Basi di Dati

Progetto “Book Exchange”

Oscar Elia Conti 1071039

Marco Zanella 1074420

Sommario

1.Abstract

L'istruzione italiana è suddivisa in vari gradi. Di questi l'università è il più elevato ed ha come obiettivi la promozione della ricerca e dell'istruzione di livello superiore. Esistono varie tipologie di università ognuna delle quali è suddivisa in scuole, a loro volta specializzate in vari corsi. Per ogni corso di laurea sono previsti diversi insegnamenti o corsi, che variano a seconda dell'università e del corso di laurea stesso.
Anche a causa dell'alto grado di istruzione, il costo dei libri, siano essi manuali o testi didattici, assume un peso rilevante nelle spese che ciascuno studente deve affrontare.
Per questo motivo la compravendita di libri usati è una pratica piuttosto comune. Solitamente avviene tra studenti di anni differenti, i quali, dopo aver completato un corso, decidono di rivendere il testo utilizzato agli studenti che lo affronteranno dopo di loro. L'acquisto di questi libri, tuttavia, non sempre avviene tra studenti dello stesso corso di laurea, ma può avvenire anche tra studenti di corsi diversi o università diverse e, anche se meno abitualmente, tra persone esterne all'ambiente universitario. Inoltre esistono diversi negozi, specializzati nella trattazione di libri usati, i quali acquistano i vecchi testi usati da studenti per poi rivenderli.

Sulla base di questa analisi nasce il progetto “BooX” (“Book eXchange”), il cui obiettivo è quello di creare una piattaforma univoca e di facile accesso per la gestione della compravendita di libri usati.

2.Descrizione dei requisiti

2.1 Analisi dei requisiti

Si vuole realizzare una base di dati per la gestione della compravendita di libri usati in ambito universitario.

Lo scopo della base di dati è quello di gestire una serie di annunci online, i quali possono essere:

- annunci di vendita
- annunci di ricerca

Per *annuncio di vendita* si intende la pubblicazione da parte di un utente di un inserzione riguardante la vendita di un libro (in possesso dell'utente che ha creato l'inserzione), dotato anche di un prezzo di vendita. Un esempio di annuncio di vendita potrebbe essere: *"vendo libro 'Basi di dati-modelli e linguaggi di interrogazione' di Paolo Atzeni, Stefano Ceri, Piero Fraternali, Stefano Tiraboschi e Riccardo Tolone, editore McGrow-Hill, in buono stato. Prezzo €20,00*".

Un *annuncio di ricerca* sarà, invece, un'inserzione creata da un utente che vuole acquistare un libro che non riesce a reperire. Un esempio potrebbe essere:"*cerco libro 'Basi di dati-modelli e linguaggi di interrogazione' di Paolo Atzeni, Stefano Ceri, Piero Fraternali, Stefano Tiraboschi e Riccardo Tolone, editore McGrow-Hill*".

Nel caso in cui fosse già presente un annuncio di vendita per il libro cercato sarà lo stesso possibile per l'utente inserire un annuncio di ricerca per lo stesso libro. In modo analogo viene gestita la situazione opposta.

Gli annunci si dividono ulteriormente in:

- annunci in bacheca
- annunci non in evidenza, i quali possono essere:
  - annunci scaduti
  - annunci ritirati

Gli *annunci in bacheca* sono le pubblicazioni visualizzate nel momento in cui un utente cerca un determinato libro. Queste diventano *annunci non in evidenza* nel momento in cui scadono, oppure vengono ritirate dall'utente stesso che le ha pubblicate (in caso di vendita, acquisto libro o motivi personali). Ogni annuncio rimane in bacheca per un periodo di tempo pari a 6 mesi dalla data di pubblicazione.

Gli annunci vengono postati da un singolo utente, mentre un utente, ovviamente, può postare più annunci, sia di vendita che di ricerca.

Ogni *utente* è identificato attraverso uno username(e-mail) che lo contraddistingue, a

cui è associata una password. Queste due proprietà permettono ad un utente di effettuare il login e postare nuovi annunci. Ciascun utente verrà poi caratterizzato da nome, cognome e data di nascita.
Gli utenti, tra loro, possono scambiarsi messaggi per chiedere ulteriori informazioni sull'annuncio e per concordare un eventuale luogo di incontro per lo scambio.
Un *messaggio* è identificato dal momento in cui è stato inviato e dagli utenti che si scambiano il messaggio.
Dal momento che questa piattaforma si indirizza ad un pubblico prettamente di studenti universitari e, con il fine di aiutare la ricerca dei testi, si vuole tener traccia del *corso di laurea* frequentato da tale utente, con annessi *insegnamenti* e *università*.
I *libri*, oggetto dell'inserzione, sono identificati attraverso il codice ISBN, univoco per ogni libro. Al libro vengono associati titolo, anno di pubblicazione e prezzo(di copertina). Ad ogni libro può essere associata una *descrizione,* uno o più *autori* che l'hanno scritto e l'*editore* che ha pubblicato tale testo. Ogni libro, memorizzato nel banca di dati, è associato ad uno o più insegnamenti che lo adottano, ed anche ad uno più annunci di compravendita.
Gli insegnamenti sono i singoli corsi facenti parte di un corso di laurea e, ad ognuno di questi, è associato l'anno in cui viene tenuto. Ciascun corso di laurea è collegato all'università in cui viene erogato. Di questi si vuole tener traccia solamente del nome che li identifica. Per l'università, oltre al nome, si vuole avere informazioni sull'indirizzo della sede centrale e di un numero telefonico.

2.1.1 Vincoli e semplificazioni

Per il progetto in questione è presa in esame una versione semplificata della realtà di interesse. Per questo sono state fatte delle assunzioni per restringere il campo di sviluppo della base di dati.

1. I corsi di laurea sono considerati differenti a seconda dell'università. *Per esempio il corso di laurea in informatica all'università di Padova sarà considerato un oggetto totalmente differente rispetto il corso di laurea di informatica tenuto nell'ateneo di Trento.*
2. Non è tenuta traccia della divisione interna dell'università: nella base di dati non esiste la classificazione dei corsi di laurea in indirizzi o/e scuole.
3. Ogni università è caratterizzata dal nome che la identifica univocamente.
4. Non viene tenuto conto delle possibili modifiche che i corsi di laurea possono subire, come gli insegnamenti, che, per ogni corso che possono variare tra anni differenti.
5. È possibile inserire un annuncio di ricerca di un determinato libro anche se è presente un annuncio di vendita riguardante lo stesso libro. Allo stesso modo è possibile inserire un annuncio di vendita che riguarda lo stesso testo di un annuncio di ricerca. Non sarà possibile, invece, per un utente avere in bacheca contemporaneamente un annuncio di vendita e uno di ricerca per lo stesso libro.
6. Ogni utente può frequentare al massimo un corso di laurea.
7. La descrizione di un determinato libro è trattata come entità a sé stante poiché è vista come oggetto potenzialmente di grande dimensioni, contenente la copertina e un testo descrittivo riguardante il libro che potrebbe anche essere molto lungo.
8. I messaggi sono considerati all'interno del database per completezza della stessa e poiché significativi in una possibile implementazione reale ma, per semplicità, non verranno implementate funzionalità riguardanti gli stessi. (*NB: Per questo motivo non verranno successivamente implementate altre funzionalità come query/funzioni/trigger nella parte riguardante MySQL del progetto, né a livello di interfaccia web*)

2.1.2 Glossario

Termine	Descrizione	Sinonimi	Collegamenti
Annuncio	Inserzione riguardante la compravendita di un libro, creata da un utente	Inserzione, Pubblicazione	Utente, Libro
Annuncio in bacheca	Annunci visibili ai visitatori del sito	Annuncio in evidenza	Annuncio, Utente, Libro
Annuncio non in evidenza	Annunci non più visibili agli utenti	Annuncio non in bacheca	Annuncio, Utente, Libro
Annuncio scaduto	Annuncio non in evidenza poiché scaduto il termine massimo di esposizione (6 mesi)		Annuncio, Utente, Libro
Annuncio ritirato	Annuncio non in bacheca perché rimosso dall'utente che l'ha pubblicato	Annuncio non scaduto	Annuncio, Utente, Libro
Annuncio di vendita	Inserzione per la vendita di un determinato libro		Annuncio, Utente, Libro
Annuncio di ricerca	Inserzione per la ricerca di un determinato libro		Annuncio, Utente, Libro
Utente	Persona, che può essere uno studente universitario, che può pubblicare annunci e scambiarsi messaggi con altri utenti	Utilizzatore del database, Account	Annuncio, Messaggio, Corso di laurea
Messaggio	Messaggio che viene scambiato tra utenti		Utente
Libro	Libro utilizzato in un determinato insegnamento	Libro universitario, testo universitario	Annuncio, Descrizione, Autore, Editore, Insegnamento
Descrizione	Descrizione o del libro		Libro
Autore	Autore di uno o più libri		Libro
Editore	Editore di uno o più libri		Libro
Insegnamento	Insegnamento appartenente ad un corso di laurea	Corso	Corso di laurea, Libro
Corso di laurea	Insieme degli insegnamenti per conseguire una determinata laurea		Utente, Insegnamento
Università	Ente che eroga i corsi di laurea(rappresenta un'università italiana)		Corso di laurea

2.2 Operazioni

Le operazioni tipiche previste per il database sono:

- Iscrizione:

  Questa operazione inserisce un nuovo utente nella base di dati con tutti i suoi dati(nel caso sia studente anche i dati riguardanti l'università che frequenta).

  Operazione eseguita da una persona che vuole diventare un utente.

- Pubblicazione annuncio vendita:

  Inserisce un nuovo annuncio tra gli annunci di vendita collegato ad un utente.

