Relazione Progetto Basi Di Dati

Autori: Cervone Giuseppe (134373) e Adami Nicola (106925)

1 Introduzione

Si vuole soddisfare, come richiesto dall'Università degli studi di Ferrara, la richiesta inerente alla creazione di un sistema informatico in grado di gestire la biblioteca universitaria per quanto concerne i libri, le utenze ed i prestiti. Un primo ostacolo da affrontare è la sezione in dipartimenti dell'ateneo, che di conseguenza richiede una gestione più capillare dato che, un libro può avere una o più copie di esso distribuite in un numero variabile di Facoltà, in aggiunta ad altre problematiche relative all'archivio dei tomi, come i dati a loro relativi non sempre di facile ripartizione; per poi passare all'amministrazione dell'utenza, che può in alcuni casi incappare in una situazione eccezionale come l'omonimia; fino ad arrivare alla situazione dei prestiti che richiede un'attenzione meticolosa nel verificare eventuali "accavallamenti" di prenotazioni. Gli strumenti utilizzati per creare uno strumento ad-hoc volto a sopperire tali problematiche sono stati HTML, con l'appoggio in alcune occorrenze del framework Bootstrap, oltre a CSS per completare l'esperienza FRONT END. La basi dati ove verranno registrati i vari dati e manipolati verrà interagita con SQL, che a sua volta PHP veicolerà con dovuti script sul nostro sito web.

2 Definizione delle Funzionalità

L'applicativo viene concepito per essere uno strumento funzionale ed intuitivo, con un Home-Page essenziale e chiara, volendo rendendo l'esperienza d'uso non stressante, considerando che il potenziale addetto che la potrebbe usare sottoposto a tale pressione. Partendo dalla cima di "index.php": - logo con collegamento ipertestuale all'HomePage, che ha lo scopo principale del "back to Home"; - barra principale con funzionalità specifiche per le quali non si necessita di una ricerca specifica; - doppia barra di ricerca per una rapida consultazione di utenti e libri. Le utenze si possono creare dall'apposito tasto nel menù a tendina ADD dove, una volta richiesti nome, cognome, telefono e indirizzo, verranno aggiunti alla base dati e simultaneamente si assegnerà Id univoco generato automaticamente; la barra di ricerca dedicata agli studenti permette sia una ricerca di tutte le persone inserendo un campo vuoto, altrimenti la ricerca parziale permette di cercare tutti i nomi con tale stringa; per intervenire sulla modifica o cancellazione di un'anagrafica è presente un ipertesto nel relativo Id e nella pagina seguente verrà mostrato lo stato attuale dei vari dati(la rimozione fa si che vengano rimossi anche i prestiti a carico dell'utente); per ogni nome ci sarà anche la possibilità di visitare i tutti prestiti a carico di tale soggetto. Per quanto concerne i libri all'interno della biblioteca la ricerca è simile a quella degli utenti, a differenza che cliccando sul titolo si potranno consultare tutti i dettagli relativi al libro, tra cui i vari autori ed il n. di copie; inoltre è presente per ogni tomo la possibilità di visualizzare tutti i prestiti in essere per le varie copie di tale libro. I prestiti vengono gestiti a partire dal tasto dedicato (Av. Loans) che permette una rapida consultazione all'intera tabella di quelli esistenti; qualora si voglia aggiungerne uno verranno richiesti Nome e cognome dell'utente, il nome del libro, la succursale interessata e la data di inizio prestito (la fine verrà calcolata secondo lo standard dei

30 gg.), se ci sarà almeno una copia disponibile allora l'aggiunta verrà eseguita correttamente. Mentre per le eventuali modifiche o rimozioni si potrà usufruire del tasto "mod/del" posizionato nelle varie tabelle con dei prestiti.

Le varie informazioni relative succursali della biblioteca sono facilmente accessibili dalla barra principale. Infine vengono offerte alcune statistiche relative all'archivio, tra le quali una dedicata alla filiale con più libri nei loro scaffali e ai libri con il maggior numero di copie.

