento) REFERENCES
COMPLETAMENTO(NumeroProgressivo) ON DELETE CASCADE,
    FOREIGN KEY(TitoloTest, NumeroProgressivoQuesito) REFERENCES
QUESITORISPOSTACHIUSA(TitoloTest, NumeroProgressivo) ON DELETE CASCADE

)  ENGINE=INNODB;

CREATE TABLE RISPOSTAQUESITOCODICE  (
    NumeroProgressivoCompletamento INT NOT NULL,
    TitoloTest VARCHAR(100) NOT NULL,
    Testo VARCHAR(500),
    NumeroProgressivoQuesito INT,
    Esito BOOLEAN,

    PRIMARY KEY (NumeroProgressivoCompletamento, TitoloTest,
NumeroProgressivoQuesito),

    FOREIGN KEY(NumeroProgressivoCompletamento) REFERENCES
COMPLETAMENTO(NumeroProgressivo) ON DELETE CASCADE,
    FOREIGN KEY(TitoloTest, NumeroProgressivoQuesito) REFERENCES
QUESITOCODICE(TitoloTest,NumeroProgressivo) ON DELETE CASCADE
```

```
)  ENGINE=INNODB;

CREATE TABLE OPZIONERISPOSTA (
    NumeroProgressivoOpzione INT auto_increment,
    TitoloTest VARCHAR(100) NOT NULL,
    NumeroProgressivoQuesito INT NOT NULL,
    CampoTesto VARCHAR(200),
    RispostaCorretta BOOLEAN,

    PRIMARY KEY (NumeroProgressivoOpzione, NumeroProgressivoQuesito,
TitoloTest),

    FOREIGN KEY(TitoloTest,NumeroProgressivoQuesito) REFERENCES
QUESITORISPOSTACHIUSA(TitoloTest,NumeroProgressivo) ON DELETE CASCADE
)  ENGINE=INNODB;

CREATE TABLE COSTITUZIONE (
    TitoloTest VARCHAR(100) NOT NULL,
    NumeroProgressivoQuesito INT NOT NULL,
    NomeTabella VARCHAR(20) NOT NULL,

    PRIMARY KEY(TitoloTest, NumeroProgressivoQuesito, NomeTabella),

    FOREIGN KEY(TitoloTest,NumeroProgressivoQuesito) REFERENCES
QUESITO(TitoloTest,NumeroProgressivo) ON DELETE CASCADE,
    FOREIGN KEY(NomeTabella) REFERENCES TABELLADIESERCIZIO(Nome) ON
DELETE CASCADE

)  ENGINE=INNODB;
```

Stored Procedure

```sql
USE ESQL;

DROP PROCEDURE IF EXISTS VisualizzaDocenti;
DROP PROCEDURE IF EXISTS VisualizzaTestDisponibili;
DROP PROCEDURE IF EXISTS VisualizzaQuesitiPerTest;
DROP PROCEDURE IF EXISTS AutenticazioneDocente;
DROP PROCEDURE IF EXISTS AutenticazioneStudente;
DROP PROCEDURE IF EXISTS RegistrazioneDocente;
DROP PROCEDURE IF EXISTS RegistrazioneStudente;
DROP PROCEDURE IF EXISTS CreazioneTabellaEsercizio;
DROP PROCEDURE IF EXISTS ModificaVisualizzazioneRisposte;
DROP PROCEDURE IF EXISTS CreazioneTest;
DROP PROCEDURE IF EXISTS CreazioneQuesitoRispostaChiusa;
DROP PROCEDURE IF EXISTS CreazioneQuesitoCodice;
DROP PROCEDURE IF EXISTS CreazioneCostituzione;
DROP PROCEDURE IF EXISTS InserimentoSoluzione;
DROP PROCEDURE IF EXISTS InserimentoOpzioneRisposta;
DROP PROCEDURE IF EXISTS SetOpzioneRispostaCorretta;
DROP PROCEDURE IF EXISTS InserimentoMessaggioDocente;
DROP PROCEDURE IF EXISTS inserisciRispostaQuesitoCodice;
DROP PROCEDURE IF EXISTS inserisciRispostaQuesitoRispostaChiusa;
DROP PROCEDURE IF EXISTS visualizzaEsitoRisposta;
DROP PROCEDURE IF EXISTS inserisciMessaggioStudente;
DROP PROCEDURE IF EXISTS eliminaTest;
DROP PROCEDURE IF EXISTS eliminaQuesito;
DROP PROCEDURE IF EXISTS verificaPresenzaCollegamento;
DROP PROCEDURE IF EXISTS ricezioneMessaggiStudente;



-- PROCEDURE PER TUTTI GLI UTENTI
DELIMITER //
CREATE PROCEDURE VisualizzaDocenti ()
BEGIN
    SELECT * FROM DOCENTE;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE VisualizzaTestDisponibili ()
```

```
BEGIN
    SELECT * FROM TEST;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE VisualizzaQuesitiPerTest (
    IN TitoloTestTemp VARCHAR(100)
    )
BEGIN
    -- Seleziona i quesiti corrispondenti al titolo del test
specificato
    SELECT * FROM QUESITO WHERE TitoloTest = TitoloTestTemp;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE AutenticazioneDocente (
    IN EmailTemp VARCHAR(100),
    IN PasswordTemp VARCHAR(20),
    OUT AutenticatoTemp BOOLEAN
)
BEGIN
    -- Verifica se l'email esiste nella tabella Utenti e corrisponde
alla password fornita
    IF EXISTS (SELECT * FROM DOCENTE WHERE Email = EmailTemp AND
PasswordDocente = PasswordTemp) THEN
        SET AutenticatoTemp = TRUE;
    ELSE
        SET AutenticatoTemp = FALSE;
    END IF;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE AutenticazioneStudente (
    IN EmailTemp VARCHAR(100),
    IN PasswordTemp VARCHAR(20),
    OUT AutenticatoTemp BOOLEAN
)
```

