Database esami_università

Progetto di Laboratorio di Basi di dati

Oscar Mostafa Fath el bab

Matricola: 01415A

Primo appello del 15 giugno 2023

Indice

1 Introduzione

Questo elaborato ha lo scopo di illustrare la progettazione e implementazione di una base di dati di una piattaforma per la gestione degli esami universitari.

La base di dati è stata implementata utilizzando *PostgreSQL* (Database Management System). Inoltre, per interfacciarsi con la base di dati si è anche sviluppata un'applicazione web, la quale è stata realizzata utilizzando tecnologie come *PHP*, *HTML* e *CSS*.

Per usare l'applicazione web è necessario seguire le istruzioni di installazione e le indicazioni per l'uso che si trovano all'interno del manuale utente allegato.

2 Documentazione tecnica

2.1 Progettazione concettuale

2.1.1 Schema ER della base di dati

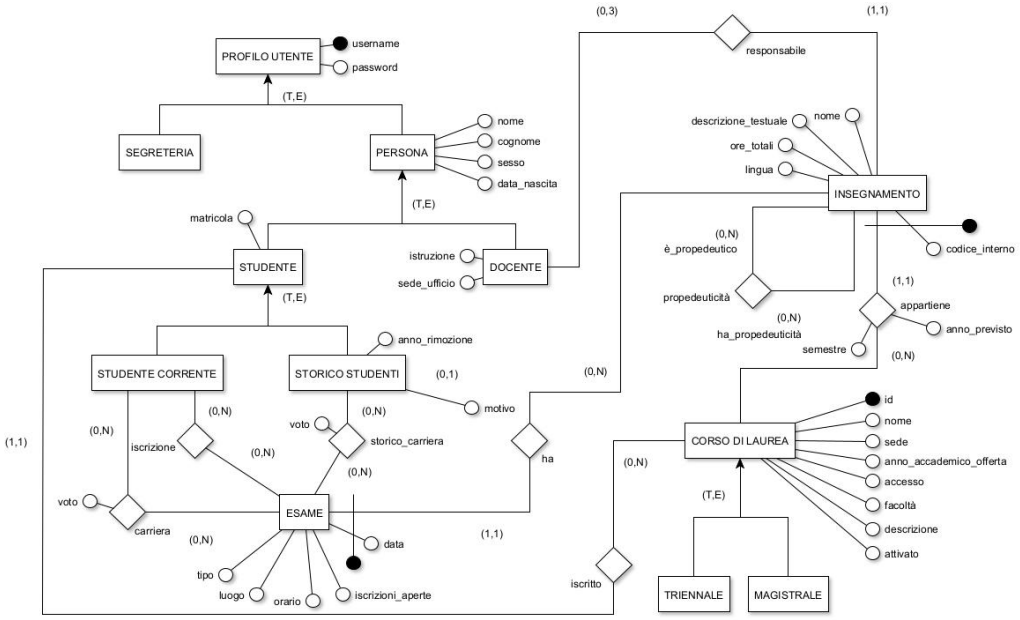

Figura 1: Schema ER della base di dati.

Entità

Posteriormente alle fasi di analisi dei requisiti e identificazione delle funzionalità da sviluppare, sono state scelte come entità, conforme al modello *Entity-Relationship*, le seguenti classi di oggetti che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione in questione:

- *Profilo utente*, rappresenta gli utenti della piattaforma, ed ha come sottoentità:
  - *Segreteria*, rappresenta gli utenti della segreteria dell'università.
  - *Persona*, rappresenta gli utenti della piattaforma considerati persone, ha come sottoentità:
    - *Studente*, rappresenta gli studenti dell'università, a sua volta a come sottoentità:
      - *Studente corrente*, rappresenta gli studenti correnti dell'università.
      - *Storico studenti*, rappresenta gli ex-studenti dell'università.
    - *Docente*, rappresenta i docenti dell'università.
- *Esame*, rappresenta gli esami dei diversi insegnamenti appartenenti ai corsi di laurea dell'università.
- *Insegnamento*, rappresenta gli insegnamenti nei diversi corsi di laurea dell'università.
- *Corso di laurea*, rappresenta i corsi di laurea dell'università, avendo come sottoentità:
  - *Triennale*, rappresenta i corsi di laurea triennali.
  - *Magistrale*, rappresenta i corsi di laurea magistrali.

Gerarchie di generalizzazione

Alcune entità sono in relazione tra di loro mediante una gerarchia di generalizzazione, dove coesistono entità padre ed entità figlie, di seguito si illustreranno le gerarchie di generalizzazione presenti nello schema ER:

- La gerarchia che esiste tra *Profilo utente*, il quale svolge il ruolo di entità padre, e le entità figlie *Segreteria* e *Persona*, è una gerarchia

di tipo (T,E) (Totale - Esclusiva), dal momento che un profilo utente, nel dominio applicativo interessato, può essere solamente un utente segreteria oppure essere un utente che rappresenta una persona, e non entrambi allo stesso tempo, dato che un utente segreteria non può mai rappresentare una persona e viceversa.

- La gerarchia tra *Persona* (entità padre), *Studente* e *Docente* (entità figlie), è di tipo (T,E) (Totale - Esclusiva), giacché una persona, in questo contesto, può essere solamente uno studente o un docente, e non entrambi contemporaneamente per rendere indipendente il fatto che una persona sia uno studente dal fatto che una persona sia un docente.

- La gerarchia tra *Studente* (entità padre), *Studente corrente* e *Storico studenti* (entità figlie), è di tipo (T,E) (Totale - Esclusiva), considerando che uno studente, può essere solamente uno studente corrente oppure uno studente appartenente allo storico studenti, inoltre, la scelta del vincolo di disgiunzione è motivata dal fatto che, anche se uno studente è uno studente corrente e allo stesso tempo è anche un ex-studente, viene considerato come due oggetti indipendenti, perché nell'applicazione bisogna distinguere in maniera totale gli studenti correnti dagli ex-studenti, dato che ogni caso potrebbe avere molte caratteristiche diverse dall'altro.