3 Definizione Modello della Base Dati

3.1 Modello Entità' Relazione

Per la definizione del modello della base dati, partiamo col definire il diagramma Entità Relazione che descrive al meglio la base stessa.
Siamo partiti dal definire l'entità che descrive il libro per risolvere il problema dell'identificazione di più copie di un libro. Come gruppo siamo arrivati alla conclusione che un entità *BOOK*, collegata tramite l'associazione identificante *EXISTS* ad un entità debole *COPY* e' la soluzione ottimale, che ci permette un maggiore potere espressivo sia per il singolo problema, che per le altre relazioni che si dovranno formare. Abbiamo ritenuto corretto utilizzare un'associazione identificante ed un'entità debole perché non può esistere una copia di un libro se non esiste un libro; in più le relazioni che coinvolgono la singola copia di un libro possono essere differenziate dalle relazioni che coinvolgono il libro in generale e infine, grazie a questa scelta di progettazione, non si devono creare ulteriori chiavi identificative per una singola copia, ma e' sufficiente usare l'identificatore per il singolo libro (ISBN) in associazione ad un contatore di copie legate ad un ISBN.
Gli attributi di *BOOK* sono:
"Title" (titolo), "ISBN" (identificativo), "Language" (lingua), "Publication year" (Anno di pubblicazione), mentre *COPY* ha come unico attributo COPY_N (valore del contatore copia che e' parte dell'identificativo), e utilizzerà l'ISBN per identificarsi completamente.
Dato che ogni libro può essere scritto da uno o più autori, abbiamo di fatto creato l'entità *AUTHOR* con attributi "SSN" (identificatore), "First and Last Name" (nome e cognome), "Date of Birth" (data di nascita), "Place of birth" (luogo di nascita). Questa entità si collega con l'entità *BOOK* grazie alla relazione *Written_by*; vediamo quindi una prima differenza con le entità che si collegano con *COPY*:
Ogni libro può essere pubblicato da un editore, avendo più informazioni che vogliamo memorizzare riguardo il singolo editore abbiamo creato l'entità *EDITOR*, con attributi "Editor ID" (identificativo), "Name" (nome dell'editore), "Address" (indirizzo della sede), "Phone number" (numero di telefono). L'entità *EDITOR* e' connessa a *BOOK* grazie alla relazione *Published_by*.
Nel nostro mini-mondo esistono più librerie, e delle tali viene richiesto di memorizzare le informazioni basilari, quindi abbiamo l'entità *LIBRARY* con attributi "Library_id" (Identificativo), "Name" (Nome della libreria), "Address" (Indirizzo), "Telephone" (Numero di telefono). Memorizziamo grazie alla relazione *Is_Available* con attributo "Number of copies" collegata a *BOOK* se possiamo trovare un libro in una succursale, l'attributo tornerà utile in futuro. L'entità *LIBRARY* parteciperà anche alla relazione *Found_in* con *COPY*, cosi' da poter memorizzare, ove possibile, il prestito della copia di un libro.
L'entità *STUDENT* memorizza le informazioni legate ad ogni singolo studente iscritto alla libreria, ogni studente avrà uno "Student Number" (identificativo), "First and Last name" (nome e cognome), "Address" (indirizzo) e "Telephone" (numero di telefono), grazie all'aggiunta di studente possiamo delineare l'ultima relazione utile. La relazione *Rented_by* e' una relazione ternaria che mette in relazione lo studente con una copia che prende in affitto, da una libreria specifica;

la relazione avrà anche attributi "Start date" e "End date" per definire una data di inizio e fine prestito. Questa relazione ci aiuterà a memorizzare dove un singolo studente prende in prestito un libro particolare, ovviamente quella copia dovrà essere disponibile in quella libreria, ma questo verrà gestito in un secondo momento.

