Documentazione Progetto Basi Di Dati

Modellazione Concettuale (Modello ER)

In questa sezione discuterò le mie scelte per la progettazione concettuale, per la quale ho prodotto un modello ER che contiene generalizzazioni e ridondanze, durante la progettazione logica esse verranno eliminate tramite la ristrutturazione del modello.

Partiamo dalla rappresentazione degli utenti: nella nostra applicazione avremo i docenti, gli studenti e la segreteria, che formano tre diversi gruppi di utenti.

Abbiamo quindi un’entità utente padre di una generalizzazione che ha come figli le entità docenti, studenti e segreteria.

Immagine che contiene schermata, diagramma, linea, spazio

Descrizione generata automaticamente

Come si può notare dall’immagine, ho deciso di non creare relazioni tra l’entità segreteria e tutte le entità che essa andrà a gestire nell’applicazione web.

Ho deciso di non mostrare queste relazioni per mantenere semplice il modello ER, e soprattutto perché le operazioni di gestione sono rese disponibili alla segreteria grazie all’applicazione web, lato database non sono necessarie relazioni tra la tabella segreteria e le altre tabelle presenti.

La segreteria, i docenti e gli studenti hanno delle credenziali che useranno per accedere all’applicazione web.

Inoltre, per i docenti abbiamo anche il loro nome e cognome e il loro id, che funge da chiave primaria.  
Nel caso ci siano professori con lo stesso nome e cognome, l’id permetterà di identificarli e distinguerli.  
Per gli studenti abbiamo il loro nome e cognome, il loro anno di immatricolazione e la loro matricola che rappresenta la loro chiave primaria.   
La Matricola permette di identificare gli studenti anche nei casi (presumo rari) in cui ci siano studenti con lo stesso nome, cognome e anno di immatricolazione.

Immagine che contiene diagramma, schermata, testo

Descrizione generata automaticamente

Passiamo a parlare delle entità che rappresentano i docenti e gli studenti e delle altre entità in relazione ad esse.

Un docente può essere responsabile di al massimo tre insegnamenti, e per ogni insegnamento c’è un solo responsabile.

Per quanto riguarda le propedeuticità, ho deciso di rappresentarle con una relazione ricorsiva per l’entità insegnamento.   
Non tutti gli insegnamenti hanno propedeuticità o sono propedeutici per altri insegnamenti, ho quindi assegnato ad entrambi i versi della relazione una partecipazione facoltativa (cardinalità minima zero), un insegnamento può essere propedeutico per uno o più insegnamenti e un insegnamento può avere una o più propedeuticità, abbiamo quindi cardinalità massima N.

Per Insegnamento abbiamo un codice identificativo (che farà da chiave primaria), il suo nome, l’anno in cui è previsto e la descrizione.

Gli insegnamenti prevedono degli esami, rappresentati dall’entità Esame, ad ogni insegnamento possono corrispondere uno o più esami (per esempio per programmazione 1 ci possono essere più appelli disponibili) e ad ogni esame corrisponde un solo insegnamento.

Ho deciso di formare un’entità debole per Esame, perché, per sua natura, per identificare un esame c’è bisogno, oltre che alla data, anche dell’insegnamento relativo all’esame.

Nell’entità Esame abbiamo quindi data esame che forma chiave primaria assieme alla chiave di insegnamento, e il nome dell’esame (per esempio “Esame di Programmazione 1”).

Per la composizione dei corsi di laurea, ho deciso che un insegnamento, gestito sempre dallo stesso docente, può comparire in più corsi di laurea, permettendo scenari simili a quello che abbiamo nel nostro corso di laurea con l’insegnamento Programmazione 1: Programmazione 1 è gestito da un docente ed è presente nei corsi di Programmazione, Sicurezza e Comunicazione Digitale e gli esami relativi sono disponibili agli studenti di tutti i corsi di laurea che condividono l’insegnamento.

Per i casi dove ci sono insegnamenti simili tra loro ma con programmi didattici diversi, come analisi 1 e matematica del continuo, essi verranno considerati due insegnamenti indipendenti, indipendentemente dal loro responsabile.

I corsi di laurea si suddividono in Triennali e Magistrali, c’è quindi una generalizzazione con entità padre Corso Di Laurea e come figli Triennale e Magistrale.

I corsi di Laurea hanno un codice identificativo (che fa da chiave primaria) e dal loro nome.

Gli studenti frequentano un solo corso di laurea, e quindi abbiamo una relazione uno a molti tra studente e corso di laurea: ad uno studente corrisponde un solo corso, mentre ad un corso corrispondono più studenti.

Discuterò ora della carriera e della registrazione dei voti.

Immagine che contiene schermata, diagramma

Descrizione generata automaticamente

Per ogni studente è associata una carriera, ma la carriera è formata dai vari esami, e quindi ad uno studente possono corrispondere nessuno, uno, o più esami che uniti formano la carriera dello studente.