  Operazione eseguita dagli utenti.

- Pubblicazione annuncio cerca:

  Inserisce un nuovo annuncio tra gli annunci di ricerca collegato ad un utente.

  Operazione eseguita dagli utenti.

- Rimozione annuncio non scaduto:

  Cancellazione di un annuncio tra quelli in bacheca da parte dell'utente che lo ha pubblicato e inserimento tra gli annunci rimossi.
  Operazione eseguita dagli utenti.

- Rimozione annuncio scaduto:

  Cancellazione di un annuncio in bacheca, il quale ha superato il termine massimo di esposizione, e inserimento tra gli annunci scaduti.
  Operazione eseguita in automatico dalla base di dati.

- Messaggio:

  Scambio di messaggi tra utente mittente e utente destinatario.
  Operazione eseguita dagli utenti.

- Ricerca annuncio:

  Ricerca di un libro tra gli annunci presenti in bacheca da parte di un utente o persona non iscritta

  Operazione eseguita dagli utenti.

- Inserimento libro non presente nel database:

  Inserimento di un nuovo libro con relativo autore, editore e eventuale descrizione.

  Operazione eseguita dagli utenti.

- Rimozione account:

  Rimozione di un utente per decisione dell'utente stesso. Operazione eseguita dagli utenti.
- Login:

  Controllo che l'email e la password inserita da parte di un visitatore corrispondano ad un account presente nel database. Operazione eseguita dagli utenti.
- Modifica dati:

  Modifica da parte di un utente dei dati personali ad esso associati. Operazione eseguita dagli utenti.
- Inserimento università:

  Inserimento di nuove università non presenti nel database. Operazione eseguita dagli amministratori.
- Inserimento corsi di laurea:

  Inserimento di corsi di laurea non presenti nel database. Operazione eseguita dagli amministratori.
- Visualizzazione annunci:

  Visualizzazione di tutti gli annunci in bacheca.

  Operazione eseguita utenti, persone non iscritte e amministratori.
- Visualizzazione messaggi:

  Visualizzazione di tutti i messaggi inviati e ricevuti da parte di un utente.

  Operazione eseguita da utenti e amministratori.
- Visualizzazione di annunci per utente:

  Visualizzazione da parte di un utente degli annunci in bacheca ad esso associati, degli annunci scaduti e degli annunci rimossi. Operazione eseguita dagli utenti.
- Visualizzazione dati:

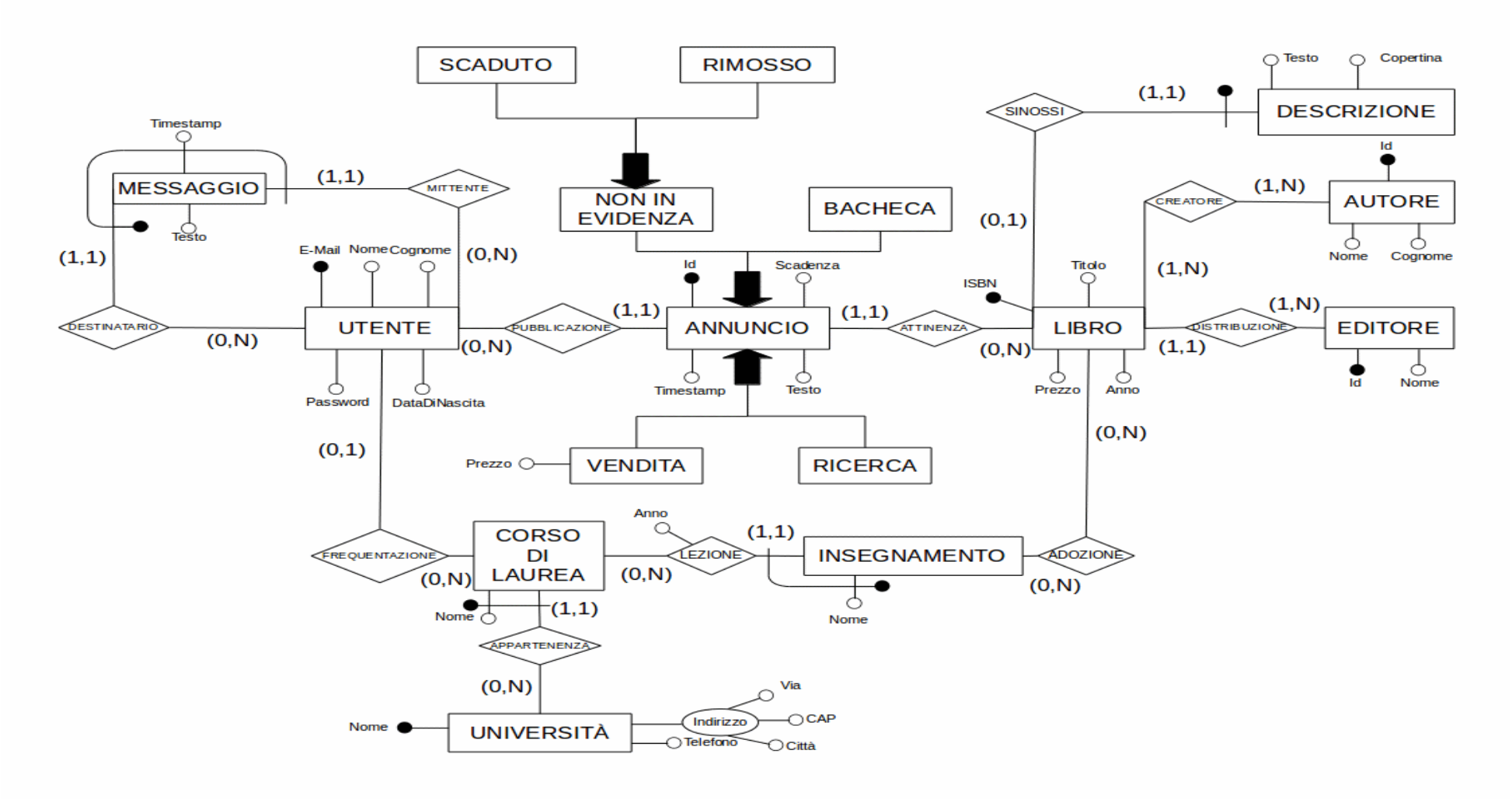
  Visualizzazione da parte di un utente dei dati presenti nel suo profilo. Operazione eseguita dagli utenti.

2.2.1 Tavola delle operazioni

#	Descrizione	Tipo	Esecutori
1	Iscrizione	Interattiva	Non iscritti
2	Pubblicazione annuncio vendita	Interattiva	Utente
3	Pubblicazione annuncio ricerca	Interattiva	Utente
4	Rimozione annuncio non scaduto	Interattiva	Utente
5	Rimozione annuncio scaduto	Batch	Database
6	Scambio messaggi	Interattiva	Utente
7	Ricerca annuncio	Interattiva	Utente
8	Inserimento libro	Interattiva	Amministratore
9	Rimozione libro	Interattiva	Amministratore
10	Login	Interattiva	Utente
11	Modifica dati	Interattiva	Utente
12	Inserimento università	Interattiva	Amministratore
13	Inserimento corso di laurea	Interattiva	Amministratore
14	Visualizzazione annunci	Interattiva	Tutti
15	Visualizzazione messaggi	Interattiva	Utente
16	Visualizzazione annunci per utente	Interattiva	Utente
17	Visualizzazione dati	Interattiva	Utente

3 Progettazione concettuale

3.1 Schema ER



3.2 Descrizione entità

Utente: entità che rappresenta gli account del sito

- *E-Mail* tipo: stringa, identificatore dell'entità
- *Password* tipo: stringa
- *Nome* tipo: stringa
- *Cognome* tipo: stringa
- *DataDiNascita* tipo: date

Messaggio: entità che rappresenta i messaggi scambiati tra gli utenti

- *Timestamp* tipo: date&time, data e ora invio messaggio
- *Testo* tipo: stringa, corpo del messaggio

Entità identificata esternamente dall'utente mittente del messaggio, utente destinatario e dal *Timestamp* di invio.

Annuncio: entità che rappresenta un annuncio pubblicato nel sito

- *Id* tipo: integer, identificatore univoco delle istanze dell'entità
- *Timestamp* tipo: date&time, data e ora pubblicazione
- *Scadenza* tipo:d ate, data in cui l'annuncio sarà rimosso dalla bacheca
- *Testo* tipo: stringa, descrizione dell'annuncio

Vendita: generalizzazione di a*nnuncio* che rappresenta un annuncio di vendita di un libro

- *Prezzo* tipo: decimal(5,2), prezzo di vendita del libro nell'annuncio

Ricerca: generalizzazione di a*nnuncio* che rappresenta un annuncio per la ricerca di un libro.

Bacheca: generalizzazione di a*nnuncio* che rappresenta l'insieme degli annunci non ancora scaduti e non rimossi dell'utente.

Non_In_Evidenza: generalizzazione di *annuncio* che rappresenta l'insieme degli annunci non ricercabili.

Scaduto: generalizzazione di *Non_In_Evidenza* che rappresenta l'insieme degli annunci scaduti.

Rimosso: generalizzazione di *Non_In_Evidenza* che rappresenta l'insieme degli annunci rimossi dall'utente.

Libro: entità che rappresenta un libro nel database

- *ISBN* tipo: bigint(13 numeri), identificatore di libro
- *Titolo* tipo: stringa
- *Prezzo* tipo: decimal(5,2), prezzo di copertina del libro
- *Anno* tipo: date, anno di pubblicazione

Descrizione: entità che rappresenta la descrizione di un certo libro

- *Testo* tipo: TEXT
- *Copertina* tipo: BLOB, copertina del libro

Entità identificata esternamente dal libro a cui è associata.