```
BEGIN
    -- Verifica se l'email esiste nella tabella Utenti e corrisponde
alla password fornita
    IF EXISTS (SELECT * FROM STUDENTE WHERE Email = EmailTemp AND
PasswordStudente = PasswordTemp) THEN
        SET AutenticatoTemp = TRUE;
    ELSE
        SET AutenticatoTemp = FALSE;
    END IF;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE RegistrazioneDocente (
    IN EmailTemp VARCHAR(100),
    IN PasswordTemp VARCHAR(20),
    IN Nome VARCHAR (50),
    IN Cognome VARCHAR (50),
    IN RecapitoTelefonicoDocente INT,
    IN NomeDipartimento VARCHAR(100),
    IN NomeCorso VARCHAR(100)
)
BEGIN
    -- Verifica se l'email non esiste già nella tabella Docente
    IF NOT EXISTS (SELECT * FROM Docente WHERE Email = EmailTemp) THEN
        -- Inserisce l'utente nella tabella Utenti
        INSERT INTO Docente
(Email,PasswordDocente,Nome,Cognome,RecapitoTelefonicoDocente,NomeDipar
timento,NomeCorso) VALUES
(EmailTemp,PasswordTemp,Nome,Cognome,RecapitoTelefonicoDocente,NomeDipa
rtimento,NomeCorso);
    ELSE
     -- Se l'email esiste già, restituisce un messaggio di errore
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "L\'email inserita è
già presente nella tabella Docente";
    END IF;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE RegistrazioneStudente (
    IN EmailTemp VARCHAR(100),
```

```
    IN PasswordTemp VARCHAR(20),
    IN Nome VARCHAR (50),
    IN Cognome VARCHAR (50),
    IN RecapitoTelefonicoStudente INT,
    IN AnnoImmatricolazione INT,
    IN CodiceAlfaNumerico CHAR(16)
)
BEGIN
    -- Verifica se l'email non esiste già nella tabella Utenti
    IF NOT EXISTS (SELECT * FROM Studente WHERE Email = EmailTemp) THEN
        -- Inserisce l'utente nella tabella Utenti
        INSERT INTO STUDENTE
(Email,PasswordStudente,Nome,Cognome,RecapitoTelefonicoStudente,AnnoImm
atricolazione,CodiceAlfaNumerico) VALUES
(EmailTemp,PasswordTemp,Nome,Cognome,RecapitoTelefonicoStudente,AnnoImm
atricolazione,CodiceAlfaNumerico);
    ELSE
     -- Se l'email esiste già, restituisce un messaggio di errore
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "L\'email inserita è
già presente nella tabella Studente";
    END IF;
END //
DELIMITER ;



-- PROCEDURE PER I DOCENTI

DELIMITER //
CREATE PROCEDURE CreazioneTabellaEsercizio (
    IN nomeTabella VARCHAR(20),
    IN dataCreazione DATETIME,
    IN numRighe INT,
    IN emailDocente VARCHAR(100)
)
BEGIN
-- controllo che la tabella non esista già e che esista il docente
DECLARE tabellaNonEsistente INT DEFAULT 0;
DECLARE docenteEsistente INT DEFAULT 0;
SET tabellaNonEsistente = ( SELECT COUNT(*) FROM TABELLADIESERCIZIO
WHERE (nomeTabella=TABELLADIESERCIZIO.Nome) );
SET docenteEsistente = ( SELECT COUNT(*) FROM DOCENTE WHERE
(emailDocente = DOCENTE.Email) );
```

```
-- se non esiste la tabella ed esiste il docente la inserisco
IF (TabellaNonEsistente = 0 AND docenteEsistente=1) THEN
    INSERT INTO TABELLADIESERCIZIO VALUES(NomeTabella, dataCreazione,
numRighe, emailDocente);
END IF;

END
// DELIMITER ;


DELIMITER //
CREATE PROCEDURE ModificaVisualizzazioneRisposte (
    IN TitoloTestTemp VARCHAR(100),
    IN ValoreTemp BOOLEAN
)
BEGIN
    -- Imposta il campo VisualizzaRisposte al valore specificato per il
test specificato
    UPDATE Test SET VisualizzaRisposte = ValoreTemp WHERE Titolo =
TitoloTestTemp;
END
// DELIMITER ;



DELIMITER //
CREATE PROCEDURE CreazioneTest (
    IN TitoloTest VARCHAR(100),
    IN DataCreazione datetime,
    IN Foto VARCHAR(255),
    IN VisualizzaRisposte BOOLEAN,
    IN EmailDocente VARCHAR(100)
)
BEGIN
    DECLARE docenteEsistente INT DEFAULT 0;
    DECLARE TestNonEsistente INT DEFAULT 0;
    SET docenteEsistente = ( SELECT COUNT(*) FROM DOCENTE WHERE
(EmailDocente=DOCENTE.Email));
    SET TestNonEsistente = ( SELECT COUNT(*) FROM Test WHERE
(TitoloTest=TEST.Titolo));
-- se il docente esiste, e il test non esiste, inserisce i dati
IF (docenteEsistente = 1 AND TestNonEsistente = 0) THEN
```

```
    INSERT INTO TEST VALUES (TitoloTest, DataCreazione, Foto,
VisualizzaRisposte, EmailDocente);
END IF;
END
// DELIMITER ;


DELIMITER //
CREATE PROCEDURE CreazioneQuesitoRispostaChiusa (
    IN TitoloTestTemp VARCHAR(100),
    IN LivelloDifficoltaTemp ENUM("Basso","Medio","Alto"),
    IN DescrizioneTemp VARCHAR(255),
    OUT numProgressivo INT
)
BEGIN
    DECLARE TestEsistente INT DEFAULT 0;
    DECLARE UltimoNumeroProgressivo INT;
    SET TestEsistente = (SELECT COUNT(*) FROM TEST WHERE Titolo =
TitoloTestTemp);
    IF (TestEsistente = 1) THEN
        INSERT INTO QUESITO(TitoloTest, LivelloDifficolta, Descrizione,
NumeroRisposte)
        VALUES (TitoloTestTemp, LivelloDifficoltaTemp, DescrizioneTemp,
0);