In carriera andranno tutti gli esiti degli esami sostenuti dagli studenti, per poter identificare e distinguere i record, c’è bisogno delle chiavi esterne di tutte le entità coinvolte nella registrazione del voto: la matricola di uno studente indica che l’esame è stato sostenuto da quello studente.   
L’id del docente responsabile e l’id dell’insegnamento identificano l’esame e lo distinguono dagli esami di altri insegnamenti gestiti da altri docenti, e infine la data dell’esame distingue sia gli esami di insegnamenti diversi sia gli esami corrispondenti allo stesso insegnamento, permettendo di inserire più appelli per gli stessi insegnamenti senza problemi.  
Per questo l’entità Carriera è debole e ha come chiave primaria le chiavi di tutte le entità coinvolte.

Per quanto riguarda le cardinalità, alla registrazione di un voto nella carriera devono corrispondere un docente e un insegnamento, mentre per gli esami la partecipazione minima non è obbligatoria: uno studente può non aver sostenuto un esame, nella carriera ci sono solo gli esami che lo studente ha sostenuto.

Parliamo ora degli storici per gli studenti e le carriere.

Immagine che contiene schermata, linea, diagramma, Diagramma

Descrizione generata automaticamente

Per prima cosa farò una premessa simile a quella fatta per la segreteria: Per entrambe le entità riguardante gli storici bisognerebbe disegnare delle relazioni che le collegano agli esami, ai corsi di laurea e alle restanti entità.  
Ma, anche in questo caso, gli storici vengono coinvolti nelle operazioni delle applicazioni solo quando gli studenti vengono rimossi, ed è un’operazione che avviene tramite l’applicazione web.

Come spiegherò nella parte di spiegazione delle funzioni svolte, l’archiviazione dei dati avviene tramite dei trigger prima della cancellazione di righe sia sullo studente che sulla carriera.  
Le operazioni svolte non richiedono legami (join) tra gli storici e le entità associate, non essendoci quindi il bisogno di relazioni “logiche” nelle operazioni, ho deciso di non collegare le entità con delle relazioni, mantenendo inoltre il diagramma semplice da consultare.

In storico studente sono presenti tutti i dati di quello che era, prima dell’archiviazione, uno studente “attivo”, ho aggiunto id\_corso non avendo collegato l’entità a delle relazioni.  
Nello storico carriera sono presenti tutti i dati della carriera di uno studente archiviato, sono quindi presenti anche le chiavi esterne che componevano la chiave primaria della carriera originaria, perché al momento dell’archiviazione di uno studente viene effettuata una copia dei dati relativi allo studente negli storici.  
Ho assegnato queste chiavi esterne come attributi perché, come detto poc’anzi, ho deciso di non rappresentare le relazioni che collegherebbero le entità coinvolte.

Inoltre, la chiave dello storico della carriera è formata, come per la carriera attiva, dalla matricola, e dai vari id dei corsi, insegnamenti dei docenti e dalla data dell’esame.

Di seguito trovate il modello ER completo.

Immagine che contiene diagramma

Descrizione generata automaticamente

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

Progettazione Logica

In questa sezione parlerò della progettazione logica, che consiste nella ristrutturazione del modello ER per rimuovere ridondanze e generalizzazioni, e dalla transizione dal modello ER al modello logico (modello relazionale).

Ristrutturazione Modello Concettuale

In questa sezione parlerò delle modifiche apportate al modello ER mostrato nella sezione precedente, per renderlo conforme al modello relazionale e alle possibilità di PostgreSQL.

Eliminazione delle generalizzazioni

Immagine che contiene schermata, testo, linea

Descrizione generata automaticamente

Ho eliminato la generalizzazione di utente scomponendola nelle tre entità figlie: Studente, Segreteria e Docente.

La chiave primaria per Docente e Studente non è più formata da nome utente e matricola o id, ma solo dalla matricola per lo studente, e dall’id per il docente, mentre per la segreteria rimane come chiave il nome utente.

Eliminando in questo modo la generalizzazione, gli attributi che erano prima di Utente sono passati a tutte le entità figlie; pertanto, abbiamo nome utente e password in tutte e tre le entità figlie.

Immagine che contiene testo, schermata, Carattere, diagramma

Descrizione generata automaticamente

Per eliminare la generalizzazione che aveva come padre Corso di Laurea, ho eliminato le entità figlie Triennale e Magistrale assegnando un nuovo attributo a Corso di Laurea che ho chiamato tipo: questo attributo assumerà solo i valori “Triennale” o “Magistrale” e indicherà appunto il tipo di corso di laurea, eliminando così la generalizzazione.

Eliminazione delle ridondanze

La relazione Registrazione tra insegnamento, carriera, esame e docente contiene delle ridondanze: in esame abbiamo già l’id dell’insegnamento e quindi il collegamento con insegnamento è ridondante, anche il collegamento con docente è ridondante poiché quando passeremo al modello relazionale, in esame avremo anche l’id del docente che sarà presente in insegnamento, e quindi abbiamo anche qua una ridondanza.

