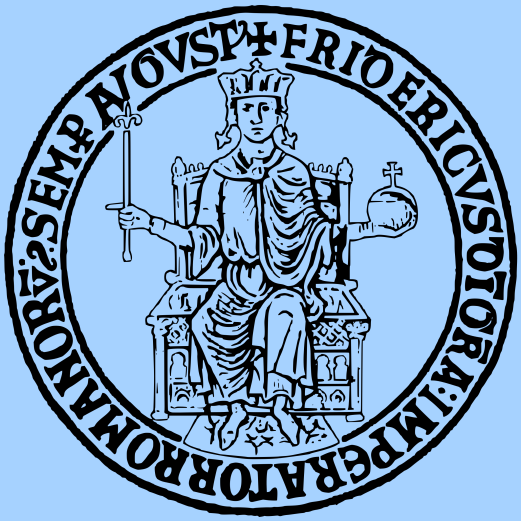
UNIVERSITA' DEGLI STUDI DI NAPOLI "FEDERICO II"
Scuola Politecnica e Delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione



Corso di Laurea in Informatica
Progetto di Basi di Dati
A.A. 2021/2022

Progettazione e sviluppo di una Base di dati relazionale per la gestione di Test basati su Quiz

Antonio Lanuto **Erasmo Prosciutto**

N86003762 N86003546

24/11/2021

Questa pagina è stata intenzionalmente lasciata in bianco

Indice

Capitolo 1

Descrizione e Analisi del Progetto

1.1 Descrizione sintetica e analisi del problema

Si vuole sviluppare un sistema informativo composto da una base di dati relazionale per la gestione di Test basati su Quiz.
Un Test consiste di un insieme di Quiz che deve essere fissato alla creazione. Ogni Test è caratterizzato da un Nome univoco che lo identifica.
Esistono due tipi di **Quiz**: quelli a risposta multipla e quelli a risposta aperta.
Un **Quiz a risposta multipla** è caratterizzato da una domanda, un elenco di possibili risposte, il punteggio da assegnare in caso di risposta esatta e il punteggio (anche negativo) da assegnare in caso di risposta errata.
Un **Quiz a risposta aperta** è invece caratterizzato, oltre che da un breve testo che descrive la domanda posta, dalla massima lunghezza prevista per il testo di risposta e da un punteggio minimo e massimo che il docente potrà assegnare in base alla correttezza della risposta.
Uno studente potrà sostenere un Test fornendo una risposta tra quelle proposte per ogni Quiz a risposta multipla e un testo per ogni Quiz a risposta aperta.
Il docente provvederà alla valutazione di tutte le risposte inserite, assegnando un punteggio compreso tra il minimo e il massimo previsti per quel Quiz.
Lo studente avrà infine la possibilità di visualizzare i **Risultati Finali** per ogni Test che sarà stato completato e successivamente valutato dal docente.

1.2 Identificazione e Analisi dei Requisiti:

Sono state individuate le seguenti Entità:

1. **Persona**: Generalizzazione di uno Studente e di un Docente.
2. **Studente**: Specializzazione di una Persona.
3. **Docente**: Specializzazione di una Persona.
4. **Test**: Entità atta ad identificare la conformazione di un Test.

5. **Risultato Test**: Entità atta ad identificare le valutazioni finali ottenute in un Test.
6. **Gestione**: Association Class atta ad identificare le molteplici interazioni tra Docente e Test (*Spiegazione approfondita in: **"Scelte Progettuali"**, chapter 1 - section 3*).
7. **Quiz**: Generalizzazione di un Quiz a risposta aperta e di un Quiz a risposta multipla.
8. **Quiz a risposta aperta**: Specializzazione di un Quiz.
9. **Quiz a risposta multipla**: Specializzazione di un Quiz.
10. **Valutazione**: Association Class atta a valutare, attraverso l'assegnazione di un Punteggio, un Quiz a risposta aperta.

1.3 Scelte Progettuali

Al fine di creare un sistema quanto più scalabile e aperto a future implementazioni/modifiche, si è deciso di implementare alcune soluzioni aggiuntive alla traccia proposta, e di strutturare la progettazione in modo anche da permettere all'utenza una completa personalizzazione e gestione.

In particolare:

- Ad un **Test** sarà possibile assegnare, oltre ad un Nominativo, anche una Tipologia, per stabilirne le proprietà (Test con Quiz a risposta multipla, a risposta aperta, misto), un Limite di Tempo, per stabilire le tempistiche a disposizione per il completamento del Test da parte dello studente, un Numero Quiz, per permettere al docente di stabilire precisamente quanti Quiz inserire in un Test e una Data di Scadenza, che permette al docente di stabilire giorno/mese/anno in cui il Test non sarà più sostenibile dagli studenti.
- Per un **Quiz a risposta multipla**, invece di implementare le risposte come semplici attributi *(ciò avrebbe limitato il docente a dover creare un Quiz con un numero prestabilito di risposte)*, nel modello ristrutturato si è scelto di creare una vera e propria Classe Risposta con indicazione della Lettera e del valore della risposta.
- Il sistema manterrà le più complete informazioni per quanto riguarda **Docente** e **Studente**, come ad esempio: il suo Ruolo (Docente di Informatica... ecc.), il suo Numero di Telefono e un suo URL che indirizza alla sua pagina docente. Per lo studente invece: oltre naturalmente alle informazioni di base, il suo Corso di Laurea.

- Per i **Docenti** si è scelto di fare un Mapping degli Insegnamenti e per **Studenti e Docenti** un Mapping dei Dipartimenti.
- Per un **Dipartimento**, il sistema manterrà le informazioni di: Codice Struttura, Nome, Direttore, Città, Provincia, Via, CAP.
- Per un **Insegnamento**, il sistema manterrà le informazioni di: Denominazione e Codice.
- Prevedendo la possibilità per un **Docente**, di poter non solo creare, ma anche modificare ed eliminare un Test, si è scelto di aggiungere una *Association Class* denominata Gestione, che permetterà di capire se il Docente avrà effettuato una Operazione di: *CREAZIONE*, *MODIFICA*, *ELIMINAZIONE* di un Test, nonchè la Data in cui l'operazione sarà stata effettuata, e un Orario.

Si noti che alcune scelte sono state rese effettive solo nel Class Diagram Ristrutturato.