Autore: entità che rappresenta gli autori dei libri presenti nel database

- *Id* tipo: integer, identificatore
- *Nome* tipo: stringa
- *Cognome* tipo: stringa

Editore: entità che rappresenta gli editori dei libri presenti nel database

- *Id* tipo: integer, identificatore
- *Nome* tipo: stringa

Corso_Di_Laurea: entità che rappresenta un corso di laurea

- *Nome* tipo: stringa

Università: entità che rappresenta un'università italiana

- *Nome* tipo: stringa
- *Indirizzo* attributo composto da:
  - *Via* tipo: stringa
  - *CAP* tipo: decimal(5)
  - *Città* tipo: stringa
- *Telefono* tipo: stringa

3.3 Descrizione delle relazioni

Destinatario: relazione che associa ad un *messaggio* l'*utente* che riceve il messaggio

- Molteplicità: (1,N)
- Totalità: (1,0)
- Entità coinvolte: Messaggio e Utente

Un messaggio ha un unico destinatario, che deve essere sempre presente, mente un utente potrà essere destinatario di più messaggi ma può anche non averne ricevuti.

Mittente: relazione che associa ad un *messaggio* l'*utente* che invia il messaggio

- Molteplicità: (1,N)
- Totalità: (1,0)
- Entità coinvolte: Messaggio e Utente

Un messaggio ha un unico mittente, che deve essere sempre presente, mentre un utente potrà essere mittente di più messaggi, ma può anche non averne inviati.

Pubblicazione: relazione che associa un *utente* agli *annunci* pubblicati

- Molteplicità: (N,1)
- Totalità: (0,1)
- Entità coinvolte: Utente e Annuncio

Un annuncio è pubblicato da un unico utente che deve essere sempre presente. Un utente può pubblicare più annunci ma può non averne pubblicati.

Attinenza: relazione che associa un *annuncio* al *libro* a cui si riferisce quel determinato annuncio

- Molteplicità: (1,N)
- Totalità: (0,1)
- Entità coinvolte: Libro e Annuncio

Un annuncio riguarda un unico libro che deve essere sempre presente. Per un determinato libro possono essere presenti più annunci ma possono non essercene.

Sinossi: relazione che associa un *libro* alla propria *descrizione*

- Molteplicità: (1,1)
- Totalità: (0,1)
- Entità coinvolte: Libro e Descrizione

Una descrizione riguarda un unico libro, mentre un libro può avere un'unica descrizione ma può non essere presente.

Creatore: relazione che associa un *libro* all'*autore* che l'ha scritto

- Molteplicità: (N,N)
- Totalità: (1,1)
- Entità coinvolte: Autore e Libro

Un libro è scritto da uno o più autori. Un autore presente nel database può scrivere uno o più libri presenti nel database.

Distribuzione: relazione che associa un *libro* all'*editore* che l'ha pubblicato

- Molteplicità: (1,N)
- Totalità: (1,1)
- Entità coinvolte: Libro e Editore

Un libro ha un unico editore che deve essere sempre presente. Un editore presente nel database può distribuire uno o più libri presenti nel database.

Frequentazione: relazione che associa un *utente* al *corso di laurea* che frequenta

- Molteplicità: (1,N)
- Totalità: (0,1)
- Entità coinvolte: Utente e Corso_Di_Laurea

Un utente può frequentare al massimo un corso di laurea. Un corso di laurea può essere frequentato da più utenti presenti nel database ma possono non essercene.

Lezione: relazione che associa un *insegnamento* al *corso di laurea* in cui è previsto

- Molteplicità: (1,N)
- Totalità: (1,0)
- Entità coinvolte: Insegnamento e Corso_Di_Laurea
- Attributo *Anno:* specifica in che anno di un corso di laurea viene erogato un certo insegnamento

Un insegnamento fa parte di un determinato corso di laurea che deve essere presente. Per un corso di laurea possono essere presenti uno o più insegnamenti ma possono non essercene.

Adozione: relazione che associa un *libro* agli *insegnamenti* in cui viene utilizzato

- Molteplicità: (N,N)
- Totalità: (0,0)
- Entità coinvolte: Libro e Insegnamento

Un libro può essere adottato da più insegnamenti ma può non essere adottato da alcun insegnamento presente nel database. Un insegnamento può adottare più libri, ma può non prevederne.

Appartenenza: relazione che associa un *corso di laurea* all'*università* a cui appartiene

- Molteplicità: (1,N)
- Totalità: (1,0)
- Entità coinvolte: Corso_Di_Laurea e Università

Un corso di laurea presente nel database deve essere associato ad una unica università. Per ogni università presente nel database deve essere presente almeno un corso di laurea.

3.4 Descrizione generalizzazioni

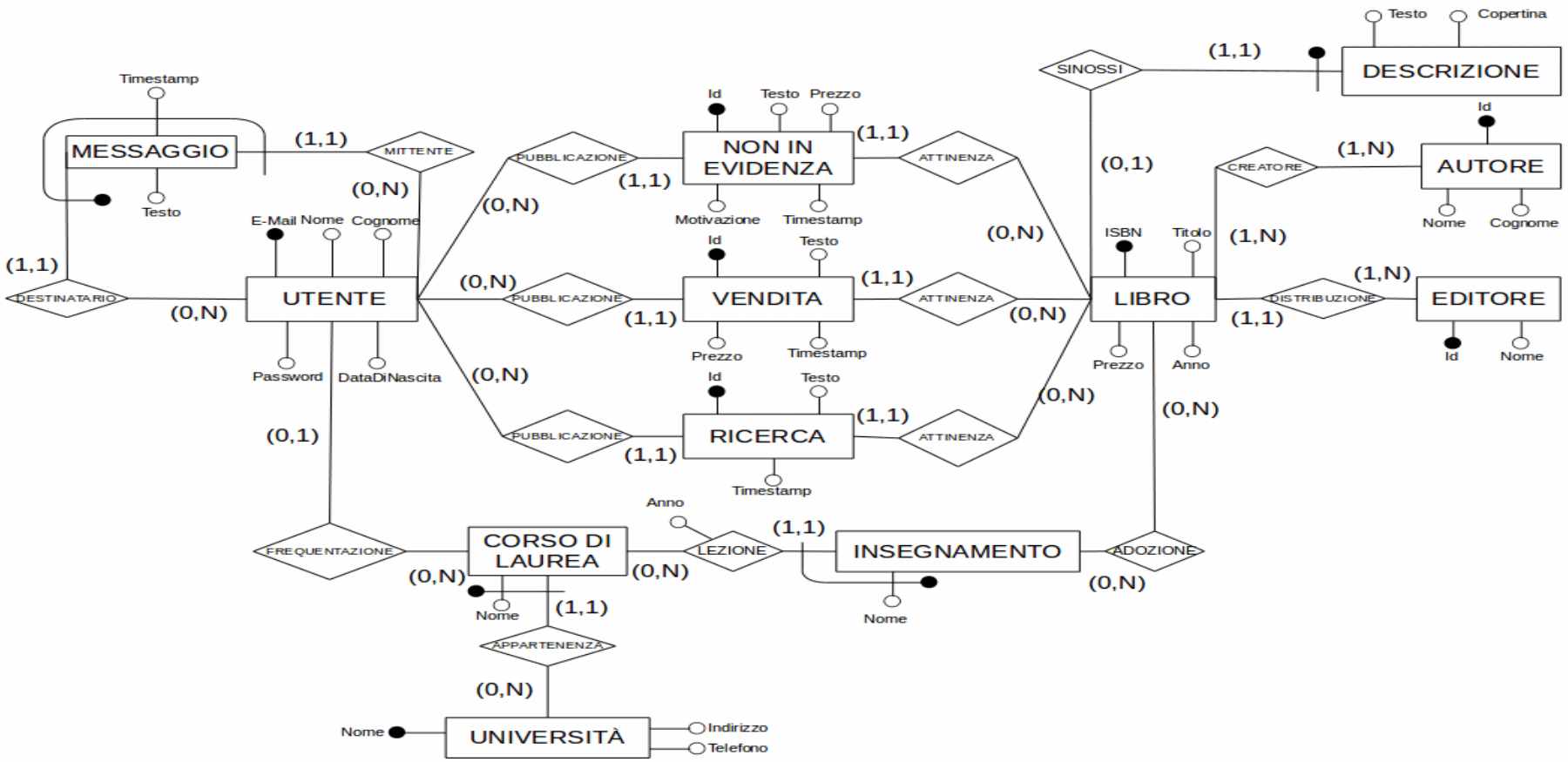
Sono presenti due gerarchie con l'entità annuncio al livello più alto.
La prima gerarchia prevede la divisione delle inserzioni in annunci di vendita e annunci di ricerca. Questa generalizzazione è totale poiché gli annunci gestiti dal database sono solamente di questi due tipi e, inoltre, c'è una distinzione tra queste entità figlie. Infatti l'entità *Vendita* deve prevedere un campo prezzo che sarà il prezzo di vendita del libro, attributo che non ha senso negli annunci di ricerca.

La seconda gerarchia prevede al primo livello ancora una volta *Annuncio* come entità padre mentre *Bacheca* e *Non_In_Evidenza* come entità figlie. Quest'ultima rappresenta la classe degli annunci che non sono più ricercabili. *Bacheca*, invece, rappresenta la classe degli annunci ricercabili. Non_In_Evidenza è entità padre per un'ulteriore generalizzazione dove le figlie sono *Rimosso* e *Scaduto*. *Rimosso* rappresenta la classe degli annunci non scaduti rimossi dagli utenti, mentre *Scaduto* rappresenta la classe, appunto, degli annunci scaduti.