Risulta quindi necessario formare la relazione solo tra la carriera e l’esame, che contiene tutte le chiavi esterne necessarie.

Di seguito trovate il modello relazionale ristrutturato:

Immagine che contiene schermata, diagramma

Descrizione generata automaticamente

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

I discorsi riguardanti la mancanza di relazioni con segreteria e gli storici rimangono validi.

Sono consapevole della buona norma di evitare chiavi composte ed entità deboli, ma l’aggiunta di un id in esame o nella carriera o storico carriera mi è sembrata superflua e poco sensata da un punto di vista logico.

Passaggio Al Modello Relazionale

Segreteria non ha relazioni, quindi rimane invariata durante la transizione al modello relazionale:

Segreteria(nome\_utente, password, sede)

L’entità Docente fa parte di una relazione uno a molti, ma è dal lato “del molti”, quindi rimane anch’essa invariata:

Docente(id, nome, cognome, nome\_utente, password)

Tra insegnamento e Docente c’è una relazione uno a molti, di conseguenza in insegnamento, che nella relazione è nella “parte dell’uno”, vengono inserite le chiavi di Docente, cioè il suo id.

Insegnamento(id, id\_responsabile, anno, nome, descrizione)

Siccome abbiamo una relazione ricorsiva molti a molti per le propedeuticità, creiamo una relazione ausiliaria dello stesso nome che sarà composta da questi attributi:

Propedeuticità(insegnamento\_a,insegnamento\_b) (Insegnamento\_a e insegnamento\_b formano inoltre la chiave primaria di questa relazione)

Insegnamento a indica l’id dell’insegnamento propedeutico, mentre insegnamento\_b indica l’id dell’insegnamento con propedeuticità.

Abbiamo una relazione uno a molti tra studente e corso di laurea, quindi dovremo inserire in Studente la chiave di corso, cioè il suo id:

Studente(matricola,id\_corso, nome, cognome, anno\_immatricolazione , nome\_utente, password)

L’entità Corso di laurea fa parte di due relazioni, ed entrambe non richiedono, durante la transizione al modello relazionale, l’assimilazione di chiavi delle entità coinvolte nelle relazioni.  
Essa rimane invariata:

Corso(id, nome, tipo)

Tra insegnamento e corso c’è una relazione molti a molti, bisogna quindi creare una relazione ausiliaria:

Composizione(id\_corso,id\_insegnamento, id\_docente) (la chiave primaria verrà formata da questi tre attributi).

Composizione sarà quindi formata da record che contengono l’id del corso, l’id dell’insegnamento e l’id del docente responsabile di tale insegnamento, poiché l’id del docente fa parte delle chiavi di insegnamento.

In questa relazione le tuple indicheranno il corso in cui viene “inserito” l’insegnamento, l’insegnamento stesso e il docente responsabile di tale insegnamento.

Tra Insegnamento ed Esame, abbiamo una relazione uno a molti, in Esame si aggiungeranno quindi le chiavi di insegnamento, inoltre questo era già stato predisposto dal modello ER, avendo stabilito che Esame era un’entità debole.

Esame(data\_esame,id\_docente,id\_insegnamento, nome\_esame)

Tra Studente ed Esame c’è una relazione molti a molti che rappresenta l’iscrizione di uno studente agli esami, bisogna quindi creare una relazione ausiliaria Iscrizione:

Iscrizione(data\_esame, id\_insegnamento,id\_docente, matricola, id\_corso)

data\_esame, id\_insegnamento e id\_docente provengono dalla relazione Esame, mentre matricola e id\_corso da Studente.

Carriera è coinvolta in due relazioni, entrambe uno a molti, la relazione quindi sarà composta dalle chiavi delle entità coinvolte, come prestabilito dalla debolezza raffigurata nel modello ER.

Carriera(matricola, id\_corso, id\_docente, id\_insegnamento, data\_esame, voto)

Matricola e id\_corso provengono da Studente, mentre id\_insegnamento, id\_docente e data\_esame provengono da Esame.

Abbiamo infine le relazioni corrispondenti agli storici:

L’entità Storico Studente fa parte di una relazione uno a molti con Storico Carriera, avendo già descritto la debolezza dell’entità Storico Carriera nel modello ER, sarà quest’ultima ad integrare nella propria relazione la chiave di Storico Studente: la matricola.

Storico\_Studente(matricola, id\_corso, nome, cognome, anno\_immatricolazione, nome\_utente, password)

Storico\_Carriera(matricola, data\_esame, id\_insegnamento, id\_docente, id\_corso, voto).

Normalizzazione

Segreteria(nome\_utente, password, sede)

Questa relazione è in BCNF, in quanto la dipendenza funzionale nome\_utente -> sede password rispetta la definizione di BCNF, essendo nome\_utente chiave di Segreteria.