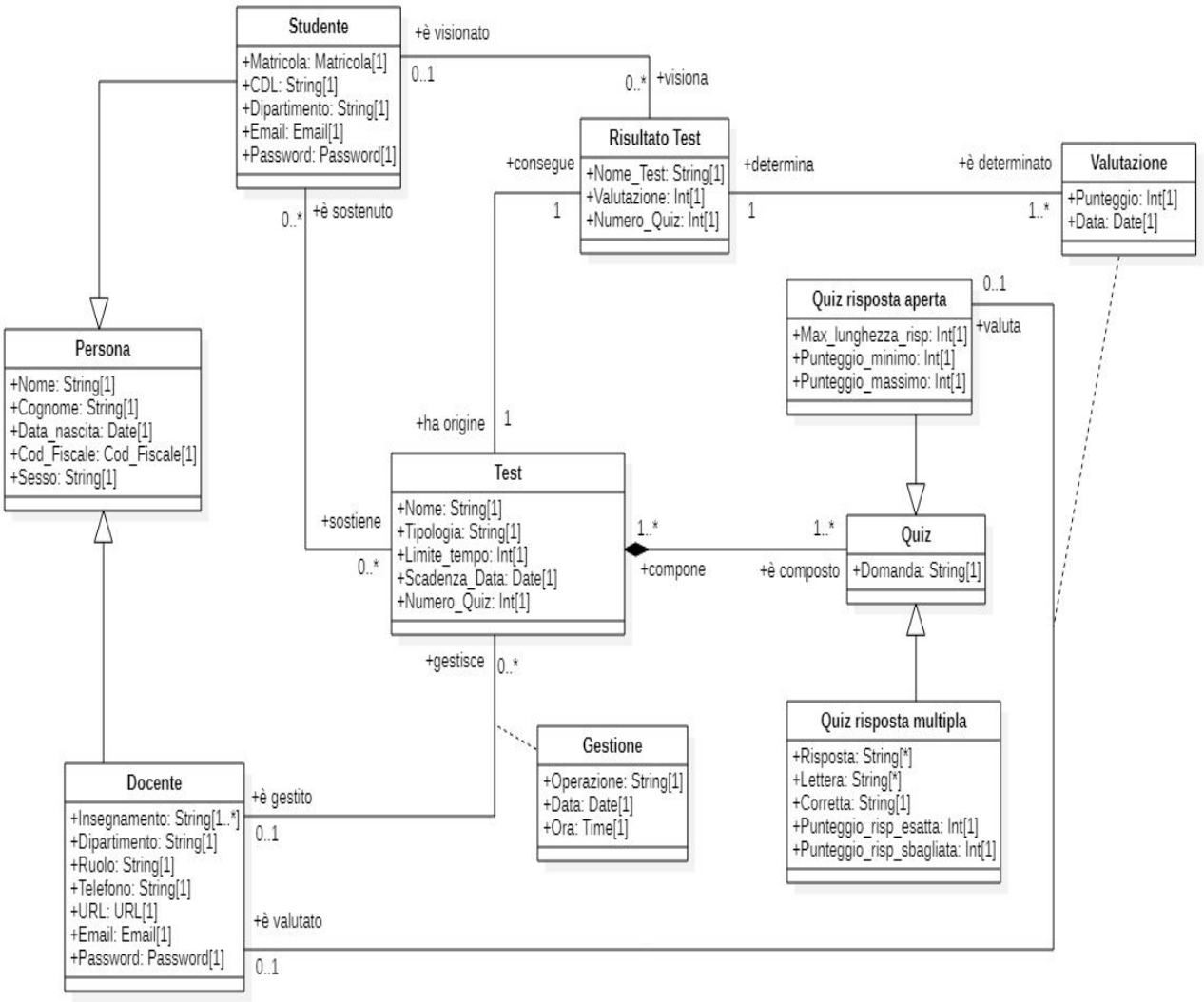
Capitolo 2

Progettazione concettuale

2.1 Introduzione

In questo capitolo mostreremo e analizzeremo nel dettaglio tutti gli aspetti della modellazione concettuale. Inizieremo con un **Class Diagram non ristrutturato**, per passare poi all'analisi dei processi che ci porteranno ad ottenere il **Class Diagram ristrutturato**. Infine, una **descrizione completa di classi, associazioni e vincoli**. Tutti i modelli sono creati tramite l'applicativo *StarUML*.

2.2 Class Diagram



2.3 Analisi della ristrutturazione del Class Diagram

In questa fase, con lo scopo di rendere il Class Diagram idoneo per la traduzione *in schemi relazionali* e di migliorare l'efficienza dell'implementazione, si procede alla **ristrutturazione** dello stesso. Al termine di questo operazione il Class Diagram non conterrà alcun attributo multiplo, composto, eventuali specializzazioni o generalizzazioni e si procederà all'inserimento di Identificativi validi.

2.3.1 Analisi delle ridondanze

In questo Class Diagram non sono presenti significative ridondanze tali da essere eliminate.

2.3.2 Analisi degli identificativi

Nell'analisi degli identificativi provvederemo a scegliere uno o più attributi che garantiranno il rispetto della proprietà di **integrità referenziale** per ogni *t-upla*.
Nel dettaglio notiamo che:

- Per uno **Studente** è già presente un attributo Matricola che rappresenta una possibile chiave primaria.
- Per un **Docente**, non ritenendo sufficiente basarci esclusivamente sul suo Codice Fiscale, abbiamo ritenuto opportuno aggiungere un attributo ID Docente e renderlo chiave primaria.
- Per un **Test**, al fine di permettere ai docenti di scegliere senza limitazioni il Nome, si è preferito aggiungere un attributo ID Test e renderlo chiave primaria.
- Per un **Insegnamento**, è già presente un codice identificativo ID Insegnamento.
- Per **Gestione** è stato aggiunto un attributo ID Operazione che identifica univocamente l'operazione effettuata.
- Per il **Dipartimento** si è scelto di identificarlo tramite un attributo già presente: Codice Struttura.
- Per i **Quiz a risposta multipla** si è scelto di inserire un attributo ID QuizM e renderlo chiave primaria.
- Per i **Quiz a risposta aperta** si è scelto di inserire un attributo ID QuizA e renderlo chiave primaria.
- Per l'entità **Risposta** si è scelto di inserire un attributo ID Risposta e renderlo chiave primaria.

- Per **Valutazione Risposta Multipla** si è scelto di inserire un attributo ID ValutazioneM e renderlo chiave primaria.
- Per **Valutazione Risposta Aperta** si è scelto di inserire un attributo ID ValutazioneA e renderlo chiave primaria.
- Per **Risultato Test** si è scelto di inserire un attributo ID RTest e renderlo chiave primaria.