4 Progettazione logica

4.1 Ristrutturazione dello schema ER

4.1.1 Schema ER ristrutturato



4.1.2 Analisi delle ridondanze

L'attributo *Scadenza* dell'entità *Annuncio* genera una ridondanza poiché rappresenta un attributo derivabile dall'attributo *Timestamp*.

Infatti, se a quest'ultimo sommiamo la costante prevista in 6 mesi per la scadenza degli annunci, otteniamo esattamente il valore previsto per tale attributo.

Nel caso in cui questo attributo non fosse presente, ogni giorno, per ogni inserzione, bisognerebbe calcolarne la data di scadenza per decidere se toglierla dalla bacheca e aggiungerla tra gli annunci scaduti o meno.

Supponiamo di avere la seguente situazione:

Annunci Non in Evidenza: 10000
Annunci di Vendita: 3000
Annunci di Ricerca: 2000

Sia nel caso di attributo non ridondante che di attributo ridondante abbiamo un unico accesso ad entrambe le tabelle.

Per ogni annuncio, in caso di attributo ridondante, basterebbe leggere *Scadenza* e fare un confronto con la data corrente.

Nel caso in cui questo attributo non ci sia, il carico di lavoro rimane pressoché lo stesso con la differenza che, per ogni annuncio, sia esso di vendita o di ricerca, bisognerebbe accedere al campo *Scadenza*, sommare a quest'ultimo 6 mesi e confrontare con la data corrente.

Quindi mantenere questo attributo non ha nessuna convenienza. Anzi questo comporterebbe uno spreco di memoria che con quella tavola dei volumi ammonterebbe a 8*(2000+3000+10000)B circa 120kB, quasi irrisorio, ma inutile.

4.1.3 Eliminazione delle generalizzazioni

Prendiamo prima di tutto in considerazione la generalizzazione composta dalle entità *Non_In_Evidenza*, *Scaduto* e *Rimosso* (figura a).

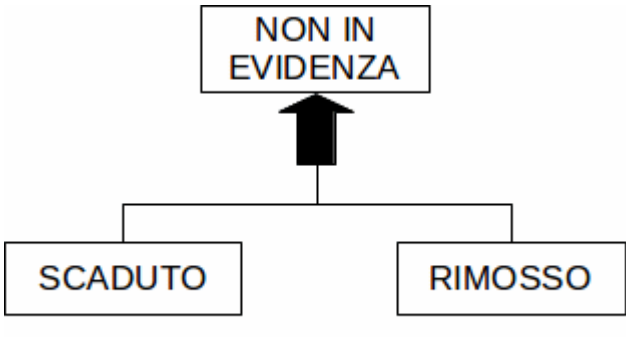


Figura a

Queste due ultime entità rappresentato rispettivamente gli annunci scaduti, quindi

non più in evidenza, e gli annunci rimossi dall'utente prima della scadenza degli stessi. L'unica distinzione tra le due è quindi la causa per cui un annuncio non si trova più tra gli annunci in evidenza. Poiché le principali operazioni presenti all'interno del database accedono ad entrambe le entità contemporaneamente, e non vi è distinzione di attributi tra il padre e le figlie, si è pensato quindi di accorpare le figlie nel padre, aggiungendo l'attributo *Motivazione* al padre, che discrimina tra attributi scaduti e attributi rimossi dall'utente creatore, ottenendo un'unica entità, *Non_In_Evidenza* (figura b).

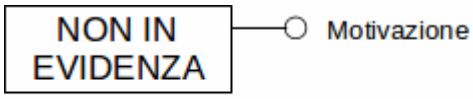


Figura b

La gerarchia così modificata sarà caratterizzata dalle entità *Annuncio*, *Bacheca* e *Non_In_Evidenza*.

Questa generalizzazione viene eliminata accorpando il padre, *Annuncio*, nelle figlie, poiché le principali operazioni effettuate sul database accedono separatamente alle due classi di annunci, ottenendo così due gerarchie (figura c).

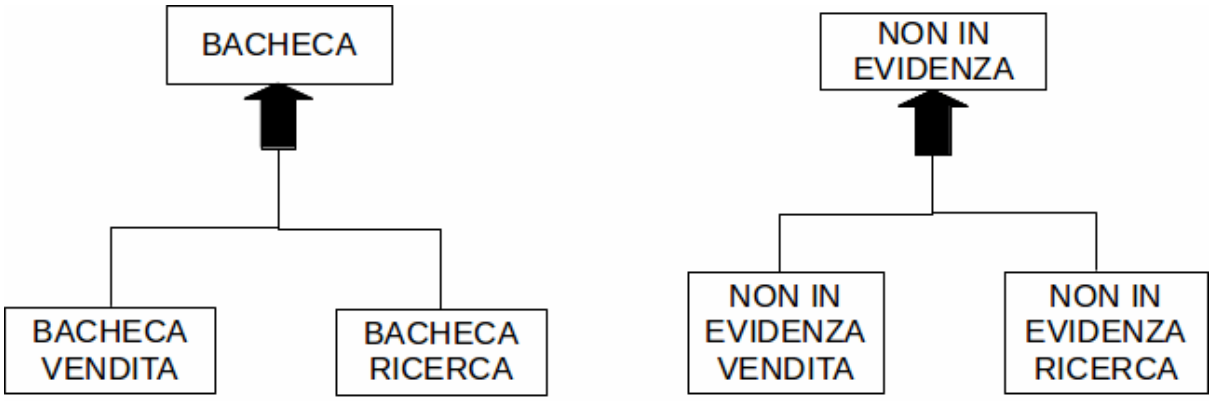


Figura c

Per queste due gerarchie vengono utilizzate tecniche di eliminazione differenti.

Prendiamo in considerazione la prima. Questa generalizzazione viene ristrutturata accorpando il padre nelle figlie. Questa decisione è motivata dal fatto che le operazioni principali sul database accedono separatamente alle entità figlie e non vi è distinzione tra gli attributi presenti nel padre da quelli presenti nelle figlie. Inoltre questa ristrutturazione permette di non sprecare memoria per l'attributo prezzo: nel caso in cui accorpassimo entrambe le entità nel padre, questo attributo, necessario in *BachecaVendita*, sarebbe sempre *NULL* per gli annunci di ricerca.

Per gli annunci non più in evidenza utilizziamo la strategia opposta poiché per questa categoria di annunci non si tiene conto della distinzione tra annuncio di ricerca ed annuncio di vendita, e quindi viene ristrutturata accorpando entrambe le entità in

Non_In_Evidenza. Vengono quindi aggiunti nel padre l'attributo *Tipo,* che specifica se l'annuncio era di ricerca o di vendita, e l'attributo *prezzo*, che sarà il prezzo di vendita nel caso di annuncio di vendita, *NULL* nel caso di annuncio di ricerca.
Alla fine quindi otteniamo tre entità che rinominiamo *Non_In_Evidenza*, *Vendita*(annunci di vendita in evidenza) e *Ricerca*(annunci di ricerca in evidenza).

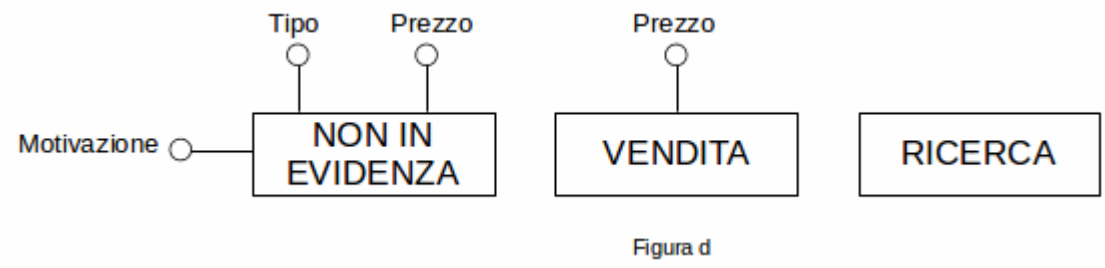


Figura d

4.1.4 Accorpamento/Partizionamento

Le entità *Libro* e *Descrizione* sono entità che potrebbero essere accorpate poiché vi è una relazione *"uno a uno"* e, inoltre, *Descrizione* è identificata esternamente proprio dal libro a cui si riferisce. Queste due classi di oggetti comunque si preferisce tenerle separate poiché *Descrizione* contiene campi di grandi dimensioni.

L'attributo composto *Indirizzo,* formato dagli attributi *Via*, *CAP* e *Città,* in *Università* viene sostituito con una stringa che rappresenta l'intero indirizzo, poiché gli attributi che formano *Indirizzo* vengono comunque acceduti come se fossero un unico campo, anche se questo comporta che ci sia la necessità di inserire in modo corretto questo campo.

4.1.5 Scelta degli identificatori principali

Per l'entità *Insegnamento*, identificata esternamente dalla relazione con l'entità *Corso_Di_Laurea,* a sua volta identificata tramite la relazione con *Università,* si è pensato di inserire un ID numerico, al fine di ridurre la dimensione della chiave primaria. L'entità *Descrizione* sarà identificata tramite la chiave del libro a cui si riferisce. Tutte le altre entità mantengono gli identificatori invariati.

4.2 Traduzione verso il modello relazionale

4.2.1 Associazioni uno a uno

Nello schema ristrutturato è presente un'unica associazione con molteplicità "*uno a uno*", ovvero la relazione tra *Descrizione* e *Libro*. Questa associazione viene tradotta inserendo la chiave di *Libro* nell'entità *Descrizione*.

4.2.2 Associazioni uno a molti

Le due associazioni "*uno a molti*" che sono presenti tra *Messaggio* e l'entità *Utente* vengono ristrutturate inserendo in ogni messaggio le chiavi corrispondenti all'utente mittente e destinatario del messaggio stesso.