Docente(id, nome, cognome, nome\_utente, password)

Ad ogni docente corrisponde un id, un professore compare una sola volta in questa tabella, essendo una lista dei docenti.

La dipendenza funzionale di questa relazione: id -> nome cognome nome\_utente password, rispetta la BCNF, poiché id è chiave di Docente.

Insegnamento(id, id\_responsabile, nome, anno, descrizione)

La dipendenza funzionale: id -> id\_responsabile nome anno descrizione è di nuovo in BCNF poiché id è chiave della relazione.

Propedeuticita(insegnamento\_a, insegnamento\_b) è in BNCF essendo entrambi gli attributi chiavi della relazione.

Studente(matricola,id\_corso, nome, cognome, anno\_immatricolazione, nome\_utente, password)

Anche in questo caso, uno studente compare una sola volta in questa relazione, essendo una “lista” degli studenti presenti.  
La dipendenza funzionale matricola -> id\_corso nome cognome anno\_immatricolazione nome\_utente password rispetta la BCNF, essendo matricola la chiave della relazione.

Corso(id, nome, tipo)

Ogni corso compare una volta in questa tabella e non ci sono ridondanze, la dipendenza funzionale della relazione: id ->nome tipo rispetta la BCNF essendo l’id la chiave.

Composizione(id\_corso,id\_insegnamento) è in BCNF, essendo id\_corso-id\_insegnamento la chiave.

Esame(data\_esame,id\_docente,id\_insegnamento, nome\_esame)

La dipendenza funzionale: data\_esame id\_docente id\_insegnamento -> nome\_esame rispetta la BCNF essendo data\_esame id\_docente id\_insegnamento la chiave della relazione.  
Ogni esame in questa relazione compare una sola volta.

Iscrizione(matricola, id\_corso, data\_esame, id\_insegnamento, id\_docente)

Come per Composizione, abbiamo BCNF essendo la relazione formata dalla sua chiave.

Carriera(matricola, id\_corso, data\_esame, id\_insegnamento, id\_docente, voto)

Abbiamo la dipendenza funzionale matricola id\_corso data\_esame id\_insegnamento id\_docente -> voto, questa dipendenza rispetta la BCNF essendo formata dalla chiave della relazione.  
Nella carriera sono presenti gli esami sostenuti dagli studenti, ha quindi senso che i voti dipendano da questi elementi, che identificano (in altre relazioni) rispettivamente lo studente e l’esame sostenuto.

Per le relazioni riguardanti gli storici abbiamo discorsi analoghi a quelli fatti per Studente e Carriera, rispettano entrambe la BCNF.

Le nostre relazioni sono quindi in BCNF, essendo le loro dipendenze funzionali dipendenti dalle loro chiavi, eliminando la possibilità di ridondanze di inserimento, cancellazione e di ridondanza.

Descrizione Delle Funzioni Realizzate

Descrizione Delle Funzioni e Strutture Relative Alle Tabelle

In questa sezione parlerò delle tabelle che ho creato, andando nel dettaglio per quanto riguarda alcune scelte delle loro strutture, e delle funzioni e dei trigger relativi alle tabelle.

Segreteria Docente e Studente

Per la table segreteria ho inserito le colonne:

* nome\_utente varchar(40)
* password varchar(32)
* sede text

Per la table studente ho inserito le colonne:

* matricola (chiave primaria) varchar(6) (ho deciso di usare delle matricole simili a quelle della nostra università)
* id\_corso varchar(6)
* nome text
* cognome text
* anno\_immatricolazione smallint
* nome\_utente varchar(40)
* password varchar(32)

Per quanto riguarda la matricola, controllo con un trigger, che si attiva prima dell’inserimento su ogni riga, che la matricola non sia già stata utilizzata in passato.  
Controllo quindi che non ci sia uno studente archiviato con la stessa matricola.  
(Sono consapevole che a lungo andare questo potrebbe portare a problemi in un caso reale, ma la lunghezza di 6 caratteri della matricola permette di avere milioni e milioni di matricole prima di finire i valori disponibili).

Per questa tabella c’è un trigger che si attiva prima della cancellazione, e si attiva per ogni riga interessata:  
Prima della cancellazione i dati dello studente vengono archiviati nella tabella storico\_studente, e inizia a cancellare i dati della carriera dello studente.  
La tabella carriera avrà a sua volta un trigger analogo che archivia i dati in storico\_carriera prima di cancellarli.

Per la table docente ho inserito le colonne:

* id varchar(6) (simile alle matricole degli studenti)
* nome text
* cognome text
* nome\_utente varchar(40)
* password varchar(32)

Per i nomi utenti ho deciso di dare un massimo di 40 caratteri, mentre per il tipo delle password ho deciso varchar(32), perché md5, il protocollo che ho utilizzato per criptare le password, crea stringhe da 32 caratteri.