2.3.3 Rimozione degli attributi multipli

In questo Class Diagram sono presenti tre attributi multipli:

- Il primo nell'entità **Docente**; l'attributo *Insegnamento* è stato modellato come vera e propria Entità, con un suo Codice Identificativo e una sua Denominazione.
- I restanti due nell'entità **Quiz risposta multipla**; l'attributo *Risposta* e l'attributo *Lettera*, il primo modellato come Entità e il secondo come attributo dell'entità **Risposta**, in modo da ottenere il risultato seguente: **Risposta**(ID Risposta, Lettera, Risposta).

2.3.4 Rimozione degli attributi composti

In questo Class Diagram non sono presenti attributi composti.

2.3.5 Partizione/Accorpamento delle associazioni

In questo Class Diagram è presente una associazione 1...1 eventualmente accorpabile, tra **Test** e **Risultato Test**.

Tuttavia:

1. Ritenendo in primo luogo fondamentale distinguere il Risultato di un Test (quindi con una valutazione finale assegnata dal docente e senza attributi come il limite di tempo e la data di scadenza) e un Test (quindi come entità che possiede una tipologia, un limite di tempo, una scadenza ecc.)...
2. Nonchè, in secondo luogo, distinguere in modo più evidente le interazioni che vi sono tra **Studente** e **Test** (uno studente sostiene un Test, non sostiene un Risultato di un Test) e tra **Studente** e **Risultato Test** (uno studente visiona il risultato di un Test)...

Si è presa la decisione di non eliminare l'associazione.

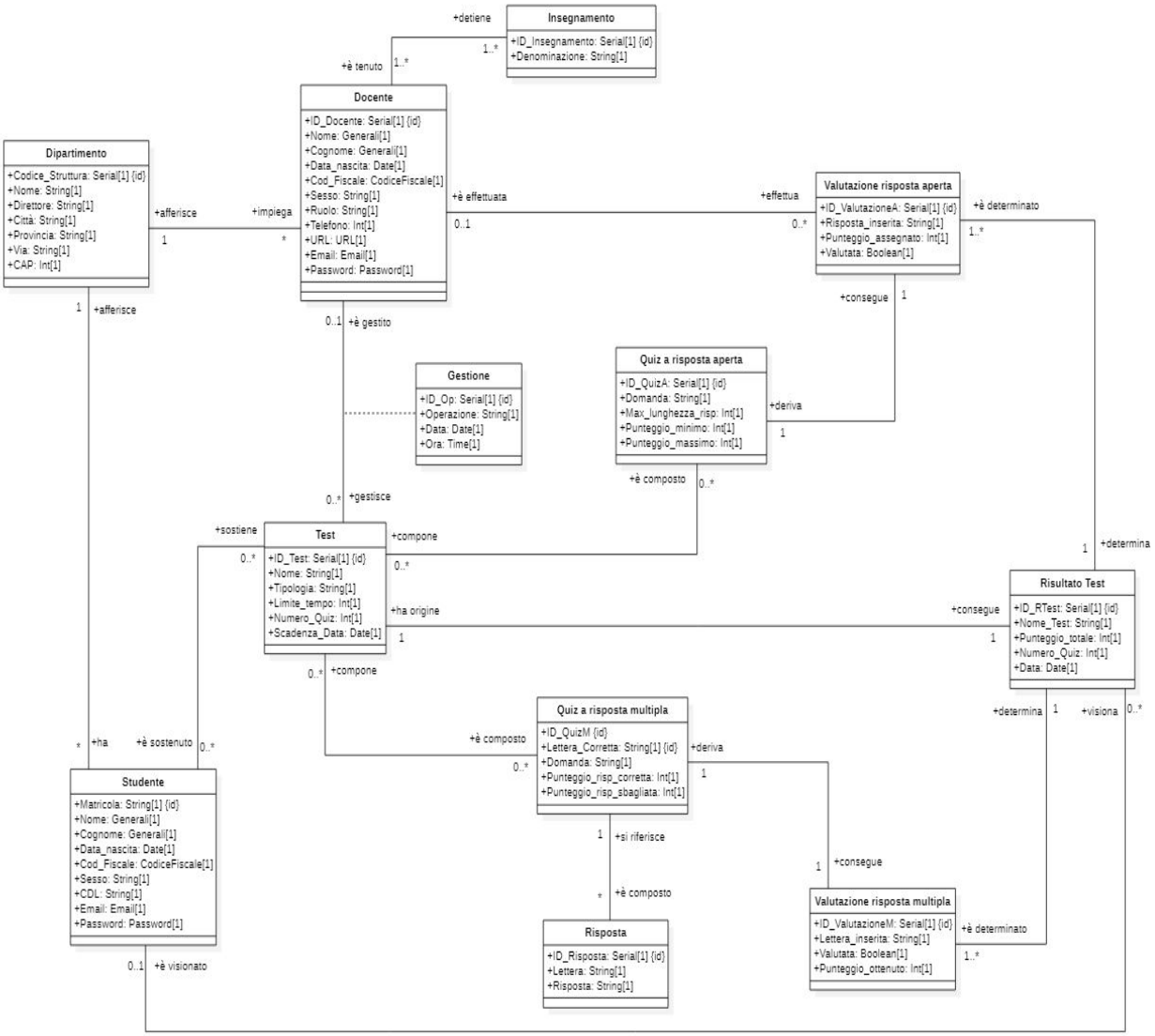
2.3.6 Rimozione delle gerarchie, delle aggregazioni e delle composizioni

In questo Class Diagram sono presenti due generalizzazioni e una composizione.

In particolare:

- Per la generalizzazione **Persona**: si è scelto di accorpare le entità figlie nell'entità padre ed ottenere come risultato finale una Classe **Docente** con tutti gli attributi di **Persona** e in più gli attributi della vecchia specializzazione **Docente**, e la Classe **Studente** con tutti gli attributi di **Persona** e in più gli attributi della vecchia specializzazione **Studente**.
- Per la generalizzazione **Quiz**: si è scelto di accorpare l'entità padre nelle entità figlie ed ottenere come risultato finale una Classe **Quiz a risposta aperta** con tutti gli attributi della precedente classe figlia **Quiz a risposta aperta** più l'attributo presente nella classe padre, e una Classe **Quiz a risposta multipla** con tutti gli attributi della precedente classe figlia **Quiz a risposta multipla** più l'attributo presente nella classe padre.
- La composizione **Test** e **Quiz** è stata sostituita con una semplice associazione.