L'associazione "*uno a molti*" presente tra *Utente* e *Corso_Di_Laurea* viene ristrutturata inserendo nella relazione corrispondente a *Utente* l'identificatore del corso di laurea a cui afferisce.

L'associazione "*uno a molti*" tra *Insegnamento* e *Corso_Di_Laurea* viene modificata nello schema relazionale inserendo nella relazione corrispondente a *Insegnamento* la chiave di *Corso_Di_Laurea*.

L'associazione "*uno a molti*" tra *Università* e *Corsi_Di_Laurea* viene tradotta inserendo nella relazione corrispondente a *Corso_Di_Laurea* la chiave di *Università*.

L'associazione "*uno a molti*" tra l'entità *Libro* e l'entità *Editore* viene ristrutturata inserendo nella relazione in cui viene tradotta l'entità *Libro* la chiave di *Editore*.

Per le associazioni "*uno a molti*" che coinvolgono le tre entità *Non_In_Evidenza*, *Vendita* e *Ricerca* verso l'entità *Utente*, viene inserita la chiave dell'utente proprietario dell'annuncio nelle relazioni corrispondenti alle entità sopra citate. In modo eguale vengono ristrutturate le associazioni che coinvolgono *Non_In_Evidenza*, *Vendita* e *Ricerca* e l'entità *Libro*.

4.2.3 Associazioni molti a molti

Per la ristrutturazione delle due associazioni "*molti a molti*" presenti rispettivamente tra *Insegnamento* e *Libro*(associazione *Adozione*) e *Autore* e *Libro*(tramite l'associazione *Creatore*) vengono inserite due relazioni: *AutoreLibro* e *LibroInsegnamento*.

La relazione *AutoreLibro* conterrà le chiavi delle relazioni in cui verranno tradotte *Autore* e *Libro*, per associare ogni autore ai libri che ha scritto e ad ogni libro i corrispettivi autori.

La relazione *LibroInsegnamento*, invece, conterrà le chiavi delle traduzioni nel modello relazionale di *Libro* e *Insegnamento*, per associare ad ogni insegnamento i libri adottati e ad ogni libro gli insegnamenti che lo utilizzano.

4.3 Schema logico relazionale

Utente(Email, Nome, Cognome, Password, DataDiNascita, *Università*, *CorsoDiLaurea*)

- Vincoli di integrità:
  - Vincolo di integrità referenziale tra CorsodiLaurea,Università e l'entità CorsoDiLaurea

Messaggio(TimeStamp, *Mittente*, *Destinatario*, Testo)

- Vincoli di integrità:
  - Vincolo di integrità referenziale tra EmailMittente, EmailDestinatario ed Utente

Libro(ISBN, Titolo, Prezzo, Anno, *Editore)*

- Vincoli di integrità:
  - Vincolo di integrità referenziale tra Editore e l'entità Editore

Descrizione(*Libro*, Testo)

- Vincoli di integrità:
  - Vincolo di integrità referenziale tra Libro e l'entità Libro

Autore(Id, Nome, Cognome)

Editore(Id, Nome)

AutoreLibro(*Libro*, *Autore*)

- Vincoli di integrità:

    - Vincolo di integrità referenziale tra Libro e l'entità Libro,
    - Vincolo di integrità referenziale tra Autore e l'entità Autore

NonInEvidenza(Id, TimeStamp, Testo, Motivazione, Prezzo, *Utente*, *Libro*)

- Vincolo di integrità:
    - Vincolo di integrità referenziale tra Utente e l'entità Utente
    - Vincolo di integrità referenziale tra Libro e l'entità Libro

Vendita(Id, TimeStamp, Testo, Prezzo, *Utente*, *Libro*)

- Vincoli di integrità:
    - Vincolo di integrità referenziale tra Utente e l'entità Utente
    - Vincolo di integrità referenziale tra Libro e l'entità Libro

Ricerca(Id, TimeStamp, Testo, *Utente*, *Libro*)

- Vincoli di integrità:
    - Vincolo di integrità referenziale tra Utente e l'entità Utente
    - Vincolo di integrità referenziale tra Libro e l'entità Libro

Università(Nome,Telefono, Indirizzo)

CorsodiLaurea(Nome,*Università)*

- Vincoli di integrità:
    - Vincolo di integrità referenziale tra Università e l'entità Università

Insegnamento(ID, Nome,*CorsoDiLaurea*,*Università*,Anno)

- Vincoli di integrità:
    - Vincolo di integrità referenziale tra CorsoDiLaurea, Università e l'entità CorsoDiLaurea

LibroInsegnamento(*Libro*,*Insegnamento*)

- Vincoli di integrità:
    - Vincolo di integrità referenziale tra Libro e l'entità Libro
    - Vincolo di integrità referenziale tra Insegnamento e l'entità Insegnamento

4.3.1 Descrizione relazioni

Utente:

- email: varchar(50), primary key
- password: varchar(256), not null
- nome: varchar(50), not null
- cognome: varchar(50), not null
- dataDiNascita: date, not null
- corso: varchar(100)
- università: varchar(50)

Messaggio:

- tmstmp: timestamp
- testo: varchar(320)
- mittente: varchar(50)
- destinatario: varchar(50)
- Primary key: tmstmp, mittente, destinatario

Libro:

- ISBN: bigint, primary key
- titolo: varchar(250), not null
- editore: integer, not null
- prezzo: deciamal(5,2)
- anno: year(4)

Autore:

- id: integer, primary key
- nome: varchar(50)

- cognome: varchar(50)

Editore:

- id: integer
- nome: varchar(50)

Descrizione:

- libro: bigint, primary key
- testo: TEXT
- copertina: BLOB

AutoreLibro:

- autore: integer
- libro: bigint
- Primary key(autore, libro)

NonInEvidenza:

- id: integer, primary key
- testo: varchar(360), default null
- timestamp: timestamp
- motivazione: bool
- prezzo: decimal(5,2)
- utente: varchar(50), not null
- libro: bigint not null

Vendita:

- id: integer, primary key

- testo: varchar(360), default null
- timestamp: timestamp
- prezzo: decimal(5,2)
- utente: varchar(50), not null
- libro: bigint not null

Ricerca:

- id: integer, primary key
- testo: varchar(360), default null
- timestamp: timestamp
- utente: varchar(50), not null
- libro: bigint not null

Università:

- nome: varchar(50), primary key
- indirizzo: varchar(150)
- telefono: varchar(150)

CorsoDiLaurea:

- nome: varchar(100)
- università: varchar(50)
- Primary key(nome, università)

Insegnamento:

- id: integer primary key

- nome: varchar(50)
- corsoDiLaurea: varchar(100)
- università: varchar(50)
- anno: enum('1', '2', '3', '4', '5', '6', '7')

LibroInsegnamento:

- insegnamento: integer
- libro: bigint
- Primary key(libro, insegnamento)