Per quanto riguarda la sede della segreteria, i nomi e i cognomi dei docenti e degli studenti ho deciso di usare text (per non avere problemi di limite di caratteri).  
L’utilizzo di text può comportare ad un utilizzo della memoria maggiore rispetto a varchar, ma per quanto riguarda le performance non c’è alcuna differenza: c’è una differenza di tre cicli di CPU per i tipi con una lunghezza predefinita, causati dalla necessità di controllare la lunghezza delle stringhe inserite (fonte https://www.postgresql.org/docs/current/datatype-character.html).

Anno immatricolazione è l’unica colonna che non ha il constraint not null.

Ogni nome\_utente è unique e tramite trigger controllo, prima di inserire e di modificare ogni riga per studente, docente e segreteria, che il nome utente non sia già stato preso da un utente di qualsiasi tipo. (esempio: Prima di modificare o inserire un docente, il database controlla che in segreteria e in studente non ci sia uno studente o una segreteria con lo stesso nome utente).

Insegnamento

In insegnamento abbiamo le colonne:

* id varchar(4)
* id\_responsabile (chiave esterna che referenzia l’id del docente)
* nome text
* anno smallint
* descrizione text

Se il docente responsabile viene eliminato, viene eliminato anche l’insegnamento.

Sia nome che descrizione sono di tipo text per non incappare in problemi legati alle lunghezze dei loro valori.

Per anno c’è un constraint che ammette solo valori tra 1 e 3 compresi.

Ho deciso di inserire il constraint unique per nome: in caso di insegnamenti con programmi simili in corsi di laurea diversi (per esempio analisi 1 e matematica del continuo) si utilizzano dei nomi diversi.   
Ho preso questa decisione per evitare casi in cui corsi di laurea diversi hanno insegnamenti con lo stesso nome ma programmi diversi (e magari anche un docente diverso), evitando confusioni possibili soprattutto per gli studenti.

Per questa tabella ho due trigger: il primo si attiva prima di inserire un nuovo insegnamento ed è di tipo for each row: controlla che il docente responsabile inserito non abbia già tre insegnamenti assegnati.  
Il secondo si attiva prima di modificare una riga, attivandosi per ogni riga affetta da una modifica, che l’anno inserito sia conforme ai corsi di laurea di cui l’insegnamento fa parte.   
Per esempio se un insegnamento che inizialmente ha anno=1 e che fa parte di un corso magistrale, se si modifica l’anno inserendo 3, il trigger ferma la modifica, poiché gli anni accettati dalla magistrale sono 1 e 2.)

Corso

Per la tabella corso ho creato le colonne:

* id (chiave primaria) varchar(6) (esempio di id ‘info01’ per il corso di informatica)
* nome text
* tipo varchar(10)

Tramite constraint, il valore tipo ammette solo i valori ‘Triennale’ e ‘Magistrale’, siccome la lunghezza massima tra le due stringhe è di 10, ho deciso di utilizzare varchar(10).

Esame

Per la tabella esame ho inserito le colonne:

* data\_esame date (qui ho utilizzato il tipo date offerto da postgre)
* id\_insegnamento (chiave esterna per l’insegnamento)
* id\_docente (chiave esterna per docente)
* nome\_esame

nome\_esame è di tipo text, sempre per evitare vincoli sulle lunghezze delle stringhe.

La chiave primaria è formata da data\_esame, id\_insegnamento e id\_docente, permettendo quindi di avere solo un esame corrispondente ad una data per ogni insegnamento specifico. (non si possono quindi inserire più volte gli stessi esami).   
Ho aggiunto nome\_esame per rendere l’iscrizione più semplice per uno studente, che oltre a vedere gli id e la data, avrà anche un nome che lo aiuterà a identificare gli esami.

Se l’insegnamento o il docente vengono eliminati, gli esami vengono rimossi.

Per questa tabella ho creato un trigger che si attiva prima di inserire o modificare un esame, attivandosi per ogni esame (riga) affetta da un inserimento o una modifica:  
Controlla che, nella data inserita, non ci sia già un esame per un altro insegnamento di uno dei corsi di laurea in comune al nuovo esame (o quello che si sta modificando) con lo stesso anno del nuovo esame (o quello che si sta modificando).

Ho anche creato un trigger che si attiva per ogni riga prima dell’inserimento, che controlla che l’id del docente inserito sia quello del docente responsabile dell’insegnamento inserito.

Composizione

Per la tabella composizione ho creato le seguenti colonne:

* id\_corso (chiave esterna per corso)
* id\_insegnamento (chiave esterna per insegnamento)
* id\_docente (chiave esterna per docente)

Tutte queste colonne fanno parte della chiave primaria della tabella, evitando possibili ridondanze e identificando ogni riga. Questa tabella deriva dalla transizione al modello relazionale e permette di inserire un insegnamento in più corsi.

Se i docenti, gli insegnamenti o i corsi vengono eliminati, le righe referenziate vengono eliminate.