2.4 Class Diagram ristrutturato



2.5 Dizionario delle classi

Classe	Descrizione	Attributi
Dipartimento	Luogo in cui si svolgono le attività didattiche	**Codice_Struttura** *(Serial)*: codice identificativo della struttura. **Nome** *(String)*: nome del dipartimento. **Direttore** *(String)*: direttore del dipartimento. **Città** *(String)*: città in cui si trova il dipartimento. **Provincia** *(String)*: provincia in cui si trova il dipartimento. **Via** *(String)*: Via in cui si trova il dipartimento. **CAP** *(String)*: CAP del dipartimento.
Studente	Persona iscritta all'università	**Matricola** *(String)*: codice identificativo di ogni studente. **Nome** *(Generali)*: nome dello studente. **Cognome** *(Generali)*: cognome dello studente. **Data_nascita** *(Date)*: data di nascita dello studente. **Cod_Fiscale** *(CodiceFiscale)*: codice fiscale dello studente. **Sesso** *(String)*: il genere dello studente. **CDL** *(String)*: corso di laurea dello studente. **Email** *(Email)*: email istituzionale dello studente. **Password** *(Password)*: codice alfanumerico.
Insegnamento	Materia insegnata dal docente	**ID_Insegnamento** *(Serial)*: codice identificativo dell'insegnamento. **Denominazione** (String): nome dell'insegnamento.

Classe	Descrizione	Attributi
Test	Prova in cui il candidato è invitato a rispondere in modo corretto a delle domande	**Limite_Tempo** *(Int)*: tempo a disposizione per lo svolgimento del Test. **ID_Test** *(Serial)*: codice identificativo di un Test. **Numero_Quiz** *(Int)*: numero totale di quiz per ogni Test. **Scadenza_Data** *(Date)*: Data ultima per sostenere il Test. **Nome** *(String)*: cognome dello studente. **Tipologia** *(String)*: tipologia di Test.
Docente	Persona che lavora all'Università	**ID_Docente** *(Serial)*: codice identificativo di ogni docente. **Nome** *(Generali)*: nome del docente. **Cognome** *(Generali)*: cognome del docente. **Data_nascita** *(Date)*: data di nascita del docente. **Cod_Fiscale** *(CodiceFiscale)*: codice fiscale del docente. **Sesso** *(String)*: genere del docente. **Ruolo** *(String)*: ruolo del docente all'interno dell'Università. **Telefono** *(Int)*: numero di telefono del docente. **URL** *(URL)*: Link rapido alla pagina del docente. **Email** *(Email)*: email istituzionale del docente. **Password** (Password): codice alfanumerico.

Classe	Descrizione	Attributi
Gestione	Descrittore delle operazioni tra Docente e Test.	**ID_Op** *(Serial)*: codice identificativo dell'operazione. **Operazione** *(String)*: nome dell'operazione (Creazione-Modifica-Eliminazione) **DataOperazione** *(TimeStamp)*: data e ora dell'operazione effettuata.
Quiz a risposta multipla	Tipologia di Quiz.	**ID_QuizM** *(Serial)*: codice identificativo del Quiz. **Lettera_Corretta** *(String)*: lettera corretta. **Domanda** *(String)*: domanda posta. **Punteggio_risp_corretta** *(Int)*: punteggio in caso di risposta corretta. **Punteggio_risp_sbagliata** *(Int)*: punteggio in caso di risposta non corretta.
Quiz a risposta aperta	Tipologia di Quiz.	**ID_QuizA** *(Serial)*: codice identificativo del Quiz. **Max_lunghezza_risp** *(Int)*: massimo numero di caratteri di una risposta. **Domanda** *(String)*: domanda posta. **Punteggio_minimo** *(Int)*: minimo punteggio assegnabile. **Punteggio_massimo** *(Int)*: massimo punteggio assegnabile.
Risposta	Elenco di risposte ad un Quiz a risposta multipla	**ID_Risposta** *(Serial)*: codice identificativo della Risposta. **Lettera** *(String)*: indice della risposta. **Risposta** *(String)*: valore della risposta.

Classe	Descrizione	Attributi
Risultato Test	Descrittore dei risultati ottenuti dopo aver sostenuto un Test	**ID_RTest** *(Serial)*: codice identificativo di un Test. **Nome_Test** *(String)*: nome del Test. **Punteggio_totale** *(Int)*: punteggio finale ottenuto al Test. **Numero_Quiz** *(Int)*: numero di quiz sostenuti nel Test. **Data** *(Date)*: data in cui è stato sostenuto il Test.
Valutazione risposta multipla	Descrittore della valutazione assegnata ad un Quiz a risposta multipla	**ID_ValutazioneM** *(Serial)*: codice identificativo della valutazione. **Lettera_Inserita** *(String)*: lettera inserita dallo studente. **Valutata** *(Boolean)*: valore di check per determinare se il Quiz è stato valutato dal sistema. **Punteggio_ottenuto** *(Int)*: punteggio assegnato dal sistema.
Valutazione risposta aperta	Tipologia di Quiz.	**ID_ValutazioneA** *(Serial)*: codice identificativo della valutazione. **Risposta_inserita** *(String)*: valore della risposta inserita. **Punteggio_assegnato** *(Int)*: punteggio assegnato dal docente. **Valutata** *(Boolean)*: valore di check per determinare se il Quiz è stato valutato dal docente.

2.6 Dizionario delle Associazioni

Nome	Descrizione	Classi coinvolte
afferisce/impiega	Un docente afferisce ad un dipartimento / Un dipartimento impiega un docente.	Tra Dipartimento e Docente [1...*]
è sostenuto/sostiene	Un test è sostenuto da uno studente / Uno studente sostiene un test.	Tra Studente e Test [*...*]
ha/afferisce	Un dipartimento ha degli studenti / Uno studente afferisce ad un dipartimento.	Tra Studente e Dipartimento [1...*]
è gestito/gestisce	Un test è gestito da un docente / Un docente gestisce un test.	Tra Test e Docente [1...*]
è tenuto/detiene	Un insegnamento è tenuto da un docente / Un docente detiene un insegnamento.	Tra Docente e Insegnamento [*...*]
è effettuata/effettua	Un docente effettua una valutazione ad un quiz a risposta aperta / La valutazione ad un quiz a risposta aperta è effettuata da un docente.	Tra Docente e Valutazione risposta aperta [1...*]
è composto/-compone	Un test è composto da quiz a risposta aperta / Un quiz a risposta aperta compone un test.	Tra Test e Quiz a risposta aperta [*...*]
ha origine/consegue	Un risultato test ha origine da un test / Ad un test consegue un risultato test.	Tra Test e Risultato Test [1...1]
è composto/-compone	Un test è composto da quia a risposta multipla / Un quiz a risposta multipla compone un test.	Tra Test e Quiz a risposta multipla. [*...*]