- La gerarchia tra *Corso di laurea* (entità padre), *Triennale* e *Magistrale* (entità figlie), è di tipo (T,E) (Totale - Esclusiva), in quanto un corso di laurea può essere solamente triennale o magistrale, ed escludendo il caso di laurea magistrale a ciclo unico (caso non trattato nel progetto per semplicità), per motivi evidenti un corso di laurea non può essere sia triennale che magistrale.

Attributi e identificatori

A continuazione, si indicheranno i corrispondenti attributi e identificatori per ogni entità:

- *Profilo utente*:
  - *username*: Identificatore (e attributo) che indica il nome utente di un fruitore della piattaforma.
  - *password*: Attributo che indica la password utilizzata da ogni utente.

- *Segreteria*: Non ci sono attributi particolari per questa entità.
- *Persona*:
  - *nome*: Attributo che indica il nome di una persona.
  - *cognome*: Attributo che indica il cognome di una persona.
  - *sesso*: Attributo che indica il sesso di una persona.
  - *data_nascita*: Attributo che indica la data di nascita di una persona.
- *Studente*:
  - *matricola*: Attributo che indica la matricola di uno studente, la quale sarà formata da un numero intero appartenente all'intervallo $[900000, 999999]$ per convenienza.
- *Docente*:
  - *istruzione*: Attributo che indica il livello di istruzione conseguito da un docente.
  - *sede_ufficio*: Attributo che indica l'indirizzo dell'ufficio di un docente.
- *Studente corrente*: Non ci sono attributi particolari per questa entità.
- *Storico studenti*:
  - *anno_rimozione*: Attributo che indica l'anno di rimozione di un ex-studente.
  - *motivo*: Attributo che indica il motivo della rimozione di un ex-studente, tra cui si distinguono i valori più importanti che appartengono al dominio associato, come sono la laurea e la rinuncia. Si noti che questo attributo ha cardinalità minima 0, dato che ci sono situazioni in cui non è possibile determinare a priori il motivo della rimozione di un ex-studente.
- *Esame*:
  - *data*: Attributo che indica la data in cui si svolge un esame.
  - *orario*: Attributo che indica l'orario in cui si terrà un esame.
  - *tipo*: Attributo che indica la tipologia di un esame, quest'ultimo può essere solamente di tipo "Scritto", "Laboratorio" oppure "Orale" per ragioni di semplicità.

- *luogo*: Attributo che indica il luogo dove si terrà un esame.
- *iscrizioni_aperte*: Attributo booleano che indica se le iscrizioni di un esame sono aperte.

L'identificatore in questo caso è formato dalla coppia (*data*, *insegnamento*), dove con *insegnamento* ci si riferisce alla relazione binaria tra *Esame* e *Insegnamento*, la quale verrà discussa nella sezione successiva. In questo caso si dice che l'entità *Esame* è debole.

- *Insegnamento*:
  - *codice_interno*: Attributo che indica il codice interno di un insegnamento.
  - *nome*: Attributo che indica il nome di un insegnamento.
  - *descrizione_testuale*: Attributo che indica la descrizione testuale di un insegnamento.
  - *ore_totali*: Attributo che indica le ore totali di un insegnamento.
  - *lingua*: Attributo che indica la lingua in cui è impartito un insegnamento.

  Anche in questo caso, l'identificatore è formato dalla coppia (*codice_interno*, *corso_di_laurea*), dove con *corso_di_laurea* ci si riferisce alla relazione binaria tra *Insegnamento* e *Corso di laurea*, la quale verrà discussa anch'essa nella sezione successiva. Inoltre, anche in questo caso si dice che l'entità *Insegnamento* è debole.

- *Corso di laurea*:
  - *nome*: Attributo che indica il nome di un corso di laurea.
  - *sede*: Attributo che indica la sede di un corso di laurea.
  - *anno_accademico_offerta*: Attributo che indica l'anno accademico a partire dal quale vale l'offerta rappresentata da un corso di laurea.
  - *id*: Identificatore (e attributo) che indica l'id di un corso di laurea, la scelta di utilizzare l'attributo id per identificare univocamente un istanza dell'entità interessata è motivata dal fatto che un'istanza di *Corso di laurea* è distinguibile da altre istanze della stessa entità solamente se, oltre al nome, la sede e l'anno accademico dell'offerta, si conosce anche se il corso di laurea è triennale o magistrale, e per motivi legati alla costruzione della gerarchia di

generalizzazione che ha come entità padre *Corso di laurea*, non si può determinare immediatamente se un'istanza di essa appartiene al sottoinsieme di corsi di laurea triennali oppure al sottoinsieme di corsi di laurea magistrali. Inoltre, questo attributo porta un vantaggio aggiuntivo, che è un miglioramento delle prestazioni nelle query sulla base di dati, dato che sarà un tipo di dato compatto composto da 3 caratteri alfanumerici, dei quali il primo e l'ultimo saranno lettere maiuscole nell'alfabeto $[A - Z]$ e il secondo carattere sarà un numero intero nell'intervallo $[0 - 9]$, a differenza di utilizzare un identificatore composto dove le prestazioni si ridurrebbero perché si dovrebbe interrogare la chiave su diverse colonne, e le operazioni di join diventerebbero più costose.

  - *accesso*: Attributo che indica il tipo di accesso che si ha su un corso di laurea, in questo attributo si può solamente assumere i valori "Libero" e "Programmato".
  - *facoltà*: Attributo che indica la facoltà a cui appartiene un corso di laurea.
  - *descrizione*: Attributo che indica la descrizione di un corso di laurea.
  - *attivato*: Attributo booleano che indica se un corso di laurea è stato attivato, cioè se è possibile che abbia studenti iscritti, la scelta dell'utilizzo di questo attributo è motivata dalla necessità di distinguere tra corsi di laurea in fase di definizione e corsi di laurea definiti in maniera completa.

- *Triennale*: Non ci sono attributi particolari per questa entità.
- *Magistrale*: Non ci sono attributi particolari per questa entità.