3.1.1 Studio Cardinalità Modello ER

Delineata la struttura del diagramma ER per la rispettiva base dati, possiamo delineare le cardinalità delle relazioni. Partendo dalla relazione *Published_by*, un editore può partecipare minimo una volta e massimo N volte visto che può pubblicare più di un libro (1:N), mentre un libro può partecipare minimo una volta e massimo una volta visto che, come da definizione del mini-mondo, sappiamo che un libro può essere pubblicato solo da un editore (1:1), quindi la relazione e' 1:N.
La relazione *Written_by* e' formata da *BOOK* e *AUTHOR*, un libro può essere scritto minimo da un autore, massimo da N autori (1:N), mentre un autore può scrivere minimo un libro e massimo M (1:M), quindi la relazione e' N:M.
Andando ad analizzare la cardinalità di *Exists*, vediamo che di un libro può esistere minimo una copia, massimo N (1:N), mentre una copia e' sempre e solo legata ad un libro (1:1), la relazione e' quindi 1:N.
Per la relazione *Is_available* partiamo da *BOOK*, dove un libro può essere disponibile minimo in una libreria, massimo in N librerie, visto che un libro può trovarsi in più succursali come da definizione di mini-mondo (1:N), una libreria invece può avere minimo 1 libro disponibili, massimo N (1:N), questo rende la relazione M:N.
Applichiamo un ragionamento simile alla relazione *Found_in*, dove l'entità *LIBRARY* ha partecipazione 1:N, mentre *COPY* ha cardinalità 0:N, quindi N:M in totale.
Infine, la relazione *Rented_by* ha partecipazione 0:N da *LIBRARY*, visto che si potrebbe avere una libreria senza prestiti attivi, ma si possono avere N librerie con prestiti attivi. Per quanto riguarda *STUDENT*, uno studente potrebbe non avere prestiti attivi, ma potrebbe averne più di uno, quindi 0:M, e una copia potrebbe non avere prestiti e al massimo avere un prestito a suo nome quindi 0:1, cardinalità risultante finale 0:N:M. P.S. Il diagramma ER finale e' incluso nell'appendice.

3.2 Modello Relazionale

3.2.1 Dal Modello ER al Modello Relazionale

Avendo delineato il modello ER che descrive la base dati possiamo, grazie ad un algoritmo di conversione, creare anche il diagramma relazionale. Il primo passo sarà convertire le entità *BOOK, COPY, STUDENT, LIBRARY, AUTHOR, EDITOR* in entità del diagramma relazionale, assegnando ad ognuna i suoi attributi e le chiavi primarie già definite. Passiamo adesso alle relazioni 1:N *Exists* e *Published_by*, per la relazione di esistenza dell'entità debole *COPY* alle quali aggiungiamo una chiave esterna "ISBN" nell'entità' *COPY*; la chiave esterna farà riferimento all'identificativo dell'entità *BOOK*. L'altra relazione 1:N e' quella che lega il libro al suo editore, visto che un libro non può essere pubblicato da più di due editori, aggiungiamo una chiave esterna in *BOOK* che referenzia l'id dell'editore. Ora ci restano solo relazioni del tipo M:N, andando a creare le tabelle *Written by, Found in, Rented by e Is Available*. Queste tabelle saranno formate da chiavi primarie esterne; tali chiavi faranno riferimento alle chiavi primarie delle entità che partecipano alla relazione, in più, nei casi di *Rented by e Is available*, aggiungeremo anche attributi "Start date", "End date" e "Number of copies" per *Is available*.
P.S. Lo schema risultante e' il secondo elemento dell'appendice.

3.2.2 Normalizzazione

Uno dei requisiti di questo modello relazionale e' la modellazione in 3NF. Partiamo quindi dall'analizzare se il diagramma parte dal soddisfare la 1NF. Per essere in 1NF, ciascun attributo del diagramma deve essere definito su un dominio con valori atomici, e ogni attributo deve contenere un singolo valore del dominio. Questa condizione e' rispettata in tutti gli attributi del diagramma tranne che dall'indirizzo trovato in più tabelle. Quindi atomizziamo l'indirizzo in "strada", "numero civico", "città"" e "CAP". Procediamo quindi a dividere l'attributo dell'indirizzo in queste tabelle.
Per rispettare le condizioni del 2NF, il diagramma deve essere in 1NF e per ogni relazione tutti gli attributi non chiave dipendono funzionalmente dall'intera chiave composta. Avendo creato la divisione tra un libro, e una copia del libro riusciamo a non avere problemi con la condizione della 2NF, in più i valori di *Rented_by* relativi alle date e il valore di *Is_available* numero copie dipendono direttamente dalla relazione, quindi le condizioni della 2NF sono rispettate.
Per essere in 3NF, il diagramma relazionale deve rispettare la condizione della 2NF e se tutti gli attributi non-chiave dipendono dalla chiave soltanto. Questa condizione può anche essere definita come la non presenza di una dipendenza transitiva all'interno delle relazioni. Questa e' rispettata per tutte le relazioni, quindi il diagramma e' in 3NF. La versione del diagramma normalizzata e' nell'appendice.