Nome	Descrizione	Classi coinvolte
è composto/si riferisce	Un quiz a risposta multipla è composto da risposte / Una risposta si riferisce ad un quiz a risposta multipla.	Tra Quiz a risposta multipla e Risposta [1...*]
è visionato/visiona	Un risultato test è visionato da uno studente / Uno studente visiona un risultato test.	Tra Studente e Risultato Test [1...*]
è determinato/determina	Un risultato test è determinato da una valutazione a risposta multipla / Una valutazione a risposta multipla determina un risultato test.	Tra Valutazione risposta multipla e Risultato Test [1...*]
è determinato/determina	Un risultato test è determinato da una valutazione a risposta aperta / Una valutazione a risposta aperta determina un risultato test.	Tra Valutazione risposta aperta e Risultato Test [1...*]
deriva/consegue	Una valutazione a risposta multipla deriva da un quiz a risposta multipla / Ad un quiz a risposta multipla consegue una valutazione a risposta multipla.	Tra Quiz a risposta multipla e Valutazione risposta multipla [1...1]
deriva/consegue	Una valutazione a risposta aperta deriva da un quiz a risposta aperta / Ad un quiz a risposta aperta consegue una valutazione a risposta aperta.	Tra Quiz a risposta aperta e Valutazione risposta aperta [1...1]

2.7 Dizionario dei Vincoli

Nome	Descrizione
EmailCheckUnina	Gli indirizzi email istituzionali di Studenti e Docenti devono rispettare i seguenti parametri: "%@studenti.unina.it" oppure "%@unina.it".
PasswordDomainCheck	Le password di Studenti e Docenti devono rispettare gli attuali standard di sicurezza: lunghezza minima 8 caratteri, tra cui numeri, lettere maiuscole, minuscole e caratteri speciali.
URLDocenteUnina	L'URL di un Docente non può essere differente da: "https://www.docenti.unina.it/%".
EmailUniqueTableDocente	Non possono esistere Docenti con la stessa Email.
EmailUniqueTableStudente	Non possono esistere Studenti con la stessa Email.
CodFiscaleCheck	Il codice fiscale può contenere solo lettere maiuscole e numeri, oltre a dover rispettare la lunghezza di 16 caratteri.
GeneraliCorrette	Il nome e il cognome di Studenti e Docenti non devono contenere numeri, caratteri speciali e non devono essere NULL.

Capitolo 3

Schema logico

3.1 Introduzione

In questo capitolo andremo ad analizzare dettagliatamente i meccanismi che ci permetteranno di passare da uno schema concettuale (già precedentemente predisposto a ristrutturazione) ad uno schema logico,scendendo ad un livello di astrazione ancora più profondo.

3.2 Schema

Dipartimento: CodiceStruttura - Nome - Direttore - Città - Provincia - Via - CAP

Studente: Matricola - Nome Cognome - DatadiNascita - CodFiscale - Sesso - CDL - Email - Password - Dipartimento

- Dipartimento $\hookrightarrow$ Dipartimento(CodiceStruttura)

Docente: IdDocente - Nome - Cognome - DataDiNascita - CodFiscale - Sesso - Telefono - URL - Email - Password - Dipartimento

- Dipartimento $\hookrightarrow$ Dipartimento(CodiceStruttura)

Insegnamento: IdInsegnamento - Denominazione - IdDocente

- IdDocente $\hookrightarrow$ Docente(IdDocente)

Test: IdTest - Nome - LimiteTempo - NumeroQuiz - ScadenzaData - ProprietarioTest

- ProprietarioTest $\hookrightarrow$ Docente(IdDocente)

Gestione: IdOperazione - Operazione - DataOperazione - IdDocente - IdTest

- IdDocente $\hookrightarrow$ Docente(IdDocente)
- IdTest $\hookrightarrow$ Test(IdTest)

QuizAperta: IdQuizA - Domanda - Risposta - MaxLunghezzaRisposta - PunteggioMinimo - PunteggioMassimo - IdTestRiferimento

- IdTestRiferimento $\hookrightarrow$ Test(IdTest)

QuizMultipla: IdQuizM - LetteraCorretta - Domanda - PunteggioRispostaCorretta - PunteggioRispostaSbagliata - IdTestRiferimento

- IdTestRiferimento $\hookrightarrow$ Test(IdTest)

Risposta: IdRisposta - Risposta - LetteraToken - IdQuizRiferimento

- IdQuizRiferimento $\hookrightarrow$ QuizMultipla(IdQuizM)

RisultatoTest: IdRisultatoTest - Matricola - PunteggioTotale - DataTest - IdTest - NumeroQuiz

- IdTest $\hookrightarrow$ Test(IdTest)
- Matricola $\hookrightarrow$ Studente(Matricola)

ValutazioneRispostaAperta: IdValutazioneAperta - RispostaInserita - PunteggioAssegnato - Valutata - Matricola - IdDocente - IdRisultatoTest - IdQuizA

- IdRisultatoTest $\hookrightarrow$ RisultatoTest(IdRisultatoTest)
- Matricola $\hookrightarrow$ Studente(Matricola)
- IdDocente $\hookrightarrow$ Docente(IdDocente)
- IdQuizA $\hookrightarrow$ QuizAperta(IdQuizA)

ValutazioneRispostaMultipla: IdValutazioneMultipla - LetteraInserita - PunteggioOttenuto - Valutata - Matricola - IdRisultatoTest - IdQuizM

- IdRisultatoTest $\hookrightarrow$ RisultatoTest(IdRisultatoTest)
- Matricola $\hookrightarrow$ Studente(Matricola)
- IdQuizM $\hookrightarrow$ QuizMultipla(IdQuizM)

Capitolo 4

Progettazione Fisica

In questo capitolo, ultimo del progetto, analizzeremo i meccanismi di traduzione da uno schema logico ad uno schema fisico. Saranno effettuate le definizioni delle *tabelle* (con i vari attributi e i loro corrispettivi tipi), le definizioni di *funzioni*, *procedure* e altre *automazioni*, i *triggers*, i *vincoli*, le *sequenze* e i *domini*.

4.1 Definizione delle tabelle

Seguono le definizioni delle tabelle estratte dal documento .sql di creazione del DataBase.