        SET UltimoNumeroProgressivo = (SELECT MAX(NumeroProgressivo)
FROM QUESITO WHERE TitoloTest = TitoloTestTemp);
        INSERT INTO QUESITORISPOSTACHIUSA(NumeroProgressivo,
TitoloTest) VALUES (UltimoNumeroProgressivo, TitoloTestTemp);

        SET numProgressivo = UltimoNumeroProgressivo;
    END IF;

END //
DELIMITER ;



DELIMITER //
CREATE PROCEDURE CreazioneQuesitoCodice (
    IN TitoloTestTemp VARCHAR(100),
    IN LivelloDifficoltaTemp ENUM("Basso","Medio","Alto"),
    IN DescrizioneTemp VARCHAR(255),
```

```
    OUT numProgressivo INT
)
BEGIN
    DECLARE UltimoNumeroProgressivo INT;
    DECLARE TestEsistente INT DEFAULT 0;
    SET TestEsistente = (SELECT COUNT(*) FROM TEST WHERE
(TitoloTestTemp = Titolo));

    IF (TestEsistente = 1) THEN
        INSERT INTO QUESITO(TitoloTest, LivelloDifficolta, Descrizione,
NumeroRisposte)
        VALUES (TitoloTestTemp, LivelloDifficoltaTemp, DescrizioneTemp,
0);

        SET UltimoNumeroProgressivo = (SELECT MAX(NumeroProgressivo)
FROM QUESITO WHERE TitoloTest = TitoloTestTemp);
        INSERT INTO QUESITOCODICE(TitoloTest,NumeroProgressivo) VALUES
(TitoloTestTemp, UltimoNumeroProgressivo);

        SET numProgressivo = UltimoNumeroProgressivo;
    END IF;

END
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE CreazioneCostituzione(
    IN numero_temp_progressivoQuesito INT,
    IN titoloTest_temp VARCHAR(100),
    IN nome_temp_tabellaEsercizio VARCHAR(20)
)
BEGIN
    DECLARE TabellaEsistente INT DEFAULT 0;
    DECLARE QuesitoEsistente INT DEFAULT 0;
    SET TabellaEsistente = (SELECT COUNT(*) FROM TABELLADIESERCIZIO
WHERE (nome=nome_temp_tabellaEsercizio));
    SET QuesitoEsistente = (SELECT COUNT(*) FROM QUESITO WHERE
(numeroProgressivo=numero_temp_progressivoQuesito)
                                                        AND
(titoloTest=titoloTest_temp));
    IF (TabellaEsistente = 1 AND QuesitoEsistente = 1) THEN
        INSERT INTO COSTITUZIONE(TitoloTest, NumeroProgressivoQuesito,
NomeTabella)
```

```
        VALUES (titoloTest_temp, numero_temp_progressivoQuesito,
nome_temp_tabellaEsercizio);
    END IF;
END
// DELIMITER ;



DELIMITER //
CREATE PROCEDURE InserimentoSoluzione (
    IN TitoloTestTemp VARCHAR(100),
    IN NumeroProgressivoTemp INT,
    IN TestoSoluzioneTemp VARCHAR(500)
)
BEGIN
    DECLARE ProgressivoETestEsistenti INT DEFAULT 0;
    SET ProgressivoETestEsistenti = (SELECT COUNT(*) FROM QUESITOCODICE
    WHERE (NumeroProgressivo=NumeroProgressivoTemp AND
TitoloTestTemp=TitoloTest));

    IF (ProgressivoETestEsistenti=1) THEN
    INSERT INTO SOLUZIONE(NumeroProgressivo, TitoloTest,
TestoSoluzione)
    VALUES (NumeroProgressivoTemp, TitoloTestTemp, TestoSoluzioneTemp);
    END IF;
END
// DELIMITER ;



DELIMITER //
CREATE PROCEDURE InserimentoOpzioneRisposta (
    IN TitoloTestTemp VARCHAR(100),
    IN NumeroProgressivoQuesitoTemp INT,
    IN CampoTestoTemp VARCHAR(200),
    IN RispostaCorrettaTemp BOOLEAN
)
BEGIN
    DECLARE ProgressivoQuesitoETestEsistente INT DEFAULT 0;

    SET ProgressivoQuesitoETestEsistente = (SELECT COUNT(*) FROM
QUESITORISPOSTACHIUSA
```

```
    WHERE (TitoloTestTemp=TitoloTest AND NumeroProgressivo =
NumeroProgressivoQuesitoTemp));

    IF (ProgressivoQuesitoETestEsistente = 1) THEN
        INSERT INTO OPZIONERISPOSTA(TitoloTest,
NumeroProgressivoQuesito, CampoTesto, RispostaCorretta)
        VALUES (TitoloTestTemp, NumeroProgressivoQuesitoTemp,
CampoTestoTemp, RispostaCorrettaTemp);
    END IF;
END //
DELIMITER ;



DELIMITER //
CREATE PROCEDURE SetOpzioneRispostaCorretta (
    IN TitoloTestTemp VARCHAR(100),
    IN NumeroProgressivoQuesitoTemp INT,
    IN CampoTestoTemp VARCHAR(200)
)
BEGIN
    UPDATE OPZIONERISPOSTA
    SET rispostaCorretta = true
    WHERE ((TitoloTestTemp=TitoloTest) AND
(NumeroProgressivoQuesitoTemp=NumeroProgressivoQuesito)
        AND (CampoTestoTemp=CampoTesto));

END
// DELIMITER ;



DELIMITER //
CREATE PROCEDURE InserimentoMessaggioDocente(
    IN TitoloTest_t VARCHAR(100),
    IN TitoloMessaggio_t VARCHAR(100),
    IN CampoTesto_t VARCHAR(500),
    IN Data_t DATETIME,
    IN EmailDocenteMittente_t VARCHAR(100)
)
BEGIN
    DECLARE TestEsistente INT DEFAULT 0;
    DECLARE DocenteEsistente INT DEFAULT 0;
```