Per questa tabella un trigger che si attiva prima dell’inserimento su tutte le righe coinvolte, che controlla che il docente inserito sia quello responsabile dell’insegnamento inserito, e soprattutto controlla che non si stia inserendo un insegnamento del terzo anno in un corso magistrale (che ammette fino al secondo anno).

Iscrizione

Per la tabella iscrizione ho creato le seguenti tabelle:

* matricola (che referenzia studente)
* id\_corso (che referenzia corso)
* id\_insegnamento (che referenzia esame)
* id\_docente (che referenzia esame)
* data\_esame (che referenzia esame)

la chiave primaria è formata da tutte le colonne.

Per questa tabella ho creato un singolo trigger che si attiva prima dell’inserimento su ogni riga.  
Controlla che gli id inseriti siano quelli corretti (per esempio che il docente inserito sia quello responsabile dell’insegnamento, ecc.) e controlla che lo studente soddisfi le propedeuticità (se ce ne sono) dell’esame a cui si sta iscrivendo.

Propedeuticita

Per la tabella propedeuticità ho create le seguenti tabelle:

* insegnamento\_a
* insegnamento\_b

Sono entrambi chiavi esterne su insegnamento e sono chiave primaria della tabella.

Le righe di questa tabella indicano gli insegnamenti propedeutici per l’insegnamento b.

Per esempio, se abbiamo i due corsi Programmazione 2 e Programmazione 1, in insegnamento a verrà inserito l’id di Programmazione 1 e in insegnamento\_b verrà inserito l’id di Programmazione 2.

Se gli insegnamenti vengono cancellati, le righe referenziate vengono eliminate.

Carriera

Per carriera ho creato le seguenti tabelle:

* matricola (chiave esterna per studente)
* id\_corso (chiave esterna per corso)
* id\_docente (chiave esterna per esame)
* id\_insegnamento (chiave esterna per esame)
* data\_esame (chiave esterna per esame)
* voto

La chiave è formata da tutte le colonne tranne il voto, che non può essere null e che ammette solo valori, grazie ad un constraint, tra 0 e 30 compresi.

Se l’esame, il docente o l’insegnamento vengono eliminati, la carriera rimane invariata.

Come per studente, è presente un trigger che si attiva prima di cancellare le righe, che archivia tutta la carriera nello storico analogo.

Ho anche creato un trigger che si attiva prima di inserire i dati che controlla che gli id siano corretti.

Storico Studente

Per storico studente ho creato le seguenti colonne:

* matricola (chiave primaria)
* id\_corso
* nome
* cognome
* anno\_immatricolazione
* nome\_utente
* password

In questa tabella ho deciso di non far referenziare l’id del corso alla tabella corso, perché il solo scopo di questa tabella è quello di mantenere i dati degli studenti che vengono rimossi, e i dati inseriti in questa tabella sono una copia diretta dei dati della tabella studente.

Storico Carriera

Per storico carriera ho creato le seguenti colonne:

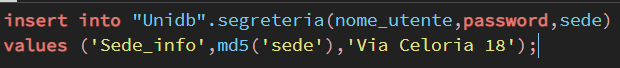
* matricola (che referenzia la matricola dello storico\_studente)
* id\_corso
* id\_insegnamento
* id\_docente
* data\_esame
* voto

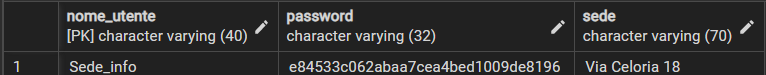
Ho deciso di non inserire i vari id come chiavi esterne, perché i dati inseriti in questa tabella sono delle copie della tabella carriera, che effettua già i controlli sui dati inseriti grazie alle sue chiavi esterne, inoltre l’unica operazione che coinvolge questa tabella e lo storico studente è quella di visualizzazione della carriera completa e valida di studenti passati e ci interessa in qualsiasi caso mantenere i dati come sono al momento dell’inserimento.

Come spiegato per ogni tabella, tutti i trigger che ho creato sono stati definiti for each row (per ogni riga) poiché i controlli da effettuare sugli inserimenti, le modifiche e le cancellazioni sono da effettuare per tutte le righe coinvolte in una update, insert o delete.  
Se avessi definito questi trigger come for each statement (per ogni comando), questi controlli non sarebbero stati efficaci né possibili, perché le variabili NEW e OLD non vengono inizializzate da postgresql per le funzioni di trigger a livello for each statement.