3.2.3 Vincoli del Modello Relazionale

Ricordiamo prima di analizzarli quali sono i tre tipi di vincoli del modello relazionale:
- Vincoli sulla chiave, quindi il fatto che una chiave debba essere univoca e presente in ogni entrata della base dati;
- Vincoli di integrità dell'entità, che specificano che gli attributi di chiave primaria non possono avere valori nulli;
- Vincoli di integrità referenziale, per cui il valore della chiave esterna della relazione può essere un valore uguale alla chiave primaria che si sta utilizzando oppure nullo, quindi la chiave non partecipa alla relazione.

Il vincolo sulla chiave viene rispettato grazie ai valori di partenza che assegniamo, nelle aggiunte che e' possibile fare (quella di un nuovo studente), abbiamo delineato grazie al tipo di dato AUTOINCREMENT viene automaticamente assegnato un identificativo.
Per studiare i vincoli di integrità dell'entità, ricordiamo che, per definizione, specificano gli attributi di chiave primaria di ciascuna relazione e non possono avere valori nulli; Le nostre chiavi primarie "ISBN", "Copy Number", "Student Number", "Library ID", "SSN" ed "Editor ID" sono per definizione non nulle, visto che averle nulle renderebbe impossibile l'identificazione e la creazione di relazioni, SQL e PHP ci aiuteranno anche a far si che questi vincoli vengano rispettati. Passando al vincolo di integrità referenziale, che e' quel vincolo per cui una chiave esterna della relazione deve essere uguale alla chiave primaria oppure avere un valore nullo, quindi non partecipare; questo vincolo e' rispettato creando le tabelle *Written by, Rented by, Found in e Is Available* e utilizzando come chiavi esterne le chiavi direttamente coinvolte nella relazioni, quindi non creando conflitti o chiavi esterne non corrette. Lo schema risultante e' il terzo elemento dell'appendice.

4 Definizione Interrogazioni e Funzionalità'

Nelle sezioni introduttive della relazione abbiamo definito le funzionalità e interrogazioni richieste per la base dati, poi le funzionalità e le interrogazioni scelte in autonomia.

A seguire le espressioni in algebra relazionale per le funzioni.

4.1 Algebra Relazionale

4.1.1 Statistiche

Cinque lingue più comuni:

$$COUNTLANG \leftarrow N\_of\_langs\, f\,_{(COUNT)}(Language)BOOK$$
$$\Pi_{(Language,N\_of\_books\_written)}COUNTLANG$$

Per questa espressione di algebra relazionale, basta utilizzare la funzione COUNT su Language di BOOK e stampare per ogni lingua, il numero di volte che si ripete.

Autore che ha scritto più libri:

$$AUTHOR\_WRITTEN \leftarrow AUTHOR \bowtie_{(SSN=AUTHOR\_ID)} WRITTEN\_BY$$
$$COUNTSSN \leftarrow N\_of\_books\_written\, f\,_{(COUNT)}(SSN)AUTHOR\_WRITTEN$$
$$MAXAUTHOR \leftarrow f_{(MAX)}(COUNTSSN)$$
$$\pi_{(First\_name,Last\_name,N\_of\_books\_written)}MAXAUTHOR$$

Per questa espressione di algebra relazionale, facciamo la JOIN tra l'entità che descrive l'autore e la relazione che include quanti libri ha scritto, poi calcoliamo il valore e lo stampiamo insieme al nome e al cognome.