```
    DECLARE IdMessaggio INT;
    DECLARE done INT DEFAULT FALSE;
    DECLARE student_email VARCHAR(100);
    DECLARE student_cursor CURSOR FOR SELECT Email FROM STUDENTE;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- mi assicuro che esistano il test e il docente
    SET TestEsistente = (SELECT COUNT(*) FROM TEST WHERE Titolo =
TitoloTest_t);
    SET DocenteEsistente = (SELECT COUNT(*) FROM DOCENTE WHERE Email =
EmailDocenteMittente_t);

    IF (TestEsistente = 1 AND DocenteEsistente = 1) THEN
        -- Inserisce il messaggio nella tabella MESSAGGIO
        INSERT INTO MESSAGGIO (TitoloTest, TitoloMessaggio, CampoTesto,
Data) VALUES (TitoloTest_t, TitoloMessaggio_t, CampoTesto_t, Data_t);

        -- Ottiene l'ID del messaggio appena inserito
        SET IdMessaggio = LAST_INSERT_ID();

        -- Inserisce il messaggio nella tabella INVIODOCENTE per ogni
docente
        INSERT INTO INVIODOCENTE (Id, TitoloTest, EmailDocenteMittente)
VALUES (IdMessaggio, TitoloTest_t, EmailDocenteMittente_t);

        OPEN student_cursor;

        -- Ciclo per inserire il messaggio nella tabella
RICEZIONESTUDENTE per ogni studente
        message_loop: LOOP
            FETCH student_cursor INTO student_email;
            IF done THEN
                LEAVE message_loop;
            END IF;
            -- Inserisce il messaggio nella tabella RICEZIONESTUDENTE
per lo studente corrente
            INSERT INTO RICEZIONESTUDENTE (Id, TitoloTest,
EmailStudenteDestinatario) VALUES (IdMessaggio, TitoloTest_t,
student_email);
        END LOOP;

        CLOSE student_cursor;
```

```
    END IF;
END //
DELIMITER ;




-- PROCEDURE PER GLI STUDENTI

DELIMITER //

CREATE PROCEDURE inserisciRispostaQuesitoCodice(
    IN idCompletamentoTemp INT,
    IN TitoloTestTemp VARCHAR(100),
    IN valoreRispostaTemp VARCHAR(500),
    IN numeroQuesitoTemp INT,
    IN esitoRisposta BOOLEAN
)
BEGIN

    DECLARE numRispostaAperta INT;
    DECLARE num_risposte INT;


    -- Controlla se esiste il quesito
    SELECT COUNT(*) INTO numRispostaAperta
    FROM QUESITOCODICE AS QC
    WHERE (QC.NumeroProgressivo = numeroQuesitoTemp) AND (QC.TitoloTest
IN (SELECT C1.TitoloTest

FROM COMPLETAMENTO AS C1

WHERE (idCompletamentoTemp = C1.NumeroProgressivo)));



    IF numRispostaAperta = 1 THEN




        -- Controllo se è già presente una risposta al quesito
        SELECT COUNT(*) INTO num_risposte
```

```
        FROM RISPOSTAQUESITOCODICE
        WHERE NumeroProgressivoCompletamento = idCompletamentoTemp AND
TitoloTest = TitoloTestTemp AND NumeroProgressivoQuesito =
numeroQuesitoTemp;



        IF (num_risposte = 0) THEN
            INSERT INTO
RISPOSTAQUESITOCODICE(NumeroProgressivoCompletamento, TitoloTest,
Testo, NumeroProgressivoQuesito, Esito)
            VALUES (idCompletamentoTemp, TitoloTestTemp,
valoreRispostaTemp, numeroQuesitoTemp, esitoRisposta);


        ELSE
            UPDATE RISPOSTAQUESITOCODICE
            SET Testo = valoreRispostaTemp, Esito = esitoRisposta
            WHERE NumeroProgressivoCompletamento = idCompletamentoTemp
            AND TitoloTest = TitoloTestTemp
            AND NumeroProgressivoQuesito = numeroQuesitoTemp;


        END IF;

    END IF;

END//

DELIMITER ;


DELIMITER //

CREATE PROCEDURE inserisciRispostaQuesitoRispostaChiusa(
    IN idCompletamentoTemp INT,
    IN TitoloTestTemp VARCHAR(100),
    IN valoreRispostaTemp VARCHAR(200),
    IN numeroQuesitoTemp INT
)
BEGIN
    DECLARE numRispostaChiusa INT;
    DECLARE esitoRisposta BOOLEAN;
```

```
    DECLARE rispostaCorretta VARCHAR(200);
    DECLARE num_risposte INT;




    -- Controlla se esiste la risposta
    SELECT COUNT(*) INTO numRispostaChiusa
    FROM QUESITORISPOSTACHIUSA AS QC
    WHERE (QC.NumeroProgressivo = numeroQuesitoTemp) AND (QC.TitoloTest
IN (SELECT C1.TitoloTest

FROM COMPLETAMENTO AS C1

WHERE (idCompletamentoTemp = C1.NumeroProgressivo)));



     IF (numRispostaChiusa = 1) THEN
        SET esitoRisposta = FALSE;


        SELECT CampoTesto INTO rispostaCorretta
        FROM OPZIONERISPOSTA AS OP
        WHERE (OP.RispostaCorretta = TRUE) AND
(OP.NumeroProgressivoQuesito = numeroQuesitoTemp) AND (OP.TitoloTest IN
(SELECT C1.TitoloTest

FROM COMPLETAMENTO AS C1

WHERE (idCompletamentoTemp = C1.NumeroProgressivo)));

        IF (valoreRispostaTemp = rispostaCorretta) THEN
            SET esitoRisposta = TRUE;
        END IF;



        -- Controllo se è già presente una risposta al quesito
        SELECT COUNT(*) INTO num_risposte
        FROM RISPOSTAQUESITORISPOSTACHIUSA
```