Prove di Funzionamento

Dato il seguente utente di tipo segreteria:





Procederò a mostrare delle immagini delle funzionalità della mia applicazione:

Login:

Immagine che contiene nuvola, testo, aria aperta, cielo

Descrizione generata automaticamente

Fatto l’accesso, si apre una pagina contenente una dashboard con tutte le operazioni che l’utente può effettuare (La dashboard presentata è quella per gli utenti segreteria):

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

Le operazioni che mostrerò ora sono operazioni che gli utenti di tipo segreteria possono svolgere:

Creazione corso di laurea:

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

Visualizzazione dei corsi (in caso di mancanza di risultati al posto della tabella il sito web mostrerà un messaggio all’utente, alla fine della sezione mostrerò un esempio):

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Modifica corso:

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Creazione Docente:

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

Visualizzazione docenti (nella parte web manca la colonna corrispondente alle password, ho deciso di non far mostrare alla segreteria le password degli utenti):  
(in caso di mancanza di risultati il sito mostrerà un messaggio)

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

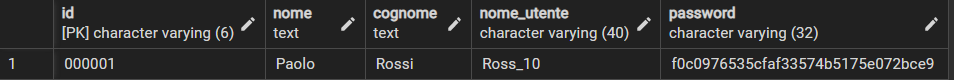
Immagine che contiene testo, Software multimediale, software, schermata

Descrizione generata automaticamente

Modifica docenti:

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente



Creazione insegnamento (e assegnamento responsabile):

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

Se il docente inserito è già responsabile di 3 insegnamenti:



Visualizzazione insegnamenti:

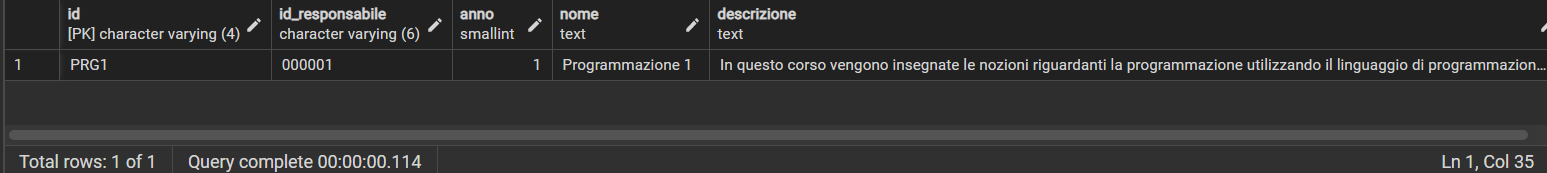


Immagine che contiene testo, Carattere, linea, schermata

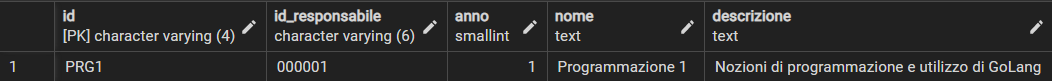
Descrizione generata automaticamente

Nel caso non ci siano risultati, il sito mostrerà un messaggio all’utente.

Modifica dati insegnamento:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente



Inserimento insegnamento in corso di laurea:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Inserimento propedeuticità (dopo aver creato un insegnamento di nome Programmazione 2 e averlo inserito nel corso di Informatica):

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Prova di funzionamento delle ultime due funzionalità e pagina visualizzazione informazioni di un corso: (nel caso non si trovino informazioni sul corso o sulle propedeuticità il sito mostrerà dei messaggi relativi all’utente)

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Nel caso non si trovino informazioni riguardanti il corso o le propedeuticità il sito mostrerà un messaggio che informerà l’utente.

Rimozione insegnamento:

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente



Immagine che contiene testo, Software multimediale, Carattere, software

Descrizione generata automaticamente

Rimozione docente:

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente



Immagine che contiene testo, Carattere, Software multimediale, software

Descrizione generata automaticamente

(per popolare il database ho rimesso gli elementi appena cancellati)

Creazione studente:

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

Prova di funzionamento e pagina di visualizzazione studenti attivi (anche qua è presente un messaggio in caso di mancanza di risultati):



Immagine che contiene schermata, testo, Software multimediale

Descrizione generata automaticamente

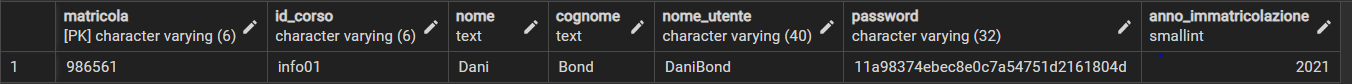
Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Modifica studente:

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente



Archiviazione Studente (rimozione dello studente dalla lista degli studenti attivi e passaggio dei suoi dati negli storici):

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, design

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Carriera completa studente archiviato:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

Carriera valida studente archiviato:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

Passo ora alle operazioni dei docenti:

Immagine che contiene testo, schermata, cielo, design

Descrizione generata automaticamente

Dashboard docente:

Immagine che contiene testo, schermata, Carattere, numero

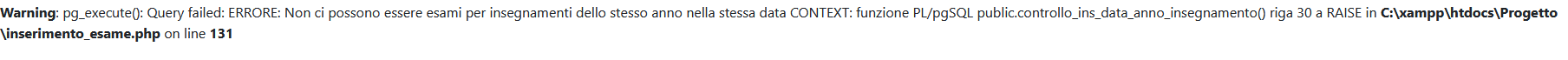
Descrizione generata automaticamente

Creazione Esame:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Se il docente prova ad inserire un esame in una data dove è già presente l’esame di un insegnamento dello stesso anno dello stesso corso di laurea:



Modifica Esami:

Vengono mostrati al docente i suoi esami e grazie al menù a tendina può selezionare quale esame modificare: (se il docente non ha esami verrà mostrato un messaggio)

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Visualizzazione Esami e prova funzionamento operazione precedente (anche qua è presente nel caso di mancanza di risultati un messaggio):

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

Pagina che consente al docente di vedere i suoi insegnamenti (messaggio in caso di mancanza di risultati):

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

Inserimento voti:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Vengono mostrati al docente gli studenti iscritti a quell’esame (l’iscrizione è stata effettuata per la prova di funzionamento e verrà mostrata a seguire nel documento, quando arriverò a parlare delle operazioni per gli studenti).

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Il docente seleziona lo studente e inserisce il voto:

Immagine che contiene testo, Carattere, schermata, design

Descrizione generata automaticamente

La prova di funzionamento di questa pagina verrà mostrata nella pagina per lo studente che permette di mostrare la sua carriera.

Passo ora alle operazioni degli studenti:

Immagine che contiene testo, schermata, cielo, design

Descrizione generata automaticamente

Dashboard studente:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Iscrizione Esame:

Vengono mostrati allo studente gli esami degli insegnamenti del suo corso di laurea e con il menu a tendina può selezionare a quale esame iscriversi (in caso di mancanza di risultati viene mostrato un messaggio):

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

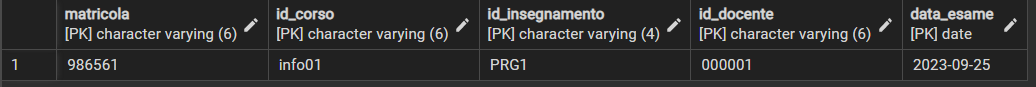
Immagine che contiene testo, linea, Carattere, Blu elettrico

Descrizione generata automaticamente

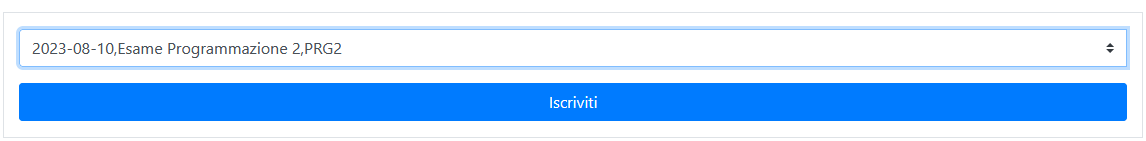
Prova di funzionamento operazione precedente e pagina di visualizzazione iscrizioni dello studente loggato:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente



Se lo studente prova ad iscriversi ad un esame per cui non rispetta le propedeuticità viene sollevata un’eccezione dai trigger:





Pagina visualizzazione carriera completa:

(ho inserito l’esito di un altro esame per poter poi confrontare carriera completa e carriera valida)

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

Pagina visualizzazione carriera valida:

Immagine che contiene testo, linea, Carattere, schermata

Descrizione generata automaticamente

Esempio di messaggio mostrato nelle pagine di visualizzazione dati:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Messaggi del genere compaiono nei casi in cui non ci sono risultati per le visualizzazioni richieste, ovviamente i messaggi sono diversi in base al contesto.

Per ogni utente è disponibile grazie alla navbar la pagina di gestione utente:



(nel caso d’esempio sono loggato con l’utente del docente Paolo Rossi (utente rosso password rosso):

prima della modifica:

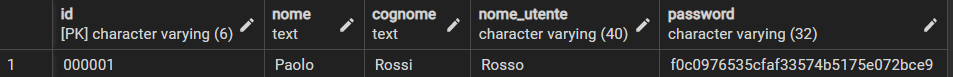
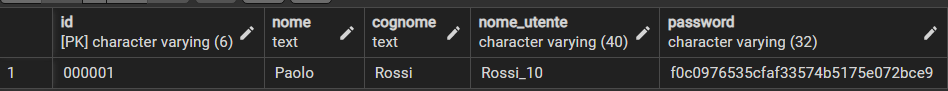


Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

La password è rimasta invariata (ho reinserito la stessa password):

Dopo la modifica:



Note finali riguardanti l’implementazione web

Ho deciso di non permettere alla segreteria di cancellare i corsi di laurea, perché avrebbe effetti troppo onerosi sugli studenti, e il resto delle entità coinvolte, l’ho trovato inoltre un caso troppo specifico per rappresentarlo.

Dopo l’inserimento tramite form, PHP o Xampp segnalano una linea: non è un errore ma una segnalazione dell’attivazione di un input nascosto che ho utilizzato per creare una variabile di sessione. Questa variabile e questo input nascosti li ho creati per evitare che, dopo aver ricaricato (tramite F5 per esempio) la pagina, essa ritenti di inserire i dati appena inseriti.