Nota: Si consideri che le entità figlie di una gerarchia di generalizzazione, ereditano gli identificatori e gli attributi della corrispondente entità padre. Inoltre, si osservi che tutti gli attributi hanno cardinalità $(1, 1)$ eccetto l'attributo *motivo* dell'entità *Storico studenti*.

Associazioni

Sono state determinate le seguenti associazioni, ognuna di esse rappresenta un legame logico tra le entità che collega, significativo per la piattaforma:

- *iscritto* (Studente - Corso di laurea): Associazione binaria uno-a-molti che descrive la relazione *"Studente è iscritto a Corso di laurea"*, avente

come vincolo di cardinalità dal lato di *Studente*: $(1, 1)$, dato che uno studente deve essere iscritto esattamente ad un corso di laurea, invece, dal lato di *Corso di laurea* si ha: $(0, N)$, poiché un corso di laurea può non avere iscritti, e al massimo può averne N.

- *appartiene* (Insegnamento - Corso di laurea): Associazione binaria uno-a-molti che esprime il fatto *"Insegnamento appartiene a Corso di laurea"*, avente come vincolo di cardinalità dal lato di *Insegnamento*: $(1, 1)$, considerando che un insegnamento deve appartenere esattamente ad un corso di laurea, invece, dal lato di *Corso di laurea* si ha: $(0, N)$, dato che un corso di laurea può avere 0 insegnamenti, e.g. corso di laurea in corso di definizione, e come massimo N insegnamenti. Inoltre, questa associazione ha due attributi, *anno_previsto* che indica l'anno in cui è previsto un insegnamento all'interno di un corso di laurea e *semestre* che indica il semestre in cui è previsto un insegnamento all'interno di un corso di laurea.

- *propedeuticità* (Insegnamento - Insegnamento): Associazione ricorsiva molti-a-molti che esprime il fatto *"Insegnamento 1 è propedeutico a Insegnamento 2"* o il fatto *"Insegnamento 1 ha come propedeuticità Insegnamento 2"*, avente come vincolo di cardinalità in entrambi i lati: $(0, N)$, tenendo conto che in un lato che un insegnamento può essere propedeutico a 0 insegnamenti e al massimo ad N, e dall'altro lato che un insegnamento può avere come propedeuticità 0 insegnamenti, e al massimo può avere come propedeuticità N insegnamenti.

- *responsabile* (Docente - Insegnamento): Associazione binaria molti-a-uno che esprime il fatto *"Docente è responsabile di Insegnamento"*, avente come vincolo di cardinalità dal lato di *Docente*: $(0, 3)$, alla luce del fatto che un docente può essere responsabile di 0 insegnamenti, e.g. docente nuovo, e al massimo può essere responsabile di 3 insegnamenti per la natura del dominio applicativo, inoltre è stato scelto che questo vincolo di cardinalità massima valga solamente per ogni anno accademico offerta per fare in modo che il vincolo risulti più ragionevole, dall'altro lato invece, un insegnamento deve avere come responsabile esattamente un docente.

- *ha* (Insegnamento - Esame): Associazione binaria molti-a-uno che esprime il fatto *"Insegnamento ha Esame"*, avente come vincolo di cardinalità dal lato di *Insegnamento*: $(0, N)$, visto che che un insegnamento può avere 0 esami associati, e al massimo può averne N, dall'altro lato invece, un esame deve essere associato ad esattamente un insegnamento.

- *carriera* (Studente corrente - Esame): Associazione binaria molti-a-molti che esprime il fatto *"Studente corrente ha in carriera Esame"*, avente come vincolo di cardinalità in entrambi i lati: $(0, N)$, posto che da un lato uno studente corrente può avere 0 esami in carriera, e al massimo può averne N, dall'altro lato invece, un esame può trovarsi nella carriera di 0 studenti correnti, e al massimo trovarsi nella carriera di N studenti correnti. Inoltre, questa associazione ha un attributo *voto* che indica il voto verbalizzato in carriera dell'esame sostenuto da uno studente.

- *storico_carriera* (Storico studenti - Esame): Associazione binaria molti-a-molti che esprime il fatto *"Ex-studente aveva in carriera Esame"*, avente come vincolo di cardinalità in entrambi i lati: $(0, N)$, in considerazione del fatto che da un lato un ex-studente avrebbe potuto avere 0 esami in carriera al momento della sua rimozione, e al massimo avrebbe potuto averne N, dall'altro lato invece, un esame potrebbe essersi trovato nella carriera di 0 ex-studenti, e al massimo essersi trovato nella carriera di N ex-studenti. Inoltre, questa associazione ha un attributo *voto* che indica il voto verbalizzato in carriera dell'esame sostenuto da un ex-studente.

- *iscrizione* (Studente corrente - Esame): Associazione binaria molti-a-molti che esprime il fatto *"Studente corrente è iscritto ad Esame"*, avente come vincolo di cardinalità in entrambi i lati: $(0, N)$, avendo presente che da un lato uno studente corrente può essere iscritto a 0 esami, e al massimo può essere iscritto a N esami, dall'altro lato invece, un esame può avere 0 studenti correnti iscritti, e al massimo avere N studenti correnti iscritti.

  Nota: Si osservi che è stato scelto di separare in due associazioni diverse il concetto di iscrizione ad un esame e il concetto di valutazione di un esame, questa scelta non è triviale, siccome inizialmente sembrerebbe che le istanze di queste associazioni seguano uno schema simile, ma se si considerasse la scelta di avere un unica tabella dove si registrano le iscrizioni e gli esami in carriera, avremmo un attributo voto con cardinalità minima 0, dato che, ad esempio al momento dell'iscrizione ad un esame, il voto non è conosciuto, a questo punto è facile rendersi conto che questo design porta ad una riduzione delle prestazioni al momento di interrogare la relativa tabella, perché si deve considerare il costo aggiuntivo che suppone la selezione delle righe che hanno valore NULL/NOT NULL (a seconda della query). In più, non è raro, il caso in cui il voto sia posto a NULL, per cui si deve anche considerare

lo spazio che occuperebbero complessivamente tutte le righe poste a NULL.