5 Implementazione delle base di dati in MYSQL

5.1 Creazione delle tabelle in MYSQL

```
1. /* Creazione della tabella Utente */
2. CREATE TABLE Utente(
3.       email VARCHAR(50),
4.       nome VARCHAR(50) NOT NULL,
5.       cognome VARCHAR(50) NOT NULL,
6.       password VARCHAR(100) NOT NULL,
7.       dataDiNascita DATE NOT NULL,
8.       corso VARCHAR(100),
9.       universita VARCHAR(50),
10.      PRIMARY KEY(email),
11.      FOREIGN KEY (corso,universita) REFERENCES
12.      CorsoDiLaurea(nome,universita)
13. )ENGINE=InnoDB CHARACTER SET utf8
14. COLLATE utf8_general_ci;
15.
16./* Creazione della tabella messaggio */
17.CREATE TABLE Messaggio(
18.      tmstmp TIMESTAMP,
19.      testo VARCHAR(320),
20.      mittente VARCHAR(50),
21.      destinatario VARCHAR(50),
22.      PRIMARY KEY (tmstmp, mittente, destinatario),
23.      FOREIGN KEY (mittente) REFERENCES Utente(email),
```

```
24.      FOREIGN KEY (destinatario) REFERENCES Utente(email)
25.  )ENGINE=InnoDB CHARACTER SET utf8
26.  COLLATE utf8_general_ci;
27.
28. /* Creazione della tabella annunci non in evidenza */
29.  CREATE TABLE NonInEvidenza(
30.      id INTEGER AUTO_INCREMENT,
31.      testo VARCHAR(360) DEFAULT NULL,
32.      tmstmp TIMESTAMP NOT NULL,
33.      motivazione BOOLEAN,
34.      utente VARCHAR(50) NOT NULL,
35.      libro INTEGER NOT NULL,
36.      prezzo DECIMAL(5,2) DEFAULT NULL,
37.      PRIMARY KEY(id),
38.      FOREIGN KEY (utente) REFERENCES Utente(email),
39.      FOREIGN KEY (libro) REFERENCES Libro(ISBN)
40.  )ENGINE=InnoDB CHARACTER SET utf8
41.  COLLATE utf8_general_ci;
42.
43. /* Creazione della tabella annunci di vendita */
44.  CREATE TABLE Vendita(
45.      id INTEGER AUTO_INCREMENT,
46.      testo VARCHAR(360) DEFAULT NULL,
47.      tmstmp TIMESTAMP NOT NULL,
48.      prezzo DECIMAL(5,2),
49.      utente VARCHAR(50) NOT NULL,
```

```
50.      libro INTEGER NOT NULL,
51.      PRIMARY KEY(id),
52.      FOREIGN KEY (utente) REFERENCES Utente(email),
53.      FOREIGN KEY (libro) REFERENCES Libro(ISBN)
54.  )ENGINE=InnoDB CHARACTER SET utf8
55.  COLLATE utf8_general_ci;
56.
57. /* Creazione della tabella annunci di ricerca */
58.  CREATE TABLE Ricerca(
59.      id INTEGER AUTO_INCREMENT,
60.      testo VARCHAR(360) DEFAULT NULL,
61.      tmstmp TIMESTAMP NOT NULL,
62.      utente VARCHAR(50) NOT NULL,
63.      libro INTEGER NOT NULL,
64.      PRIMARY KEY(id),
65.      FOREIGN KEY (utente) REFERENCES Utente(email),
66.      FOREIGN KEY (libro) REFERENCES Libro(ISBN)
67.  )ENGINE=InnoDB CHARACTER SET utf8
68.  COLLATE utf8_general_ci;
69.
70. /* Creazione della tabella libro */
71.  CREATE TABLE Libro(
72.      ISBN INTEGER,
73.      titolo VARCHAR(250) NOT NULL,
74.      editore INTEGER NOT NULL,
75.      prezzo DECIMAL(5,2),
```

```
76.      anno YEAR(4),
77.      PRIMARY KEY(ISBN),
78.      FOREIGN KEY (editore) REFERENCES Editore(id)
79.  )ENGINE=InnoDB CHARACTER SET utf8 COLLATE
80.  utf8_general_ci;
81.
82. /* Creazione della tabella descrizione, rappresentante la
    descrizione di un libro */
83.  CREATE TABLE Descrizione(
84.      libro INTEGER,
85.      testo TEXT,
86.      copertina BLOB,
87.      PRIMARY KEY(libro),
88.      FOREIGN KEY (libro) REFERENCES Libro(ISBN)
89.  )ENGINE=InnoDB CHARACTER SET utf8
90.  COLLATE utf8_general_ci;
91.
92. /* Creazione della tabella autore */
93. CREATE TABLE Autore(
94. id INTEGER AUTO_INCREMENT,
95. nome VARCHAR(50),
96. cognome VARCHAR(50),
97. PRIMARY KEY (id)
98. )ENGINE=InnoDB CHARACTER SET utf8
99. COLLATE utf8_general_ci;
100.
```

```
/* Creazione della tabella autore-libro */
CREATE TABLE AutoreLibro(
    autore INTEGER,
    libro INTEGER,
    PRIMARY KEY (autore, libro),
    FOREIGN KEY (autore) REFERENCES Autore(id),
    FOREIGN KEY (libro) REFERENCES Libro(ISBN)
)ENGINE=InnoDB CHARACTER SET utf8
COLLATE utf8_general_ci;

/* Creazione della tabella editore */
CREATE TABLE Editore(
    id INTEGER AUTO_INCREMENT,
    nome VARCHAR(50),
    PRIMARY KEY(id)
 )ENGINE=InnoDB CHARACTER SET utf8
 COLLATE utf8_general_ci;

/* Creazione della tabella rappresentante un corso di
  laurea */
CREATE TABLE CorsoDiLaurea(
    nome VARCHAR(200),
    universita VARCHAR(100),
    PRIMARY KEY (nome, universita),
    FOREIGN KEY (universita) REFERENCES Universita(nome)
)ENGINE=InnoDB CHARACTER SET utf8
```

```
126. COLLATE utf8_general_ci;
127.
128. /* Creazione della tabella Università */
129. CREATE TABLE Universita(
130.     nome VARCHAR(100),
131.     indirizzo VARCHAR(150),
132.     telefono VARCHAR(13),
133.     PRIMARY KEY (nome)
134. )ENGINE=InnoDB CHARACTER SET utf8
135.  COLLATE utf8_general_ci;
136.
137. /* Creazione della tabella Insegnamento, rappresentante
   gli insegnamenti di un corso di laurea */
138. CREATE TABLE Insegnamento(
139.     id INTEGER AUTO_INCREMENT,
140.     nome VARCHAR(100) NOT NULL,
141.     corsoDiLaurea VARCHAR(200),
142.     universita VARCHAR(100),
143.     anno ENUM('1','2','3','4','5','6','7'),
144.     PRIMARY KEY (id),
145.     FOREIGN KEY (corsoDiLaurea, universita) REFERENCES
146.     CorsoDiLaurea(nome, universita)
147. )ENGINE=InnoDB CHARACTER SET utf8
148.  COLLATE utf8_general_ci;
149.
150.
```

```
151. /* Creazione della tabella libro insegnamento che
     associa i libri ai loro insegnamenti */
152. CREATE TABLE LibroInsegnamento(
153.     insegnamento INTEGER,
154.     libro INTEGER,
155.     PRIMARY KEY(insegnamento, libro),
156.     FOREIGN KEY(insegnamento) REFERENCES
157.     Insegnamento(id),
158.     FOREIGN KEY(libro) REFERENCES Libro(ISBN)
159. )ENGINE=InnoDB CHARACTER SET utf8
160. COLLATE utf8_general_ci;
```

5.2 Query

5.2.1 Query 1

La *query 1* restituisce la lista dei libri di cui sono presenti annunci di vendita o di ricerca i cui libri sono utilizzati in università differenti da quella dell'utente che ha pubblicato l'annuncio.

```
1. SELECT l.ISBN, l.titolo FROM Libro AS l
2. WHERE l.ISBN IN( (SELECT r.libro FROM Ricerca AS r JOIN
3. Utente AS u ON r.utente = u.email
4. WHERE EXISTS(
5.  SELECT * FROM ((Libro AS l1 JOIN LibroInsegnamento AS li
6.  ON l1.ISBN = li.libro) JOIN
7.  Insegnamento AS i ON li.insegnamento = i.id)
8.  JOIN CorsoDiLaurea AS cdl ON i.corsoDiLaurea = cdl.nome
9.  WHERE cdl.universita<>u.universita AND
```

```
10. l1.ISBN = r.libro)
11. UNION
12. (SELECT v.libro FROM Vendita AS v JOIN Utente AS u
13. ON v.utente = u.email
14. WHERE EXISTS(SELECT * FROM ((Libro AS l1
15. JOIN LibroInsegnamento AS li ON l1.ISBN = li.libro)
16. JOIN Insegnamento AS i ON li.insegnamento = i.id)
17. JOIN CorsoDiLaurea AS cdl ON i.corsoDiLaurea = cdl.nome
18. WHERE cdl.universita<>u.universita AND
19. l1.ISBN = v.libro))));
```

Risultato:

ISBN	titolo
9780471438090	Applied Combinatorics
9788838665899	Geometria analitica con elementi di algebra lineare
9788838668005	Basi Di Dati-Modelli e linguaggi di interrogazione
9788887331943	Appunti di programmazione ad oggetti

5.2.2 Query 2

La *query 2* restituisce l'insieme di utenti che hanno pubblicato il maggior numero di annunci di ricerca e di vendita. Per la realizzazione di questa query vengono create due view(*UtenteNumeroAnnunciRicerca* e *UtenteNumeroAnnunciVendita*) che contengono rispettivamente il numero di annunci di vendita e ricerca pubblicati dal singolo utente.

```
1. DROP VIEW IF EXISTS UtenteNumeroAnnunciRicerca;
2. DROP VIEW IF EXISTS UtenteNumeroAnnunciVendita;
3. CREATE VIEW UtenteNumeroAnnunciRicerca(utente,nr) AS
4.   SELECT u.email , COUNT(r.id)
5.   FROM Utente AS u LEFT JOIN Ricerca AS r
6.   ON u.email = r.utente
```

```
7.    GROUP BY u.email;
8.  CREATE VIEW UtenteNumeroAnnunciVendita(utente,nv) AS
9.    SELECT u.email, COUNT(v.id)
10.   FROM Utente AS u LEFT JOIN Vendita AS v
11.   ON u.email = v.utente
12.   GROUP BY u.email;
13. SELECT u.email,u.nome,u.cognome,unr.nr+unv.nv AS totale
14. FROM (Utente AS u JOIN UtenteNumeroAnnunciRicerca AS unr
15. ON u.email = unr.utente) JOIN UtenteNumeroAnnunciVendita
16.  AS unv ON u.email = unv.utente
17. WHERE unr.nr + unv.nv >= ALL (
18.  SELECT DISTINCT (unr1.nr + unv1.nv)
19.  FROM UtenteNumeroAnnunciVendita AS unv1 JOIN
20. UtenteNumeroAnnunciRicerca AS unr1
21.  ON unv1.utente = unr1.utente));
```

Risultato:

email	nome	cognome	totale
email11@email.com	Ivo	Neri	2
email3@email.com	Itala	Loggia	2
marco@email.com	Marco	Zanella	2

5.2.3 Query 3

La *query 3* restituisce le università in cui sono utilizzati i libri che hanno il maggior numero di annunci di vendita associati.

```
1.  SELECT * FROM Universita
2.  WHERE Universita.nome IN(
```

```
3.  SELECT i.universita
4.  FROM Insegnamento AS i JOIN LibroInsegnamento AS li
5.  ON i.id = li.insegnamento JOIN Vendita AS v
6.  ON v.libro = li.libro
7.  GROUP BY i.universita
8. HAVING COUNT(DISTINCT v.libro) >= ALL(
9.   SELECT COUNT(DISTINCT v1.libro)
10.  FROM Insegnamento AS i1 JOIN LibroInsegnamento AS li1
11.  ON i1.id = li1.insegnamento JOIN Vendita AS v1
12.  ON v1.libro = li1.libro
13.  GROUP BY i1.universita) );
```

Risultato:

nome	indirizzo	telefono
Università degli Studi di Padova	Via VIII Febbraio, 2 - 35122 Padova	+390498275111