4.1.1 Definizione della Tabella *Dipartimento*

```
1  /*
2      ----------------------------
3          !Table-DIPARTIMENTO!
4      ----------------------------
5  */
6
7  CREATE TABLE DIPARTIMENTO
8  (
9              CodiceStruttura VARCHAR(5),
10             Nome VARCHAR(120),
11             Direttore VARCHAR(40),
12             Citta VARCHAR(40),
13             provincia VARCHAR(40),
14             Via VARCHAR(40),
15             Cap INT,
16             PRIMARY KEY (CodiceStruttura)
17 );
```

4.1.2 Definizione della Tabella *Studente*

```
/*
    ------------------------------
            !Table-STUDENTE!
    ------------------------------
*/

CREATE TABLE STUDENTE
(
            Matricola VARCHAR(9),
            Nome GENERALI,
            Cognome GENERALI,
            DatadiNascita DATE,
            CodFiscale CODICEFISCALE,
            Sesso VARCHAR(1),
            CDL VARCHAR(50),
            Email EMAIL_DOMINIO,
            Password PASSWORD_DOMINIO,
            Dipartimento VARCHAR(5),

            PRIMARY KEY(Matricola),
            FOREIGN KEY(Dipartimento) REFERENCES  DIPARTIMENTO(
    CodiceStruttura)
);
```

4.1.3 Definizione della Tabella *Docente*

```
/*
    ------------------------------
            !Table-DOCENTE!
    ------------------------------
*/
CREATE TABLE DOCENTE
(
            IdDocente VARCHAR(5),
            Nome GENERALI,
            Cognome GENERALI,
            DataDiNascita DATE,
            CodFiscale CODICEFISCALE,
            Sesso VARCHAR(1),
            Telefono INT,
            URL URL,
            Email EMAIL_DOMINIO,
            Password PASSWORD_DOMINIO,
            Dipartimento VARCHAR(5),

            PRIMARY KEY(IdDocente),
            FOREIGN KEY(Dipartimento) REFERENCES  DIPARTIMENTO(
    CodiceStruttura)
);
```

4.1.4 Definizione della Tabella *Insegnamento*

```
/*
    ------------------------------
        !Table-INSEGNAMENTO!
    ------------------------------
*/
CREATE TABLE INSEGNAMENTO
(
            IdInsegnamento VARCHAR(5),
            denominazione VARCHAR(80),
            IdDocente VARCHAR(5),

            PRIMARY KEY(IdInsegnamento),
            FOREIGN KEY(IdDocente) REFERENCES  DOCENTE(IdDocente)
);
```

4.1.5 Definizione della Tabella *Test*

```
/*
    ------------------------------
            !Table-TEST!
    ------------------------------
*/
CREATE TABLE TEST
(
            IdTest SERIAL,
            Nome VARCHAR(80),
            LimiteTempo INT,
            NumeroQuiz INT,
            ScadenzaData DATE,
            PropietarioTest VARCHAR(5),

            PRIMARY KEY(IdTest),
            FOREIGN KEY(PropietarioTest) REFERENCES  DOCENTE(IdDocente
    )
);
```

4.1.6 Definizione della Tabella *Gestione*

```
/*
    ------------------------------
          !Table-GESTIONE!
    ------------------------------
*/
CREATE TABLE GESTIONE
(
          IdOperazione SERIAL,
          Operazione VARCHAR(30),
          DataOperazione TIMESTAMP,
          IdDocente VARCHAR(5),
          IDTest INT,

          PRIMARY KEY(IdOperazione),
          FOREIGN KEY(IdDocente) REFERENCES DOCENTE(IdDocente),
          FOREIGN KEY(IdTest) REFERENCES TEST(IdTest)
);
```

4.1.7 Definizione della Tabella *QuizAperta*

```
/*
    ------------------------------
        !Table-QUIZAPERTA!
    ------------------------------
*/
CREATE TABLE QUIZAPERTA
(
          IdQuizA SERIAL,
          Domanda VARCHAR(800),
          Risposta VARCHAR(2000),
          MaXLunghezzaRisposta INT,
          PunteggioMinimo INT,
          PunteggioMassimo INT,
          IdtestRiferimento INT,

          PRIMARY KEY(IdQuizA),
          FOREIGN KEY(IdtestRiferimento) REFERENCES TEST(IdTest)
          ON DELETE CASCADE
);
```

4.1.8 Definizione della Tabella *QuizMultipla*

```
/*
    ------------------------------
        !Table-QUIZMULTIPLA!
    ------------------------------
*/
CREATE TABLE QUIZMULTIPLA
(
            IdQuizM SERIAL,
            LetteraCorretta VARCHAR(1),
            Domanda VARCHAR(800),
            PunteggioRispostaCorretta INT,
            PunteggioRispostaSbagliato INT,
            IdtestRiferimento INT,

            PRIMARY KEY(IdQuizM),
            FOREIGN KEY(IdtestRiferimento) REFERENCES TEST(IdTest)
            ON DELETE CASCADE
);
```

4.1.9 Definizione della Tabella *Risposta*

```
/*
    ------------------------------
        !Table-RISPOSTA!
    ------------------------------
*/
CREATE TABLE RISPOSTA
(
            IDRisposta SERIAL,
            Risposta VARCHAR(300),
            LetteraToken VARCHAR(1),
            IDQuizRiferimento INT,

            PRIMARY KEY(IDRisposta) ,
            FOREIGN KEY(IDQuizRiferimento) REFERENCES QUIZMULTIPLA(
    IdQuizM)
            ON DELETE CASCADE
);
```

4.1.10 Definizione della Tabella *RisultatoTest*

```
/*
    ----------------------------------------
                    !Table-RISULTATOTEST!
    ----------------------------------------
*/
CREATE TABLE RISULTATOTEST
(
            IdRisultatoTest SERIAL,
            Matricola VARCHAR(9),
            PunteggioTotale INT,
            DataTest DATE,
            IdTest INT,
            NumeroQuiz INT,

            PRIMARY KEY (IdRisultatoTest),
            FOREIGN KEY(IdTest) REFERENCES TEST(IdTest),
            FOREIGN KEY(Matricola) REFERENCES STUDENTE(Matricola)
);
```