Oltre a ciò, anche se a priori sembrerebbe che i due concetti seguano uno stesso schema, questo non è per forza vero, ad esempio potrebbero essere aggiunte altre informazioni associate all'iscrizione che non sono direttamente associate con la valutazione di un voto, e.g. la data in cui è stata effettuata l'iscrizione.

2.1.2 Ristrutturazione dello schema ER

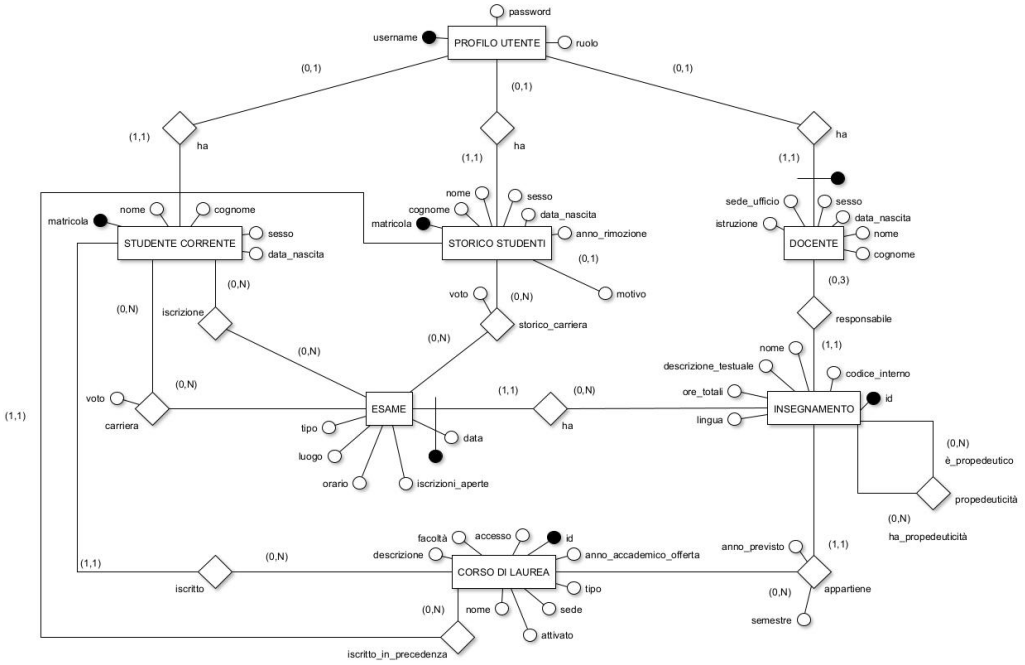

Figura 2: Schema ER ristrutturato della base di dati.

Dopo aver determinato che non ci sono né relazioni derivate, né attributi derivati, composti o multivalore, si procede con l'eliminazione delle gerarchie di generalizzazione:

Eliminazione delle gerarchie di generalizzazione

Si osservi che questa fase è stata realizzata usando una strategia che parte dal basso della gerarchia, siccome si dovevano gestire gerarchie con più di 2 livelli.

A continuazione verranno dettagliati gli aspetti metodologici utilizzati su ogni eliminazione di gerarchia di generalizzazione:

- La gerarchia tra *Studente*, *Studente corrente* e *Storico studenti* è stata eliminata mantenendo solo le entità figlie, questa scelta è stata basata sul fatto che la maggioranza delle operazioni usa contemporaneamente attributi dell'entità generica e attributi di singole specializzazioni e inoltre, la gerarchia è di tipo (T,E). Questa scelta assicura un risparmio di memoria rispetto a mantenere solo l'entità padre, per assenza di valori nulli, in aggiunta, migliora le prestazioni rispetto a mantenere tutte le entità perché non si visita l'entità padre per accedere a determinati attributi.

- La gerarchia, formata dopo l'eliminazione descritta nel punto precedente, tra *Persona*, *Studente corrente*, *Storico studenti* e *Docente* è stata eliminata mantenendo solo le entità figlie, questa scelta è stata influenzata dal fatto che, anche in questo caso, la maggioranza delle operazioni usa contemporaneamente attributi dell'entità generica e attributi di singole specializzazioni, oltre a ciò anche qui abbiamo una gerarchia di tipo (T,E). Questa scelta offre gli stessi vantaggi descritti nel punto precedente.

- La gerarchia, formata dopo l'eliminazione descritta nel punto precedente, tra *Profilo utente*, *Studente corrente*, *Storico studenti*, *Docente* e *Segreteria* è stata eliminata mantenendo tutte le entità per quanto riguarda *Profilo utente*, *Studente corrente*, *Storico studenti* e *Docente*, questa scelta è stata guidata dal fatto che, la maggioranza delle operazioni usa attributi dell'entità generica e attributi di singole specializzazioni anche non contemporaneamente (e.g. login). Questa scelta assicura un risparmio di memoria rispetto all'opzione di mantenere solo l'entità padre, per assenza di valori nulli e riduce i tempi di accesso rispetto all'opzione di mantenere solo le entità figlie per un numero minore di attributi; per *Segreteria* invece è stato deciso di mantenere solo l'entità padre, cioè *Profilo utente*, dato che nel caso di un utente segreteria non si hanno attributi caratteristici, per cui un utente segreteria non è distinguibile da un altro utente segreteria per quanto riguarda gli attributi della specializzazione, questa scelta porta ad un miglioramento delle prestazioni dovuto alla riduzione del numero di accessi rispetto a mantenere solo le entità figlie oppure rispetto a mantenere tutte le entità.

- La gerarchia tra *Corso di laurea*, *Triennale* e *Magistrale* è stata eliminata mantenendo solo l'entità padre, questa scelta è gisutificata dal

fatto che, la maggioranza delle operazioni usa attributi dell'entità generica e non fa distinzione tra le istanze delle specializzazioni. Questa scelta assicura un minor numero di accessi rispetto a mantenere solo le entità figlie o rispetto a mantenere tutte le entità, come nel caso della gerarchia di *Segreteria*.