5.2.4 Query 4

La *query 4* restituisce l'utente con il maggior scarto fra la media dei prezzi di vendita degli annunci che ha pubblicato in *Vendita* e la media dei prezzi di copertina associati ai libri degli annunci a cui fanno riferimento.

```
1. SELECT u.email, u.nome, u.cognome,
2. AVG(l.prezzo) -AVG (v.prezzo) AS scarto
3. FROM Utente AS u JOIN Vendita AS v ON u.email = v.utente
4. JOIN Libro AS l ON v.libro = l.ISBN
5. GROUP BY u.email
6. HAVING (AVG(l.prezzo) - AVG(v.prezzo)) >= ALL(
7.  SELECT AVG(l1.prezzo) - AVG(v1.prezzo)
```

```
8.  FROM Vendita AS v1 JOIN Libro AS l1
9.  ON v1.libro = l1.ISBN GROUP BY v1.utente)
```

Risultato:

email	nome	cognome	scarto
email11@email.com	Ivo	Neri	37.00

5.2.5 Query 5

La *query 5* restituisce l'insieme di libri che sono associati ad una ed una sola università.

```
1. SELECT l.ISBN, l.titolo, l.prezzo FROM Libro AS l
2. WHERE l.ISBN NOT IN(
3.  SELECT li1.libro
4.  FROM (LibroInsegnamento AS li1 JOIN Insegnamento AS i1
5.  ON li1.insegnamento = i1.id) JOIN
6. (LibroInsegnamento AS li2 JOIN Insegnamento AS i2
7.  ON li2.insegnamento = i2.id) ON li1.libro = li2.libro
8.  AND i1.universita<>i2.universita)
9.  AND l.ISBN IN(
10. SELECT LibroInsegnamento.libro FROM LibroInsegnamento );
```

Risultato:

ISBN	titolo	prezzo
9780471438090	Applied Combinatorics	185.78
9788838665899	Geometria analitica con elementi di algebra lineare	29.00
9788838668005	Basi Di Dati-Modelli e linguaggi di interrogazione	42.00
9788887331943	Appunti di programmazione ad oggetti	22.00
9788896477670	Calcolo numerico-esercizi	22.00

5.2.6 Query 6

La *query 6* restituisce l'insieme di utenti che non hanno mai pubblicato annunci oppure che hanno inserito solamente annunci che sono scaduti da almeno un anno.

```
1. SELECT u.email,u.nome,u.cognome FROM Utente AS u
2. WHERE u.email NOT IN(
3.   SELECT utente FROM Vendita UNION
4.   SELECT utente FROM Ricerca UNION
5.   SELECT utente FROM NonInEvidenza ) OR u.email IN(
6.   SELECT utente FROM NonInEvidenza AS n
7. WHERE n.utente NOT IN( SELECT utente FROM Vendita UNION
8.       SELECT utente FROM Ricerca )
9.    AND n.motivazione = TRUE
10.   AND NOT EXISTS(
11.   SELECT * FROM NonInEvidenza AS n1
12.   WHERE n1.utente = n.utente AND
13.   SUBDATE(NOW(),INTERVAL 1 YEAR) <
14.  ADDDATE(n1.tmstmp, INTERVAL 6 MONTH)) )
```

Risultato:

email	nome	cognome
email17@email.com	Luigi	Verdi
email18@email.com	Simone	Capon
email19@email.com	Cristiano	Palermo
email20@email.com	Alessandra	Trevisan
email21@email.com	Mario	Cremonesi
email22@email.com	Alberto	Ricci
email5@email.com	Angelica	Padovano
email6@email.com	Marcello	Esposito
email8@email.com	Luigi	De Luca
email9@email.com	Fabrizia	Giordano

5.3 Funzioni

5.3.1 Funzione 1

La funzione *LibriPerUtente*, una volta passato lo username di un utente, restituisce la lista di tutti i libri associati all'università che l'utente stesso frequenta. Se non dovesse essere associata alcuna facoltà la lista ritornata sarebbe vuota. Poiché non è possibile in MYSQL ritornare una tabella, si è deciso di utilizzare la funzione GROUP_CONCAT() che concatena l'ISBN ed il titolo di un libro utilizzando come separatori dei tag HTML. Questi tag creano una riga di una tabella in HTML ed in questo modo i risultati verranno formattati nel modo corretto all'interno della pagina HTML. Questa scelta è stata fatta poiché è stato previsto che i risultati di questa funzione vengono utilizzati solamente da una pagina HTML.

```
1. DELIMITER $$
2. CREATE FUNCTION LibriPerUtente(emailutente VARCHAR(50))
3. RETURNS TEXT
4. BEGIN
5.   DECLARE uni VARCHAR(50);
6.   DECLARE ris TEXT;
7.   SELECT universita INTO uni FROM Utente WHERE
8.  Utente.email=emailutente;
9.   SELECT GROUP_CONCAT(
10. DISTINCT '<tr><td>',l.ISBN,'</td><td>',
11. l.titolo SEPARATOR '</td></tr>') INTO ris
12. FROM Libro AS l JOIN LibroInsegnamento AS li ON
13. l.ISBN=li.libro JOIN Insegnamento AS i ON
14. i.id=li.insegnamento
15. WHERE universita=uni;
16. RETURN ris;
```

```
17. END $$
```

Risultato:

Selezionando come username "email7@email.com" il risultato è il seguente:

ISBN	titolo
9780471438090	Applied Combinatorics
9788838665899	Geometria analitica con elementi di algebra lineare
9788838668005	Basi Di Dati-Modelli e linguaggi di interrogazione
9788887331943	Appunti di programmazione ad oggetti
9788896477670	Calcolo numerico-esercizi

5.3.2 Funzione 2

La funzione *MediaAnnuaPv* dato un anno, restituisce la media dei prezzi degli annunci di vendita in quel determinato anno. Poiché gli annunci si trovano sia all'interno di *NonInEvidenza* che all'interno di *Vendita*, per ottenere la media totale, vengono contati tutti gli annunci di vendita presenti in *NonInEvidenza* facendone anche la somma dei prezzi di vendita, poi viene fatta la stessa cosa sugli annunci in *Vendita* ed infine viene calcolata la media.

```
1. DELIMITER $$
2. CREATE FUNCTION MediaAnnuaPV(anno YEAR)
3. RETURNS DECIMAL(6,2)
4. BEGIN
5.   DECLARE sumScad DECIMAL(8,2) DEFAULT 0;
6.   DECLARE sumVend DECIMAL(8,2) DEFAULT 0;
7.   DECLARE nScad INTEGER;
8.   DECLARE nVend INTEGER;
9.      DECLARE media DECIMAL(8,2) DEFAULT 0;
10.  SELECT SUM(ne.prezzo),COUNT(*) INTO sumScad, nScad
11.  FROM NonInEvidenza AS ne WHERE ne.prezzo IS NOT NULL
12. AND YEAR(ne.tmstmp) = anno;
```

```
13.  SELECT SUM(v.prezzo),COUNT(*) INTO sumVend, nVend
14.  FROM Vendita AS v WHERE YEAR(v.tmstmp) = anno;
15.  IF nScad = 0 AND nVend = 0 THEN
16.      RETURN NULL;
17.  ELSEIF nScad = 0 THEN
18.      SET sumScad = 0;
19.  ELSEIF nVend = 0 THEN
20.      SET sumVend = 0;
21.  END IF;
22.  SET media = (sumScad + sumVend) / (nScad + nVend);
23.  RETURN media;
24. END $$
25. DELIMITER ;
```

Risultato:

Selezionando come anno il 2015 il risultato restituito dalla funzione è **€20.50**.

5.4 Trigger

5.4.1 Trigger 1 e 2

I trigger *elimina_annunci_ricerca* ed il trigger *elimina_annunci_vendita*, sono dei trigger di tipo `after` che vengono rispettivamente attivati quando viene effettuata una delete su *Ricerca* o su *Vendita*. Il loro compito è quello di controllare se l'annuncio rimosso era scaduto o meno per poi inserirlo all'interno della tabella *NonInEvidenza* settando a `false` la motivazione nel caso lo fosse, a `true` nel caso non lo fosse. In questo modo gli annunci rimossi da *Vendita* e *Ricerca* vengono subito copiati all'interno di *NonInEvidenza* rendendo il tutto automatico.

```
1.  DELIMITER $$
2.  CREATE TRIGGER elimina_annunci_ricerca AFTER DELETE ON
```

```
  Ricerca FOR EACH ROW
3. BEGIN
4.   DECLARE m BOOLEAN;
5.   DECLARE scadenza DATE;
6.   SET scadenza = ADDDATE(DATE(OLD.tmstmp), INTERVAL 6
7.  month);
8.   IF scadenza < CURDATE() THEN
9.       SET m = FALSE;
10.  else
11.      SET m = TRUE;
12.  END IF;
13.  INSERT INTO NonInEvidenza
14.   (testo,tmstmp,motivazione,utente,libro)
15.   VALUES
16.   (OLD.testo, OLD.tmstmp,m,OLD.utente,OLD.libro);
17. END $$
18. DELIMITER;
19.
20. DELIMITER $$
21. CREATE TRIGGER elimina_annunci_vendita AFTER DELETE ON
  Vendita FOR EACH ROW
22. BEGIN
23.  DECLARE m BOOLEAN;
24.  DECLARE scadenza date;
25.  SET scadenza = ADDDATE(DATE(OLD.tmstmp), interval 6
26.  month);
```

```
27.  IF scadenza < CURDATE() THEN
28.      SET m = FALSE;
29.  ELSE
30.      SET m = TRUE;
31.  END IF;
32.  INSERT INTO NonInEvidenza
33. (testo,tmstmp,motivazione,utente,libro,prezzo)
34.  VALUES
35. (OLD.testo,OLD.tmstmp,m,OLD.utente,OLD.libro,OLD.prezzo);
36. END $$
37. DELIMITER;
```

5.4.2 Trigger 3 e 4

Il trigger *elimina_incoerenza_ricerca* è un trigger di tipo `after` che viene attivato dopo l'inserimento di una nuova entry nella tabella *Ricerca*. Lo scopo di questo trigger è di controllare che l'utente non abbia inserito un annuncio di ricerca di un libro quando ha già messo in vendita lo stesso libro. Viene fatta una `select` in vendita andando a controllare se sono presenti annunci di vendita riguardanti il libro appena inserito in annunci di ricerca, andando a memorizzare il loro numero all'interno della variabile *contaV*. Successivamente, attraverso un ciclo `while` che itera *contaV* volte, vengono eliminati tutti gli annunci presenti in *Vendita* aventi lo stesso *ISBN* dell'annuncio appena inserito in *Ricerca*. Una volta finito il ciclo `while` all'interno della tabella Vendita non sarà più presente alcun annuncio riguardante lo stesso utente e lo stesso libro presente anche in ricerca.