4.1.11 Definizione della Tabella *ValutazioneRispostaAperta*

```
/*
    ---------------------------------------
    !Table-VALUTAZIONERISPOSTAAPERTA!
    ---------------------------------------
*/
CREATE TABLE VALUTAZIONERISPOSTAAPERTA
(
            IdValutazioneAperta SERIAL,
            RispostaInserita VARCHAR(2000),
            PunteggioAssegnato INT,
            VALUTATA BOOLEAN,
            Matricola VARCHAR(9),
            IdDocente VARCHAR(5),
            IdRisultatoTest INT,
            IdQuizA INT,

            PRIMARY KEY (IdValutazioneAperta),
            FOREIGN KEY(IdRisultatoTest) REFERENCES RISULTATOTEST(
    IdRisultatoTest)
            ON DELETE CASCADE,
            FOREIGN KEY(IdDocente) REFERENCES DOCENTE(IdDocente),
            FOREIGN KEY(Matricola) REFERENCES STUDENTE(Matricola),
            FOREIGN KEY(IdQuizA) REFERENCES QUIZAPERTA(IdQuizA)

);
```

4.1.12 Definizione della Tabella *ValutazioneRispostaMultipla*

```
/*
    ----------------------------------------
    !Table-VALUTAZIONERISPOSTAMULTIPLA!
    ----------------------------------------
*/
CREATE TABLE VALUTAZIONERISPOSTAMULTIPLA
(
            IdValutazioneMultipla SERIAL,
            LetteraInserita VARCHAR(1),
            Matricola VARCHAR(9),
            PunteggioOttenuto INT,
            VALUTATA BOOLEAN,
            IdRisultatoTest INT,
            IdQuizM INT,

            PRIMARY KEY (IdValutazioneMultipla),
            FOREIGN KEY(IdRisultatoTest) REFERENCES RISULTATOTEST(
    IdRisultatoTest)
            ON DELETE CASCADE,
            FOREIGN KEY(Matricola) REFERENCES STUDENTE(Matricola),
            FOREIGN KEY(IdQuizM) REFERENCES QUIZMULTIPLA(IdQuizM)
);
```

4.2 Sequence

4.2.1 *Test*

```
CREATE SEQUENCE TestId
START 1
INCREMENT 1
MINVALUE 1
MAXVALUE 99999
OWNED BY TEST.IdTest;
```

4.2.2 *Gestione*

```

CREATE SEQUENCE GestioneId
START 1
INCREMENT 1
MINVALUE 1
MAXVALUE 99999
OWNED BY GESTIONE.IdOperazione;
```

4.2.3 *QuizAperta*

```

CREATE SEQUENCE IdQuizSeq
START 1
INCREMENT 1
MINVALUE 1
MAXVALUE 300000
OWNED BY QUIZAPERTA.IdQuizA;
```

4.2.4 *QuizMultipla*

```
CREATE SEQUENCE IdQuizMSeq
START 1
INCREMENT 1
MINVALUE 1
MAXVALUE 300000
OWNED BY QUIZMULTIPLA.IdQuizM;
```

4.2.5 *Risposta*

```
CREATE SEQUENCE IdRiSpostaSeq
START 1
INCREMENT 1
MINVALUE 1
MAXVALUE 300000
OWNED BY RISPOSTA.IDRisposta;
```

4.2.6 *ValutazioneRispostaAperta*

```
1 CREATE SEQUENCE IdValutazioneApertaSeq
2 START 1
3 INCREMENT 1
4 MINVALUE 1
5 MAXVALUE 300000
6 OWNED BY VALUTAZIONERISPOSTAAPERTA.IdValutazioneAperta;
```

4.2.7 *ValutazioneRispostaMultipla*

```
1 CREATE SEQUENCE IdValutazioneMultiplaSeq
2 START 1
3 INCREMENT 1
4 MINVALUE 1
5 MAXVALUE 300000
6 OWNED BY VALUTAZIONERISPOSTAMULTIPLA.IdValutazioneMultipla;
```

4.3 Funzioni, Procedure ed altre Automazioni

4.3.1 Funzione di Correzione Automatica delle Domande a Risposta Multipla

```
1  --Fase 1
2
3  --Creo come prima cosa la funzione che aggiorna il punteggio del
       risultato del test
4  CREATE OR REPLACE FUNCTION CorreggiMultipla(ValidDaFunction
       valutazionerispostamultipla.idvalutazionemultipla%TYPE)
5  RETURNS void
6
7  AS
8  $CorreggiMultipla$
9  DECLARE
10 letteraCorrettaDaquery quizmultipla.letteracorretta%TYPE;
11 valutazioneMultipla valutazionerispostamultipla%ROWTYPE;
12 quizMultiplo quizmultipla%ROWTYPE;
13 idquizmDaquery quizmultipla.idquizm%TYPE;
14 punteggioOttenutoNuovo int;
15
16 BEGIN
17           SELECT * FROM valutazionerispostamultipla
18           INTO valutazioneMultipla WHERE idvalutazionemultipla=
       ValidDaFunction;
19
20           idquizmDaquery:=valutazioneMultipla.idquizm;
21
22
23           SELECT * FROM quizmultipla INTO quizMultiplo WHERE idquizm
       =idquizmDaquery;
24
25           letteraCorrettaDaquery:=quizMultiplo.letteracorretta;
26           IF letteraCorrettaDaquery = valutazioneMultipla.
       letterainserita THEN
27                   punteggioOttenutoNuovo:=quizmultiplo.
       punteggiorispostacorretta;
28
29           ELSE
30           punteggioOttenutoNuovo:=quizmultiplo.
       punteggiorispostasbagliato;
31           END IF;
32           UPDATE valutazionerispostamultipla
33           set punteggioottenuto=punteggioOttenutoNuovo
34           WHERE idvalutazionemultipla=ValidDaFunction;
35
36           UPDATE risultatotest
37           SET punteggiototale=punteggiototale+punteggioOttenutoNuovo
38           WHERE idrisultatotest=valutazioneMultipla.idrisultatotest;
39
40 END;
41 $CorreggiMultipla$ LANGUAGE plpgsql;
```

```
42
43 --Fase 2
44
45 --Creo il trigger addetto al controllo dell effettivo lancio della
    funzione
46
47 CREATE OR REPLACE FUNCTION TriggerCorreggiMultipla()
48   RETURNS TRIGGER
49   AS
50 $TriggerCorreggiMultipla$
51 BEGIN
52             IF NEW.valutata=false THEN
53             PERFORM correggimultipla(NEW.idvalutazionemultipla);
54             UPDATE valutazionerispostamultipla
55             SET valutata = true
56             WHERE idvalutazionemultipla=NEW.idvalutazionemultipla;
57             END IF;
58             RETURN NEW;
59 END;
60 $TriggerCorreggiMultipla$ LANGUAGE plpgsql;
61
62 --Fase 3
63
64 --Creo il trigger vero e proprio lanciatore
65 CREATE TRIGGER correggiMultipleTriggerLanciatore
66 AFTER INSERT
67 ON valutazionerispostamultipla
68 FOR EACH ROW
69 EXECUTE PROCEDURE TriggerCorreggiMultipla();
```