Modifiche agli identificatori delle entità

- Per *Studente corrente* e *Storico studenti* è stato scelto come identificatore l'attributo *matricola*, dato che è un attributo NOT NULL, identifica univocamente un istanza delle entità, ed è una chiave compatta (chiave intera con un dominio finito), inoltre, essa è la chiave naturale delle entità.

- Per *Insegnamento* è stato scelto come identificatore, un nuovo attributo id che è NOT NULL e identifica univocamente un insegnamento, questa scelta è stata presa per ridurre il numero di attributi che compongono l'identificatore di questa entità, dato che, l'entità *Esame* utilizza come identificatore l'associazione con *Insegnamento* più l'attributo *data*, quindi senza cambiare l'identificatore questa tabella avrebbe 3 attributi come identificatore, ciò può portare ad una riduzione delle prestazioni in caso di join o selezione delle istanze nell'entità. Vale la pena sottolineare che questa scelta porta vantaggi anche per l'entità *Insegnamento*, la quale otterrà una migliore indicizzazione insieme ad un ottimizzazione delle query per la riduzione del numero di attributi che compongono il suo identificatore.

Vincoli extra-schema

I seguenti vincoli verranno implementati attraverso regole attive:

1. L'eliminazione di uno studente corrente comporta lo spostamento delle sue informazioni in Storico studenti, e la sua carriera esami in Storico carriera.

2. L'iscrizione ad un esame da parte di uno studente corrente si deve effettuare solamente se lo studente non ha ancora verbalizzato un voto per quell'esame, si sta iscrivendo ad un esame del suo corso di laurea e lo studente rispetta tutte le propedeuticità.

3. Gli esami di insegnamenti previsti per lo stesso anno in uno stesso corso di laurea non si possono programmare in una stessa giornata.

4. Un corso di laurea magistrale non può avere un insegnamento previsto per il terzo anno.

5. Un docente può essere responsabile al massimo di 3 insegnamento in un offerta di un corso di laurea.

6. Un profilo utente deve appartenere in modo mutuamente esclusivo ad uno studente o ad un ex-studente.

7. Un profilo utente con ruolo Docente o Segreteria non può essere assegnato ad uno studente o ad un ex-studente.

8. Un profilo utente con ruolo Studente o Segreteria non può essere assegnato ad un Docente.

9. La data di un esame deve essere sempre successiva al giorno odierno.

10. Le propedeuticità si devono definire solamente all'interno di un corso di laurea, e gli anni previsti e i semestri degli insegnamenti propedeutici devono essere coerenti con quelli dell'insegnamento a cui si accede rispettando la propedeuticità.

11. Gli studenti ed ex-studenti devono essere iscritti solamente a corsi di laurea attivati.

12. Non si possono eliminare docenti responsabili di almeno un insegnamento.

13. Non si possono eliminare né insegnamenti appartenenti a corsi di laurea già attivati, né corsi di laurea già attivati.

14. Non si possono eliminare esami per i quali è stato già verbalizzato un voto.

15. Non si possono creare insegnamenti in corsi di laurea già attivati.

16. Le matricole devono essere uniche tra gli studenti e gli ex-studenti.

17. Gli studenti correnti si possono iscrivere solamente ad esami che hanno le iscrizioni aperte.

2.2 Progettazione logica

2.2.1 Schema logico (relazionale) della base di dati

Dopo la ristrutturazione dello schema ER, si è effettuata la traduzione dello schema ER in uno schema logico, cercando di ridurre il più possibile il numero di tabelle ma allo stesso tempo facendo scelte sensate per l'ottimizzazione delle query, struttura logica, ecc.

A continuazione il risultato:

profilo_utente(username, password, ruolo)

studente_corrente(matricola, username, nome, cognome, sesso, data_nascita, corso_di_laurea)

storico_studenti(matricola, username, nome, cognome, sesso, data_nascita, corso_di_laurea, motivo*, anno_rimozione)

docente(username, nome, cognome, sesso, data_nascita, istruzione, sede_ufficio)

esame(insegnamento, data, orario, tipo, luogo, iscrizioni_aperte)

insegnamento(id, corso_di_laurea, codice_interno, nome, docente, anno_previsto, semestre, ore_totali, lingua, descrizione_testuale)

corso_di_laurea(id, nome, tipo, anno_accademico_offerta, facoltà, descrizione, sede, accesso, attivato)

carriera(matricola, data, insegnamento, voto)

iscrizione(matricola, data, insegnamento)

storico_carriera(matricola, data, insegnamento, voto)

propedeuticità(insegnamento_1, insegnamento_2)

2.3 Funzioni realizzate e prove di funzionamento

2.3.1 Viste

Carriera completa

Implementata mediante una vista (non materializzata), la produzione della carriera completa di un qualsiasi studente è accessibile completamente da un qualsiasi utente segreteria e accessibile in maniera parziale da un utente studente (solamente per quanto riguarda la sua carriera completa).

E' stata effettuata l'unione tra le tabelle *carriera* e *storico_carriera*, proiettando sugli attributi delle tabelle *matricola*, *data* e *voto* e sugli attributi *id* e *nome*, i quali sono stati ottenuti attraverso join con la tabella *insegnamento*.

Prova:

Carriere in archivio

MATRICOLA	ID INSEGNAMENTO	NOME INSEGNAMENTO	DATA ESAME	VOTO
900000	F1X002	Programmazione 1	2023-09-13	23
900000	F1X003	Matematica del continuo	2023-09-22	17
900000	F1X002	Programmazione 1	2023-09-23	30
900001	F1X001	Architettura degli elaboratori 1	2023-09-11	20
900001	F1X002	Programmazione 1	2023-09-13	28
900001	F1X003	Matematica del continuo	2023-09-22	15
900001	F1X002	Programmazione 1	2023-09-23	29
900003	C2X001	Istituzioni di matematica	2023-09-14	21

Figura 3: Carriera completa su un utente segreteria.

Carriera completa

Contiene tutti gli esami sostenuti

ID INSEGNAMENTO	NOME INSEGNAMENTO	DATA ESAME	VOTO
F1X002	Programmazione 1	2023-09-13	23
F1X003	Matematica del continuo	2023-09-22	17
F1X002	Programmazione 1	2023-09-23	30

Figura 4: Carriera completa sull'utente studente con matricola 900000.