```
        WHERE NumeroProgressivoCompletamento = idCompletamentoTemp AND
TitoloTest = TitoloTestTemp AND NumeroProgressivoQuesito =
numeroQuesitoTemp;



        IF (num_risposte = 0) THEN
            INSERT INTO
RISPOSTAQUESITORISPOSTACHIUSA(NumeroProgressivoCompletamento,
TitoloTest, OpzioneScelta, NumeroProgressivoQuesito, Esito)
            VALUES (idCompletamentoTemp, TitoloTestTemp,
valoreRispostaTemp, numeroQuesitoTemp, esitoRisposta);


        ELSE
            UPDATE RISPOSTAQUESITORISPOSTACHIUSA
            SET OpzioneScelta = valoreRispostaTemp, Esito =
esitoRisposta
            WHERE NumeroProgressivoCompletamento = idCompletamentoTemp
            AND TitoloTest = TitoloTestTemp
            AND NumeroProgressivoQuesito = numeroQuesitoTemp;


        END IF;

      END IF;
END

//
DELIMITER ;


DELIMITER //

CREATE PROCEDURE visualizzaEsitoRisposta(
    IN idCompletamentoTemp INT,
    IN TitoloTestTemp VARCHAR(100),
    IN numQuesito INT,
    OUT esitoRisposta BOOLEAN
)
BEGIN
    DECLARE esitoTemp BOOLEAN;
```

```
    -- Verifica se esiste una risposta per il quesito di tipo "Risposta
Chiusa"
    IF EXISTS (
        SELECT 1
        FROM RISPOSTAQUESITORISPOSTACHIUSA AS RC
        WHERE RC.NumeroProgressivoQuesito = numQuesito AND
RC.TitoloTest = TitoloTestTemp AND RC.NumeroProgressivoCompletamento =
idCompletamentoTemp
    ) THEN


        SELECT esito INTO esitoTemp
        FROM RISPOSTAQUESITORISPOSTACHIUSA AS RC
        WHERE RC.NumeroProgressivoQuesito = numQuesito AND
RC.TitoloTest = TitoloTestTemp AND  RC.NumeroProgressivoCompletamento =
idCompletamentoTemp
        LIMIT 1; -- Assicura che venga restituita una sola riga


    END IF;

    -- Verifica se esiste una risposta per il quesito di tipo "Codice"
    IF EXISTS (
        SELECT 1
        FROM RISPOSTAQUESITOCODICE AS QC
        WHERE QC.NumeroProgressivoQuesito = numQuesito AND
QC.TitoloTest = TitoloTestTemp AND QC.NumeroProgressivoCompletamento =
idCompletamentoTemp
    ) THEN


        SELECT esito INTO esitoTemp
        FROM RISPOSTAQUESITOCODICE AS QC
        WHERE QC.NumeroProgressivoQuesito = numQuesito AND
QC.TitoloTest = TitoloTestTemp AND  QC.NumeroProgressivoCompletamento =
idCompletamentoTemp
        LIMIT 1; -- Assicura che venga restituita una sola riga


    END IF;
```

```
    -- Imposta il valore di esitoRisposta in base al valore di
esitoTemp
    IF esitoTemp IS NOT NULL THEN
        SET esitoRisposta = esitoTemp;
    ELSE
        SET esitoRisposta = NULL;
    END IF;

END//

DELIMITER ;




DELIMITER //
CREATE PROCEDURE inserisciMessaggioStudente(
    IN emailStudenteTemp VARCHAR(100),
    IN emailDocenteTemp VARCHAR(100),
    IN titoloTestTemp VARCHAR(100),
    IN titoloMess VARCHAR(100),
    IN testoMess VARCHAR(500)
)
BEGIN
    DECLARE IDMess INT;

    INSERT INTO MESSAGGIO (TitoloTest, TitoloMessaggio, CampoTesto,
Data)
    VALUES (titoloTestTemp, titoloMess, testoMess, NOW());

    -- salvo l'ID del messaggio
    SELECT Id INTO IDMess
    FROM MESSAGGIO
    WHERE (TitoloTest=titoloTestTemp) AND (TitoloMessaggio =
titoloMess) AND (testoMess = CampoTesto);

    -- Invio del messaggio a tutti i docenti
    INSERT INTO RICEZIONEDOCENTE VALUES (IDMess, titoloTestTemp,
emailDocenteTemp);

    -- Aggiornamento tabella INVIOSTUDENTE
```

```
    INSERT INTO INVIOSTUDENTE VALUES(IDMess, titoloTestTemp,
emailStudenteTemp);

END
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE eliminaTest(
    IN titoloTest VARCHAR(100)
)
BEGIN
    DELETE FROM TEST WHERE Titolo = titoloTest;
END
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE eliminaQuesito(
    IN titoloTestTemp VARCHAR(190),
    IN numeroProgressivoTemp INT
)
BEGIN
    DELETE FROM QUESITO WHERE TitoloTest = titoloTestTemp AND
NumeroProgressivo = numeroProgressivoTemp;
END
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE verificaPresenzaCollegamento(
    IN titoloTestTemp VARCHAR(100),
    IN numeroProgressivoTemp INT,
    OUT presente BOOLEAN
)
BEGIN
    SELECT COUNT(*) INTO presente FROM COSTITUZIONE WHERE TitoloTest =
titoloTestTemp AND NumeroProgressivoQuesito = numeroProgressivoTemp;
END
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE ricezioneMessaggiStudente(
    IN emailStudenteTemp VARCHAR(100)
)
BEGIN
```

```
    SELECT *
    FROM messaggio as M, ricezionestudente as S, invioDocente AS id
    WHERE (M.Id = S.Id) AND (emailStudenteTemp =
S.EmailStudenteDestinatario) AND (M.TitoloTest=S.TitoloTest) AND (id.Id
= M.Id);
END
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE messaggiInviatiDaStudente(
    IN emailStudenteTemp VARCHAR(100)
)
BEGIN
    SELECT *
    FROM messaggio as M, inviostudente as D, ricezioneDocente as RD
    WHERE (M.Id = D.Id) AND (D.EmailStudenteMittente =
emailStudenteTemp) AND (RD.Id = M.Id);
END
// DELIMITER ;
```

Trigger