Il trigger *elimina_incoerenza_vendita* è speculare al trigger sopra descritto, andando ad invertire le tabelle *Ricerca* e *Vendita*.

```
1.  DELIMITER $$
2.  CREATE TRIGGER elimina_incoerenza_ricerca AFTER INSERT ON
3.  Ricerca FOR EACH ROW
```

```
4. BEGIN
5.  DECLARE idVendita INTEGER DEFAULT NULL;
6.  DECLARE contaV INTEGER DEFAULT 0;
7.  SELECT COUNT(*) INTO contaV
8.  FROM Vendita
9.  WHERE Vendita.libro = NEW.libro AND
10.  Vendita.utente = NEW.Utente;
11.
12. WHILE contaV > 0 DO
13.     SELECT Vendita.id INTO idVendita
14.     FROM Vendita
15.     WHERE Vendita.libro = NEW.libro AND
16.     Vendita.utente = NEW.Utente LIMIT 1;
17.     DELETE FROM Vendita WHERE Vendita.id = idVendita;
18.     SET contaV = contaV - 1;
19. END WHILE;
20.
21. END $$
22. DELIMITER;
23.
24. DELIMITER $$
25. CREATE TRIGGER elimina_incoerenza_vendita AFTER INSERT ON
26. Vendita FOR EACH ROW
27. BEGIN
28.  DECLARE idRicerca INTEGER DEFAULT NULL;
29.  DECLARE contaV INTEGER DEFAULT 0;
```

```
30.  SELECT COUNT(*) INTO contaV
31.  FROM Ricerca
32.  WHERE Ricerca.libro = NEW.libro AND
33.  Ricerca.utente = NEW.Utente;
34.
35.  WHILE contaV > 0 DO
36.      SELECT Ricerca.id INTO idRicerca
37.      FROM Ricerca
38.      WHERE Ricerca.libro = NEW.libro AND
39.      Ricerca.utente = NEW.Utente LIMIT 1;
40.      DELETE FROM Ricerca WHERE Ricerca.id = idRicerca;
41.      SET contaV = contaV - 1;
42.  END WHILE;
43. END $$
44. DELIMITER;
```

5.5 Procedure

5.5.1 Procedura 1

La procedura *elimina_scaduti* ricerca all'interno delle tabelle *Vendita* e *Ricerca* se sono presenti annunci scaduti e nel caso lo fossero li elimina.

```
1.  DELIMITER $$
2.  CREATE PROCEDURE elimina_scaduti ()
3.  BEGIN
4.  DELETE FROM Ricerca WHERE
```

```
5. Ricerca.tmstmp < DATE_SUB(NOW(),INTERVAL 6 MONTH);
6. DELETE FROM Vendita
7. WHERE Vendita.tmstmp < DATE_SUB(NOW(),INTERVAL 6 MONTH);
8. END $$
9. DELIMITER;
```

5.5.2 Procedura 2

La procedura *inserimento_libro* inserisce un nuovo libro all'interno del database. I parametri che si aspetta questa procedura sono i dati del libro, del suo autore, dell'editore e del corso a cui viene associato. In prima istanza viene controllato se è presente l'editore del libro all'interno della tabella *Editore* e, nel caso non lo fosse, verrebbe creato un nuovo editore con i dati dell'editore passati alla procedura. Successivamente viene selezionato l'identificatore dell'editore dalla tabella *Editore* ed inserito all'interno della variabile *eid*. Lo stesso procedimento viene ripetuto con l'autore del libro e l'identificatore dell'autore viene inserito all'interno della variabile *aid*. A questo punto il libro viene inserito all'interno della tabella *Libro* utilizzando come editore il valore presente nella variabile *eid*. Poi viene aggiornata la tabella *AutoreLibro* andando ad aggiungere una nuova entry utilizzando come autore il valore presente all'interno della variabile *aid*. Il controllo successivo viene effettuato su *Insegnamento*, andando a controllo se l'insegnamento passato come parametro della funzione è presente all'interno della tabella stessa e nel caso non lo fosse verrebbe inserito il nuovo insegnamento. Al passo successivo viene aggiornata la tabella *LibroInsegnamento*, andando ad associare al libro appena inserito, il suo insegnamento.

```
1. DELIMITER $$
2. CREATE PROCEDURE inserimento_libro
3. (nomeAutore VARCHAR(50), cognomeAutore VARCHAR(50),
4. nomeEditore VARCHAR(50), lisbn INT(11),
5. ltitolo VARCHAR(250), lprezzo decimal(5,2),
```

```
6. lanno YEAR(4), linsegnamento VARCHAR(100),
7. lcorso VARCHAR(200), luniversita VARCHAR(100))
8. BEGIN
9.  DECLARE c INTEGER;
10. DECLARE eid INTEGER;
11. DECLARE aid INTEGER;
12. DECLARE iid INTEGER;
13. SELECT COUNT(*) INTO c FROM Editore
14. WHERE Editore.nome = nomeEditore;
15. IF c = 0 THEN
16.     INSERT INTO Editore(nome) VALUES (nomeEditore);
17. END IF;
18. SELECT Editore.id INTO eid FROM Editore
19. WHERE Editore.nome = nomeEditore;
20. SELECT COUNT(*) INTO c FROM Autore
21. WHERE Autore.nome = nomeAutore AND
22. Autore.cognome = cognomeAutore;
23. IF c = 0 THEN
24.     INSERT INTO Autore(nome,cognome)
25. VALUES (nomeAutore,cognomeAutore);
26. END IF;
27. SELECT Autore.id INTO aid FROM Autore
28. WHERE Autore.nome = nomeAutore AND
29. Autore.cognome = cognomeAutore;
30. INSERT INTO Libro(ISBN, titolo, editore, prezzo, anno)
31. VALUES (lisbn, ltitolo, eid, lprezzo, lanno);
```

```
32.  INSERT INTO AutoreLibro(autore, libro)
33.  VALUES (aid,lisbn);
34.  SELECT COUNT(*) INTO c FROM Insegnamento
35.  WHERE Insegnamento.nome = linsegnamento AND
36.  Insegnamento.corsoDiLaurea = lcorso AND
37.  Insegnamento.universita = luniversita;
38.  IF c = 0 THEN
39.   INSERT INTO Insegnamento(nome,corsoDiLaurea,universita)
40.   VALUES (linsegnamento,lcorso,luniversita);
41.  END IF;
42.  SELECT Insegnamento.id INTO iid FROM Insegnamento
43.  WHERE Insegnamento.nome = linsegnamento AND
44.  Insegnamento.corsoDiLaurea = lcorso AND
45.  Insegnamento.universita = luniversita;
46.  INSERT INTO LibroInsegnamento(insegnamento, libro)
47.  VALUES (iid,lisbn);
48. END $$
49. DELIMITER $$
```

5.6 View

5.6.1 View 1 e 2

Per la visualizzazione degli annunci di vendita e di ricerca sono state create le view *mostra_ricerca* e *mostra_vendita*. Queste view inoltre sono state utilizzate per semplificare le query di ricerca di annunci di vendita e di annunci di ricerca.

```
1. DROP VIEW IF EXISTS mostra_ricerca;
2. DROP VIEW IF EXISTS mostra_vendita;
3. CREATE VIEW mostra_ricerca(IDAnnuncio,ISBN,
4. Titolo,Editore,PrezzoCopertina,Testo,Utente,Universita)
5. AS SELECT r.id,l.ISBN, l.titolo, e.nome, l.prezzo,
   r.testo, r.utente, u.universita
6. FROM Libro AS l JOIN Editore AS e ON l.editore = e.id
7. JOIN Ricerca AS r ON r.libro = l.ISBN
8. JOIN Utente AS u ON u.email = r.utente;
9.
10. CREATE VIEW mostra_vendita(IDAnnuncio,ISBN,Titolo,
11. Editore,PrezzoCopertina, PrezzoVendita,Testo,
12. Utente,Universita) AS
13. SELECT v.id,l.ISBN, l.titolo, e.nome, l.prezzo, v.prezzo,
14. v.testo, v.utente, u.universita
15. FROM Libro AS l JOIN Editore AS e ON l.editore = e.id
16.   JOIN Vendita AS v ON v.libro = l.ISBN
17.   JOIN Utente AS u ON u.email = v.utente;
```

5.7 Eventi

5.7.1 Evento elimina scaduti

Per eliminare gli annunci scaduti in modo automatico, è stato creato l'evento “event daily_expired_ads”. Questo evento, una volta al giorno, chiama la procedura “elimina_scaduti” che va a controllare la presenza di annunci scaduti e ad eliminarli.

```
1. DELIMITER $$
2. CREATE event daily_expired_ads
3.   ON SCHEDULE EVERY 1 DAY
4.      STARTS NOW()
5.      DO
6.       call elimina_scaduti();
7. $$
```