Carriera valida

Implementata mediante una vista (non materializzata), la produzione della carriera valida di un qualsiasi studente è accessibile completamente da un qualsiasi utente segreteria e accessibile in maniera parziale da un utente studente (solamente per quanto riguarda la sua carriera valida).

E' stata effettuata l'unione tra le tabelle *carriera* e *storico_carriera*, ordinando per *matricola*, *id* dell'insegnamento e *data* in modo decrescente, selezionando, senza ottenere duplicati sulla coppia (*matricola*, *id*), solamente i voti maggiori o uguali a 18, proiettando poi, sugli attributi delle tabelle *matricola*, *data* e *voto* e sugli attributi *id* e *nome*, i quali sono stati ottenuti attraverso join con la tabella *insegnamento*. Alla fine di tutto ciò viene effettuato un ordinamento sull'attributo *id* in modo crescente.

Prova:

Carriere valide in archivio

MATRICOLA	ID INSEGNAMENTO	NOME INSEGNAMENTO	DATA ESAME	VOTO
900000	F1X002	Programmazione 1	2023-09-23	30
900001	F1X001	Architettura degli elaboratori 1	2023-09-11	20
900001	F1X002	Programmazione 1	2023-09-23	29
900003	C2X001	Istituzioni di matematica	2023-09-14	21

Figura 5: Carriera valida su un utente segreteria.

Carriera valida

Contiene i voti e le date più recenti di tutti gli esami superati

Esami registrati: 1 **Media dei voti:** 30

ID INSEGNAMENTO	NOME INSEGNAMENTO	DATA ESAME	VOTO
F1X002	Programmazione 1	2023-09-23	30

Figura 6: Carriera valida sull'utente studente con matricola 900000.

Informazioni del corso di laurea

Implementata mediante una vista (non materializzata), la produzione delle informazioni dei corsi di laurea è accessibile completamente da un qualsiasi utente segreteria e accessibile in maniera parziale da un utente studente (solamente per quanto riguarda i corsi di laurea attivati).

E' stata effettuata una selezione dalla tabella *insegnamento* con un inner join con la tabella *docente*, ordinate in modo crescente sul codice *corso_di_laurea* e il *nome* dell'insegnamento e proiettando su *corso_di_laurea*, *id*, *nome* dell'insegnamento, *descrizione_testuale*, una colonna creata concatenando i valori degli attributi *nome* e *cognome* della tabella *docente*, *anno_previsto*, *semestre*, *ore_totali* e *lingua*.

Prova:

Corso di laurea

Codice: F1X
Nome: Informatica
Tipo: Triennale
Anno accademico offerta: 2022-2023
Facoltà: Scienze e Tecnologie
Sede: Città Studi
Accesso: Programmato
Descrizione: Gli obiettivi del corso di laurea in Informatica sono: da una parte fornire una solida conoscenza di base e metodologica dei principali settori delle scienze informatiche e matematiche e dall'altra fornire una buona padronanza delle metodologie e tecnologie proprie dell'Informatica, offrendo una preparazione adeguata per imparare e conoscere i diversi ambiti applicativi della disciplina e poter assimilare, comprendere e valutare l'impatto dei costanti progressi scientifici e tecnologici nell'ambito della disciplina.

Insegnamenti del corso di laurea

ID INSEGNAMENTO	NOME	DESCRIZIONE TESTUALE	DOCENTE	ANNO PREVISTO	SEMESTRE	ORE TOTALI	LINGUA
F1X001	Architettura degli elaboratori 1	L'insegnamento introduce le conoscenze dei principi che sottendono al funzionamento di un elaboratore digitale; partendo dal livello delle porte logiche si arriva, attraverso alcuni livelli di astrazione intermedi, alla progettazione di ALU firmware e di un'architettura MIPS in grado di eseguire il nucleo delle istruzioni in linguaggio macchina.	Luigi Palermo	1	1	60	Italiano
F1X002	Programmazione 1	Obiettivo dell'insegnamento e' introdurre gli studenti alla programmazione imperativa strutturata e al problem solving in piccolo.	Paolo Beneventi	1	1	120	Italiano
F1X003	Matematica del continuo	L'obiettivo dell'insegnamento è duplice. Anzitutto, fornire agli studenti un linguaggio matematico di base, che li metta grado di formulare correttamente un problema e di comprendere un problema formulato da altri. Inoltre, fornire gli strumenti matematici indispensabili per la soluzione di alcuni problemi specifici, che spaziano dal comportamento delle successioni a quello delle serie e delle funzioni di una variabile.	Paolo Beneventi	1	1	112	Italiano
F1X004	Algoritmi e Strutture Dati	L'obiettivo dell'insegnamento è duplice. Anzitutto, fornire agli studenti un linguaggio matematico di base, che li metta grado di formulare correttamente un problema e di comprendere un problema formulato da altri. Inoltre, fornire gli strumenti matematici indispensabili per la soluzione di alcuni problemi specifici, che spaziano dal comportamento delle successioni a quello delle serie e delle funzioni di una variabile.	Paolo Beneventi	2	1	112	Italiano

Propedeuticità

INSEGNAMENTO 1	INSEGNAMENTO 2
Matematica del continuo	Algoritmi e Strutture Dati
Programmazione 1	Algoritmi e Strutture Dati

Attenzione: L'insegnamento 1 è propedeutico all'insegnamento 2.

Figura 7: Informazioni di un corso di laurea.

2.3.2 Funzioni

Verbalizzazione

Implementata come una funzione in *PLPGSQL*.

Verifica se uno studente è iscritto ad un determinato esame, se lo è e non era assente il giorno dell'esame, verbalizza il voto passato in input, inserendo una riga nella tabella *carriera* e cancella la sua iscrizione, altrimenti se lo studente era assente fa solamente la cancellazione dell'iscrizione; se non è iscritto all'esame ritorna un messaggio di errore.