```
USE ESQL;


DROP TRIGGER IF EXISTS
cambio_stato_incompletamento_rispostaquesitorispostachiusa;
DROP TRIGGER IF EXISTS
cambio_stato_incompletamento_rispostaquesitocodice;
DROP TRIGGER IF EXISTS
cambio_stato_incompletamento_rispostaquesitoRC_conUpd;
DROP TRIGGER IF EXISTS cambio_stato_incompletamento_rispostaQC_conUpd;
DROP TRIGGER IF EXISTS cambio_stato_concluso_rispostaquesitoRC_Agg;
DROP TRIGGER IF EXISTS cambio_stato_concluso_rispostaQuesitoC_Agg;
DROP TRIGGER IF EXISTS
cambio_stato_concluso_rispostaquesitorispostachiusa;
DROP TRIGGER IF EXISTS cambio_stato_concluso_rispostaquesitocodice;
DROP TRIGGER IF EXISTS testConclusoVisualizzaRisposte;
DROP TRIGGER IF EXISTS incrementaNumRighe;
DROP TRIGGER IF EXISTS decrementaNumRighe;
DROP TRIGGER IF EXISTS AggiornaNumeroRisposteQuesitoAfterInsert;
DROP TRIGGER IF EXISTS AggiornaNumeroRisposteQuesitoAfterDelete;
DROP TRIGGER IF EXISTS AggiornaNumeroRisposteQuesitoCodiceAfterInsert;
DROP TRIGGER IF EXISTS AggiornaNumeroRisposteQuesitoCodiceAfterDelete;
DROP TRIGGER IF EXISTS spaziTitoloTest;

DELIMITER //
CREATE TRIGGER
cambio_stato_incompletamento_rispostaquesitorispostachiusa
AFTER INSERT ON RISPOSTAQUESITORISPOSTACHIUSA
FOR EACH ROW
BEGIN
    DECLARE num_risposte_inserite INT;

    -- Conta quante risposte sono state inserite per lo studente per il
test corrente
```

```
    SET num_risposte_inserite = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE (TitoloTest = NEW.TitoloTest AND
NumeroProgressivoCompletamento = NEW.NumeroProgressivoCompletamento));

    -- Se il numero di risposte inserite è uguale a 1, cambia lo stato
del test in 'InCompletamento'
    IF (num_risposte_inserite >= 1) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'InCompletamento'
        WHERE TitoloTest = NEW.TitoloTest AND NumeroProgressivo =
NEW.NumeroProgressivoCompletamento;
    END IF;
END
// DELIMITER ;



DELIMITER //
CREATE TRIGGER cambio_stato_incompletamento_rispostaquesitocodice
AFTER INSERT ON RISPOSTAQUESITOCODICE
FOR EACH ROW
BEGIN
    DECLARE num_risposte_inserite INT;

    -- Conta quante risposte sono state inserite per lo studente per il
test corrente
    SET num_risposte_inserite = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
    WHERE TitoloTest = NEW.TitoloTest AND
NumeroProgressivoCompletamento = NEW.NumeroProgressivoCompletamento);

    -- Se il numero di risposte inserite è uguale a 1, cambia lo stato
del test in 'InCompletamento'
    IF (num_risposte_inserite >= 1) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'InCompletamento'
        WHERE TitoloTest = NEW.TitoloTest AND NumeroProgressivo =
NEW.NumeroProgressivoCompletamento;
    END IF;
END;
// DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER cambio_stato_incompletamento_rispostaquesitoRC_conUpd
AFTER UPDATE ON RISPOSTAQUESITORISPOSTACHIUSA
FOR EACH ROW
BEGIN
    DECLARE num_risposte_inserite INT;

    -- Conta quante risposte sono state inserite per lo studente per il
test corrente
    SET num_risposte_inserite = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE (TitoloTest = NEW.TitoloTest AND
NumeroProgressivoCompletamento = NEW.NumeroProgressivoCompletamento));

    -- Se il numero di risposte inserite è uguale a 1, cambia lo stato
del test in 'InCompletamento'
    IF (num_risposte_inserite >= 1) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'InCompletamento'
        WHERE TitoloTest = NEW.TitoloTest AND NumeroProgressivo =
NEW.NumeroProgressivoCompletamento;
    END IF;
END
// DELIMITER ;



DELIMITER //
CREATE TRIGGER cambio_stato_incompletamento_rispostaQC_conUpd
AFTER UPDATE ON RISPOSTAQUESITOCODICE
FOR EACH ROW
BEGIN
    DECLARE num_risposte_inserite INT;

    -- Conta quante risposte sono state inserite per lo studente per il
test corrente
    SET num_risposte_inserite = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
    WHERE TitoloTest = NEW.TitoloTest AND
NumeroProgressivoCompletamento = NEW.NumeroProgressivoCompletamento);
```

```
    -- Se il numero di risposte inserite è uguale a 1, cambia lo stato
del test in 'InCompletamento'
    IF (num_risposte_inserite >= 1) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'InCompletamento'
        WHERE TitoloTest = NEW.TitoloTest AND NumeroProgressivo =
NEW.NumeroProgressivoCompletamento;
    END IF;
END;
// DELIMITER ;



DELIMITER //
CREATE TRIGGER cambio_stato_concluso_rispostaquesitoRC_Agg
AFTER UPDATE ON RISPOSTAQUESITORISPOSTACHIUSA
FOR EACH ROW
BEGIN
    DECLARE num_quesiti_totali INT;
    DECLARE num_risposte_inserite_RC INT;
    DECLARE num_risposte_inserite_codice INT;
    DECLARE num_risposte_inserite INT;
    DECLARE num_risposte_corrette_RC INT;
    DECLARE num_risposte_corrette_codice INT;
    DECLARE num_risposte_corrette INT;
    DECLARE num_progressivo_completamento INT;