4.3.2 Funzione di Aggiornamento della Correzione delle Domande a Risposta Aperta

```
1
2
3
4 --Fase 1
5
6 --Creo la funzione che aggiorna il punteggio del risultato del test
7
8 CREATE OR REPLACE FUNCTION AggiornaPunteggioAperta(ValidDaFunction
    valutazionerispostaaperta.idvalutazioneaperta%TYPE)
9 RETURNS void
10
11 AS
12 $AggiornaPunteggioAperta$
13 DECLARE
14 valutazioneaperta valutazionerispostaaperta%ROWTYPE;
15 quizAperta quizaperta%ROWTYPE;
16 idquizaDaquery quizaperta.idquiza%TYPE;
17 punteggioOttenuto int;
18
```

```
19 BEGIN
20           SELECT * FROM valutazionerispostaaperta
21           INTO valutazioneaperta WHERE idvalutazioneaperta=
     ValidDaFunction;
22
23           idquizaDaquery:=valutazioneaperta.idquiza;
24
25
26           SELECT * FROM quizaperta INTO quizAperta WHERE idquiza=
     idquizaDaquery;
27
28
29           IF valutazioneaperta.PunteggioAssegnato<=quizAperta.
     PunteggioMassimo AND valutazioneaperta.PunteggioAssegnato>=
     quizAperta.PunteggioMinimo    THEN
30                   punteggioOttenuto:=valutazioneaperta.
     PunteggioAssegnato;
31
32           ELSE
33           raise notice 'Errore nell assegnazione del codice ';
34           END IF;
35           UPDATE risultatotest
36           SET punteggiototale=punteggiototale+punteggioOttenuto
37           WHERE idrisultatotest=valutazioneaperta.idrisultatotest;
38
39 END;
40 $AggiornaPunteggioAperta$ LANGUAGE plpgsql;
41
42 --Fase 2
43
44 --Creo il trigger addetto al controllo dell effettivo lancio della
     funzione
45
46 CREATE OR REPLACE FUNCTION TriggerCorreggiAperta()
47   RETURNS TRIGGER
48
49   AS
50 $TriggerCorreggiAperta$
51 BEGIN
52           IF NEW.valutata=true THEN
53           PERFORM AggiornaPunteggioAperta(NEW.idvalutazioneaperta);
54           END IF;
55           RETURN NEW;
56 END;
57 $TriggerCorreggiAperta$    LANGUAGE PLPGSQL;
58
59 --Fase 3
60
61 --Creo il trigger vero e proprio lanciatore
62
63 CREATE TRIGGER CorreggiApertaLanciatore
64 AFTER UPDATE
```

```
65 ON valutazionerispostaaperta
66 FOR EACH ROW
67 EXECUTE PROCEDURE TriggerCorreggiAperta();
```

4.3.3 Funzione di Aggiornamento Automatico dei Quiz

```
1
2 CREATE OR REPLACE FUNCTION aggiornaNumeroQuiz()
3   RETURNS TRIGGER
4 AS
5 $TriggerAggiungiQuiz$
6 BEGIN
7             UPDATE Test
8             SET numeroquiz=numeroquiz+1
9             where idtest=NEW.idtestriferimento;
10            RETURN NEW;
11 END;
12 $TriggerAggiungiQuiz$    LANGUAGE PLPGSQL;
13
14 --Trigger Aperta
15
16 create trigger AggiugniQuizApertaLanciatore
17 after INSERT
18 on quizaperta
19 FOR EACH ROW
20 EXECUTE PROCEDURE aggiornaNumeroQuiz();
21
22 --Trigger Multipla
23
24 create trigger AggiugniQuizApertaLanciatore
25 after INSERT
26 on quizmultipla
27 FOR EACH ROW
28 EXECUTE PROCEDURE aggiornaNumeroQuiz();
```

4.4 Implementazione dei Vincoli

```
1
2
3 /*
4                 ----------------------------------------
5                       !VINCOLI->TABLE->DOCENTE!
6                 ----------------------------------------
7 */
8
9 ALTER TABLE DOCENTE
10 ADD CONSTRAINT EmailUniqueTableDocente UNIQUE (email);
11
12 /*
13                ----------------------------------------
14                      !VINCOLI->TABLE->STUDENTE!
15                ----------------------------------------
16 */
17
18 ALTER TABLE STUDENTE
19 ADD CONSTRAINT EmailUniqueTableStudente UNIQUE (email);
```

4.5 Creazione Domini

```
1
2
3 /*
4                     ------------------------------
5                          !Creazione Domini!
6                     ------------------------------
7 */
8
9 --Vincolo Di Dominio : EmailCheckUnina
10 CREATE DOMAIN EMAIL_DOMINIO AS VARCHAR(60)
11             CHECK ( VALUE LIKE '%@studenti.unina.it' OR VALUE LIKE '%
    @unina.it' );
12
13 --Vincolo Di Dominio : PasswordDomainCheck
14 CREATE DOMAIN PASSWORD_DOMINIO AS VARCHAR(40)
15             CHECK (VALUE ~ '^.*(?=.*[@!#$^*%&])(?=.*[0-9])(?=.*[a-zA-Z
    ]).*$'
16                         AND VALUE LIKE '________%');
17
18 --Vincolo Di Dominio : UrlDocenteUnina
19 CREATE DOMAIN URL AS VARCHAR(60)
20             CHECK ( VALUE LIKE 'https://www.docenti.unina.it/%' );
21
22 --Vincolo Di Dominio : GeneraliCorrette
23 CREATE DOMAIN GENERALI AS VARCHAR(60)
24             CHECK ( VALUE <> '' AND VALUE NOT SIMILAR TO '%[0-9]+%'
    AND VALUE NOT SIMILAR TO '%[@!#$^*%&]+%'   );
```

```
25
26 --Vincolo Di Dominio : CodFiscaleCheck
27 CREATE DOMAIN CODICEFISCALE AS VARCHAR(16)
28             CHECK (VALUE <> '' AND VALUE ~ '^.*(?=.*[0-9])(?=.*[A-Z])
       .*$' AND VALUE NOT SIMILAR TO '%[a-z]+%' AND VALUE NOT SIMILAR TO
       '%[@!#$^*%&]+%');
29
```