Prova:

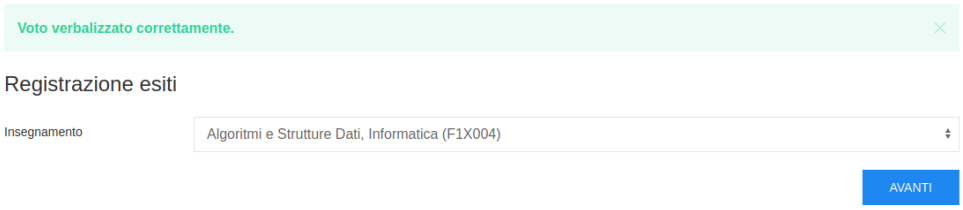

Figura 8: Verbalizzazione eseguita correttamente.

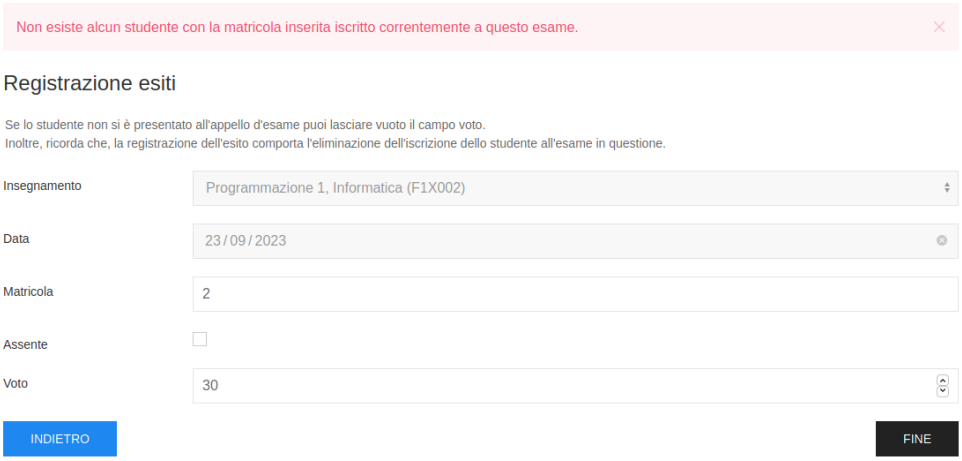

Figura 9: Errore nella verbalizzazione.

2.3.3 Trigger

Mantenimento delle informazioni e delle carriere di studenti rimossi

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'eliminazione sulla tabella *studente_corrente* viene inserita una copia della riga che si sta eliminando nella tabella *storico_studenti* insieme ad un motivo NULL (perché non noto a priori, però verrà aggiornato successivamente) e l'anno di rimozione uguale all'anno corrente. Viene anche copiata la carriera completa dello studente nella tabella *storico_carriera*, dopodiché viene eliminata la sua carriera completa dalla tabella *carriera*, e infine si ritorna all'esecuzione della *DELETE*.

Prova:

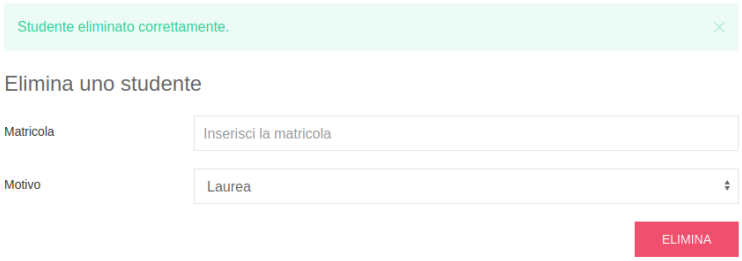

Figura 10: Eliminazione dello studente con matricola 900001.

Storico studenti

MATRICOLA	USERNAME	NOME	COGNOME	SESSO	DATA DI NASCITA	CORSO DI LAUREA	MOTIVO	ANNO RIMOZIONE
900001	pietro.lucchese1	Pietro	Lucchese	M	2002-10-17	F1X	Rinuncia	2023
900002	vincenzo.colombo1	Vincenzo	Colombo	M	1999-10-17	C1X	Rinuncia	2020
900003	gaetana.cocci1	Gaetana	Cocci	F	2003-01-05	C2X	Rinuncia	2022

Figura 11: Visualizzazione in *storico_studenti* dello studente eliminato in precedenza.

Correttezza delle iscrizioni agli esami

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *iscrizione*, controlla che si effettui un iscrizione ad un esame di un insegnamento del corso di laurea a cui appartiene lo studente che si iscrive, e che lo studente che si iscrive rispetti le propedeuticità per svolgere l'esame, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova:

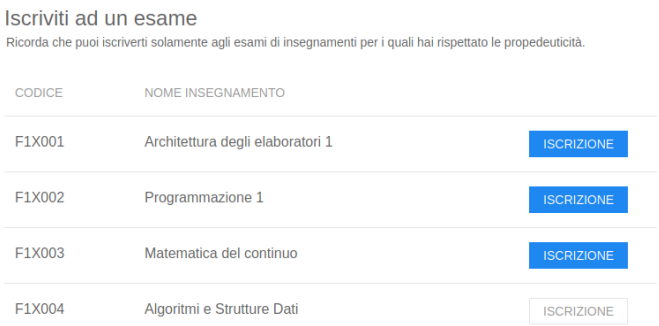

Figura 12: Studente di Informatica non può iscriversi ad Algoritmi se non ha superato prima Programmazione 1 e Matematica del Continuo.

Correttezza del calendario d'esame

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *esame*, controlla che si crei un esame che non sia programmato nello stesso giorno di un altro esame di un insegnamento previsto per lo stesso anno nella stessa offerta di un corso di laurea, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova:

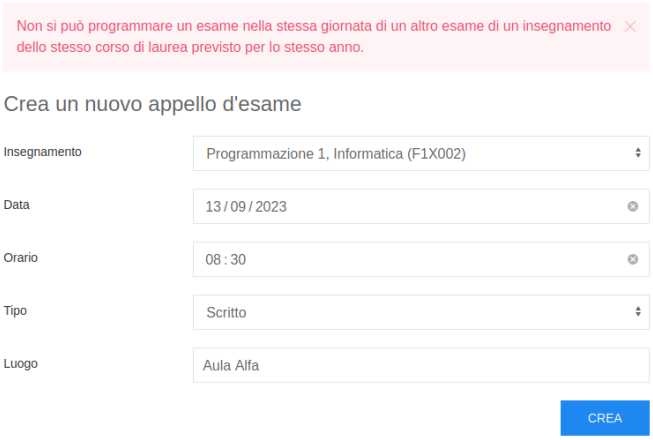

Figura 13: Errore nella creazione di un esame che non compie la condizione.