    -- Ottieni il numero progressivo di completamento
    SET num_progressivo_completamento =
NEW.NumeroProgressivoCompletamento;



    -- Conta il numero totale di quesiti per il test
    SET num_quesiti_totali = (SELECT COUNT(*) FROM QUESITO
    WHERE TitoloTest = NEW.TitoloTest);

    -- Conta il numero di risposte inserite per il test
    SET num_risposte_inserite_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);
```

```
    SET num_risposte_inserite_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
        WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite = num_risposte_inserite_codice +
num_risposte_inserite_RC;

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    SET num_risposte_corrette = num_risposte_corrette_codice +
num_risposte_corrette_RC;

    -- Se tutte le risposte sono state inserite e hanno esito True, il
test diventa Concluso
    IF (num_risposte_inserite = num_quesiti_totali AND
num_risposte_corrette = num_quesiti_totali) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'Concluso'
        WHERE NumeroProgressivo = num_progressivo_completamento;
    END IF;
END;
// DELIMITER ;



DELIMITER //
CREATE TRIGGER cambio_stato_concluso_rispostaQuesitoC_Agg
AFTER UPDATE ON RISPOSTAQUESITOCODICE
FOR EACH ROW
BEGIN
    DECLARE num_quesiti_totali INT;
    DECLARE num_risposte_inserite_RC INT;
```

```
    DECLARE num_risposte_inserite_codice INT;
    DECLARE num_risposte_inserite INT;
    DECLARE num_risposte_corrette_RC INT;
    DECLARE num_risposte_corrette_codice INT;
    DECLARE num_risposte_corrette INT;
    DECLARE num_progressivo_completamento INT;

    -- Ottiene il numero progressivo di completamento
    SET num_progressivo_completamento =
NEW.NumeroProgressivoCompletamento;

    -- Conta il numero totale di quesiti per il test
    SET num_quesiti_totali = (SELECT COUNT(*) FROM QUESITO
    WHERE TitoloTest = NEW.TitoloTest);

    -- Conta il numero di risposte inserite per il test
    SET num_risposte_inserite_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
        WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite = num_risposte_inserite_codice +
num_risposte_inserite_RC;

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    SET num_risposte_corrette = num_risposte_corrette_codice +
num_risposte_corrette_RC;
```

```
    -- Se tutte le risposte sono state inserite e hanno esito True, il
test diventa Concluso
    IF (num_risposte_inserite = num_quesiti_totali AND
num_risposte_corrette = num_quesiti_totali) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'Concluso'
        WHERE NumeroProgressivo = num_progressivo_completamento;
    END IF;
END;
// DELIMITER ;

DELIMITER //
CREATE TRIGGER cambio_stato_concluso_rispostaquesitorispostachiusa
AFTER INSERT ON RISPOSTAQUESITORISPOSTACHIUSA
FOR EACH ROW
BEGIN
    DECLARE num_quesiti_totali INT;
    DECLARE num_risposte_inserite_RC INT;
    DECLARE num_risposte_inserite_codice INT;
    DECLARE num_risposte_inserite INT;
    DECLARE num_risposte_corrette_RC INT;
    DECLARE num_risposte_corrette_codice INT;
    DECLARE num_risposte_corrette INT;
    DECLARE num_progressivo_completamento INT;

    -- Ottiene il numero progressivo di completamento
    SET num_progressivo_completamento =
NEW.NumeroProgressivoCompletamento;


    -- Conta il numero totale di quesiti per il test
    SET num_quesiti_totali = (SELECT COUNT(*) FROM QUESITO
    WHERE TitoloTest = NEW.TitoloTest);

    -- Conta il numero di risposte inserite per il test
    SET num_risposte_inserite_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
```

```
        WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite = num_risposte_inserite_codice +
num_risposte_inserite_RC;

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    SET num_risposte_corrette = num_risposte_corrette_codice +
num_risposte_corrette_RC;

    -- Se tutte le risposte sono state inserite e hanno esito True, il
test diventa Concluso
    IF (num_risposte_inserite = num_quesiti_totali AND
num_risposte_corrette = num_quesiti_totali) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'Concluso'
        WHERE NumeroProgressivo = num_progressivo_completamento;
    END IF;
END;
// DELIMITER ;



DELIMITER //
CREATE TRIGGER cambio_stato_concluso_rispostaquesitocodice
AFTER INSERT ON RISPOSTAQUESITOCODICE
FOR EACH ROW
BEGIN
    DECLARE num_quesiti_totali INT;
    DECLARE num_risposte_inserite_RC INT;
    DECLARE num_risposte_inserite_codice INT;
    DECLARE num_risposte_inserite INT;
```

```
    DECLARE num_risposte_corrette_RC INT;
    DECLARE num_risposte_corrette_codice INT;
    DECLARE num_risposte_corrette INT;
    DECLARE num_progressivo_completamento INT;

    -- Ottieni il numero progressivo di completamento
    SET num_progressivo_completamento =
NEW.NumeroProgressivoCompletamento;

    -- Conta il numero totale di quesiti per il test
    SET num_quesiti_totali = (SELECT COUNT(*) FROM QUESITO
    WHERE TitoloTest = NEW.TitoloTest);

    -- Conta il numero di risposte inserite per il test
    SET num_risposte_inserite_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
        WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento);

    SET num_risposte_inserite = num_risposte_inserite_codice +
num_risposte_inserite_RC;

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_RC = (SELECT COUNT(*) FROM
RISPOSTAQUESITORISPOSTACHIUSA
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    -- Conta il numero di risposte corrette per il test
    SET num_risposte_corrette_codice = (SELECT COUNT(*) FROM
RISPOSTAQUESITOCODICE
    WHERE NumeroProgressivoCompletamento =
num_progressivo_completamento AND Esito = TRUE);

    SET num_risposte_corrette = num_risposte_corrette_codice +
num_risposte_corrette_RC;
```