Correttezza dell'anno previsto per gli insegnamenti

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento o aggiornamento sulla tabella *insegnamento*, controlla che se l'insegnamento appartiene a un corso di laurea magistrale, l'anno previsto sia minore di 3 e maggiore di 0, nel caso di un insegnamento in un corso di laurea triennale, verifica se l'anno previsto è minore di 4 e maggiore di 0, se compie con la condizione la riga, si inserisce o aggiorna, altrimenti si evita l'inserimento o l'aggiornamento.

Prova: Nell'applicazione web non è possibile scegliere un anno previsto incoerente per un insegnamento in un corso di laurea.

Correttezza del profilo utente per gli studenti ed ex-studenti

Implementato come due trigger che eseguono ognuno una funzione in *PLPG-SQL*.

Prima dell'inserimento sulla tabella *studente_corrente*, controlla che il ruolo del profilo utente associato sia coerente e che non sia già stato preso da un ex-studente, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

L'altro trigger funziona in modo simile ma sulla tabella *storico_studenti*, invece di controllare prima dell'inserimento, lascia inserire e controlla dopo l'inserimento, se la condizione non si verifica, allora si elimina il record inserito, altrimenti non si fa nulla. Questa scelta è stata presa per fare in modo che il trigger che sposta le informazioni e la carriera di uno studente che si sta eliminando, non attivi questo trigger e provochi problemi di correttezza nei trigger, infatti questo trigger è specificamente un CONSTRAINT TRIGGER con l'opzione DEFERRABLE INITIALLY DEFERRED per spostare la sua esecuzione alla fine della transazione in cui si esegue il trigger del mantenimento delle informazioni degli ex-studenti (auto-commit).

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza della matricola per gli studenti ed ex-studenti

Implementato come due trigger che eseguono ognuno una funzione in *PLPG-SQL*. Funzionano in modo analogo ai trigger precedenti.

Prima dell'inserimento, controlla se la matricola non sia già stata presa da un ex-studente, se compie la condizione si inserisce la riga, altrimenti si evita l'inserimento.

In modo simile, dopo l'inserimento, si controlla che la matricola inserita su *storico_studenti* non sia la matricola di uno studente corrente, poiché creerebbe un inconsistenza dei dati, se la condizione si compie, non si fa nulla, altrimenti la riga viene rimossa. Anche questo trigger è un CONSTRAINT TRIGGER con l'opzione DEFERRABLE INITIALLY DEFERRED per gli stessi motivi descritti nella coppia di trigger precedente.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza del profilo utente per i docenti

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *docente*, controlla che il ruolo del profilo utente associato sia coerente, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza della data di un esame creato

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *esame*, controlla che la data in cui si vuole programmare l'esame sia successiva al giorno odierno, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova:

Figura 14: Errore nella creazione di un esame che non compie la condizione.

Correttezza delle propedeuticità

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *propedeuticita*, controlla che gli insegnamenti su cui si vuole stabile la propedeuticità appartengano alla stessa offerta dello stesso corso di laurea e inoltre che i suddetti insegnamenti abbiano anno previsto e semestre coerenti, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova:

Figura 15: Errore nella creazione di una propedeuticità che non compie la condizione.

Correttezza tra studenti e corsi di laurea

Implementato come due trigger che eseguono ognuno una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *studente_corrente*, controlla che il corso di laurea a cui si vuole iscrivere lo studente sia attivato, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

L'altro trigger funziona identicamente ma sulla tabella *storico_studenti*.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza nell'eliminazione dei docenti

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'eliminazione sulla tabella *docente*, controlla che il docente non sia responsabile di nessun insegnamento, se compie con la condizione la riga, si elimina, altrimenti si evita l'eliminazione.

Prova:

Figura 16: Errore nell'eliminazione di un docente che non compie la condizione.

Correttezza nell'eliminazione dei corsi di laurea

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'eliminazione sulla tabella *corso_di_laurea*, controlla che il corso di laurea non sia già stato attivato, se compie con la condizione la riga, si elimina, altrimenti si evita l'eliminazione.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza nell'eliminazione degli insegnamenti

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'eliminazione sulla tabella *insegnamento*, controlla che l'insegnamento non appartenga ad un corso di laurea già attivato, se compie con la condizione la riga, si elimina, altrimenti si evita l'eliminazione.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza nell'eliminazione degli esami

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'eliminazione sulla tabella *esame*, controlla che non sia stato verbalizzato nessun voto sull'esame, se compie con la condizione la riga, si elimina, altrimenti si evita l'eliminazione.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza nella creazione di esami

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *esame*, controlla che l'insegnamento associato all'esame non sia appartenente ad un corso di laurea non attivato, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza nella verbalizzazione

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *iscrizione*, controlla che lo studente non abbia già verbalizzato un voto per quello specifico esame (stessa data), se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Correttezza dell'attributo iscrizioni_aperte su iscrizioni ad esami

Implementato come un trigger che esegue una funzione in *PLPGSQL*.

Prima dell'inserimento sulla tabella *iscrizione*, controlla che l'esame a cui si iscrive lo studente abbia le iscrizioni aperte, se compie con la condizione la riga, si inserisce, altrimenti si evita l'inserimento.

Prova: Nell'applicazione web non è possibile che si verifichi la condizione presentata precedentemente.

Nota: Sono stati implementati altri trigger di descrizione e funzionamento triviale, per verificare ad esempio, la correttezza delle cardinalità dello schema ER ristrutturato.