```
    -- Se tutte le risposte sono state inserite e hanno esito True, il
test diventa Concluso
    IF (num_risposte_inserite = num_quesiti_totali AND
num_risposte_corrette = num_quesiti_totali) THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'Concluso'
        WHERE NumeroProgressivo = num_progressivo_completamento;
    END IF;
END;
// DELIMITER ;




DELIMITER //
CREATE TRIGGER testConclusoVisualizzaRisposte
AFTER UPDATE ON TEST
FOR EACH ROW
BEGIN
    IF NEW.VisualizzaRisposte = TRUE THEN
        UPDATE COMPLETAMENTO
        SET Stato = 'Concluso'
        WHERE TitoloTest = NEW.Titolo;
    END IF;
END
// DELIMITER ;


DELIMITER //
CREATE TRIGGER incrementaNumRighe
AFTER INSERT ON RIGA
FOR EACH ROW
BEGIN
    UPDATE TABELLADIESERCIZIO
    SET num_righe = num_righe + 1
    WHERE Nome = NEW.NomeTabella;
END
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER decrementaNumRighe
AFTER DELETE ON RIGA
```

```
FOR EACH ROW
    -- Aggiorna il conteggio delle righe nella tabella
TABELLADIESERCIZIO
    UPDATE TABELLADIESERCIZIO
    SET num_righe = num_righe - 1
    WHERE Nome = OLD.NomeTabella;
//
DELIMITER ;


DELIMITER //
CREATE TRIGGER AggiornaNumeroRisposteQuesitoAfterInsert
AFTER INSERT ON RISPOSTAQUESITORISPOSTACHIUSA
FOR EACH ROW
BEGIN
    UPDATE QUESITO
    SET NumeroRisposte = NumeroRisposte + 1
    WHERE NumeroProgressivo = NEW.NumeroProgressivoQuesito AND
TitoloTest = NEW.TitoloTest;
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER AggiornaNumeroRisposteQuesitoAfterDelete
AFTER DELETE ON RISPOSTAQUESITORISPOSTACHIUSA
FOR EACH ROW
BEGIN
    UPDATE QUESITO
    SET NumeroRisposte = NumeroRisposte - 1
    WHERE NumeroProgressivo = OLD.NumeroProgressivoQuesito AND
TitoloTest = OLD.TitoloTest;
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER AggiornaNumeroRisposteQuesitoCodiceAfterInsert
AFTER INSERT ON RISPOSTAQUESITOCODICE
FOR EACH ROW
BEGIN
    UPDATE QUESITO
    SET NumeroRisposte = NumeroRisposte + 1
```

```
    WHERE NumeroProgressivo = NEW.NumeroProgressivoQuesito AND
TitoloTest = NEW.TitoloTest;
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER AggiornaNumeroRisposteQuesitoCodiceAfterDelete
AFTER DELETE ON RISPOSTAQUESITOCODICE
FOR EACH ROW
BEGIN
    UPDATE QUESITO
    SET NumeroRisposte = NumeroRisposte - 1
    WHERE NumeroProgressivo = OLD.NumeroProgressivoQuesito AND
TitoloTest = OLD.TitoloTest;
END;
//
DELIMITER ;



DELIMITER //

CREATE TRIGGER spaziTitoloTest
BEFORE INSERT ON TEST FOR EACH ROW
BEGIN
    DECLARE titoloModificato VARCHAR(100);
    SET titoloModificato = TRIM(NEW.Titolo); -- Rimuove spazi vuoti
all'inizio e alla fine
    SET titoloModificato = REPLACE(titoloModificato, ' ', '_'); --
Sostituisce gli spazi vuoti con '_'
    SET NEW.Titolo = titoloModificato; -- Aggiorna il titolo nel nuovo
valore pulito
END;

//

DELIMITER ;
```

View

```
USE ESQL;

DROP VIEW IF EXISTS classifica_test_completati;
DROP VIEW IF EXISTS classifica_risposte_corrette;
DROP VIEW IF EXISTS classifica_quesiti;

-- view nr 1
CREATE VIEW classifica_test_completati AS
    SELECT s.CodiceAlfaNumerico, COUNT(*) AS numero_test_completati
    FROM STUDENTE AS s LEFT JOIN COMPLETAMENTO AS c ON s.Email =
c.EmailStudente
    WHERE c.Stato = 'Concluso'
    GROUP BY s.CodiceAlfaNumerico
    ORDER BY numero_test_completati DESC;


-- view nr 2
CREATE VIEW classifica_risposte_corrette AS
    SELECT
        S.CodiceAlfaNumerico AS codiceStudente,
        IFNULL(((SUM(RQC.Esito) + SUM(RCC.Esito)) /
(COUNT(RQC.NumeroProgressivoCompletamento) +
COUNT(RCC.NumeroProgressivoCompletamento))) * 100, 0) AS
percentualeRisposteCorrette
    FROM
        COMPLETAMENTO AS C
    LEFT JOIN
        RISPOSTAQUESITORISPOSTACHIUSA AS RQC ON C.NumeroProgressivo =
RQC.NumeroProgressivoCompletamento
    LEFT JOIN
        RISPOSTAQUESITOCODICE AS RCC ON C.NumeroProgressivo =
RCC.NumeroProgressivoCompletamento
    LEFT JOIN
```

```
        STUDENTE AS S ON C.EmailStudente = S.Email
    GROUP BY
        C.EmailStudente
    ORDER BY
        percentualeRisposteCorrette DESC;



-- view nr 3
CREATE VIEW classifica_quesiti AS
    SELECT
        NumeroProgressivoQuesito,
        Q.Descrizione,
        COUNT(*) AS numRisposteInserite
    FROM
        (SELECT NumeroProgressivoQuesito FROM
RISPOSTAQUESITORISPOSTACHIUSA
        UNION ALL
        SELECT NumeroProgressivoQuesito FROM RISPOSTAQUESITOCODICE) AS
Risposte
    LEFT JOIN QUESITO AS Q ON Risposte.NumeroProgressivoQuesito =
Q.NumeroProgressivo
    GROUP BY
        NumeroProgressivoQuesito
    ORDER BY
        numRisposteInserite DESC;
```