Editore che ha pubblicato più libri:

$$EDITOR\_PUB \leftarrow EDITOR \bowtie_{(EDITOR\_ID=ID\_FOR\_EDITOR)} BOOK$$
$$COUNTID \leftarrow N\_of\_books\_published\, f\,_{(COUNT)}(ID\_FOR\_EDITOR)EDITOR\_PUB$$
$$MAXID \leftarrow f\,_{(MAX)}(COUNTID)$$
$$\pi_{(EDITOR\_ID,EDITOR\_NAME,N\_of\_books\_published)}MAXID$$

Questa interrogazione e' simile a quella precedente, solo che le operazioni avvengono tra la tabella *BOOK* e quella *EDITOR* visto che loro formano la relazione editore che ha pubblicato più libri.

Librerie con più libri:

$$LIB\_FOUND \leftarrow LIBRARY \bowtie_{(LIBRARY\_ID=LIB\_ID)} FOUND\_IN$$
$$COUNTLIB \leftarrow N\_of\_books\_in\_library\, f\,_{(COUNT)}(LIBRARY\_ID)LIB\_FOUND$$
$$\pi_{(Name,N\_of\_books\_in\_library)}COUNTLIB$$

Un'altra interrogazione simile alle altre viste in precedenza, questa ritorna le librerie in ordine per numero di libri contenuti in ognuna.

Cinque libri con più copie:

$$BOOKJOIN \leftarrow BOOK \bowtie (ISBN = ISBN)COPY$$
$$COUNTC \leftarrow N\_of\_copies\, f\,(COUNT)(ISBN)BOOKJOIN$$
$$\pi_{(TITLE,N\_of\_copies)}COUNTC$$

In questa interrogazione andiamo a fare la JOIN tra la tabella che rappresenta l'entità libro e quella che rappresenta l'entità copia. Facendo un conteggio per ISBN del risultato conteremo quante volte stampiamo ogni ISBN, quindi il conteggio di quante copie ha ogni libro (per quanto questo conto si poteva fare anche senza l'entità *BOOK*, noi la useremo comunque per stampare il nome dei cinque libri con più copie).

Tutti i libri in prestito ad un utente:

$$RESULT1 \leftarrow STUDENT \bowtie_{(STUDENT\_N=USER\_ID)} RENTED\_BY$$
$$RESULT2 \leftarrow RESULT1 \bowtie_{ISBN=ISBN} BOOK$$
$$\pi_{(TITLE,ID\_COPY,ISBN,LIB\_ID,STARTDATE,ENDDATE)}\sigma(STUDENT\_N = INPUT)$$

In questa interrogazione l'input e' uguale all'identificatore dello studente di cui vogliamo avere i prestiti. Essa potrebbe funzionare anche senza la JOIN con l'entità' che rappresenta il libro, ma la usiamo per poter stampare a video informazioni utili legate al libro preso in affitto

Ogni utente a cui e' in prestito un libro:

$$RESULT1 \leftarrow STUDENT \bowtie_{(STUDENT\_N=USER\_ID)} RENTED\_BY$$
$$RESULT2 \leftarrow RESULT1 \bowtie_{ISBN=ISBN} BOOK$$
$$\pi_{(F\_NAME,L\_NAME,STUDENT\_N,ID\_COPY,LIBRARY\_ID,STARTDATE,ENDDATE)}\sigma(ISBN = INPUT)$$

L'interrogazione qui presentata e' molto simile all'interrogazione precedente. Per quanto cambi l'ordine delle informazioni stampate all'utente finale e quello che noi consideriamo INPUT, le condizioni per la JOIN sono le stesse. Qui il valore INPUT e' l'identificatore del libro per cui vogliamo conoscere gli utenti che lo hanno preso in prestito.

Informazioni su un libro:

$$RESULT1 \leftarrow BOOK \bowtie_{(ISBN=BOOK\_ID)} WRITTEN\_BY$$
$$RESULT2 \leftarrow RESULT1 \bowtie_{(AUTHOR\_ID=SSN)} AUTHOR$$
$$RESULT3 \leftarrow RESULT2 \bowtie_{(ID\_EDITOR=EDITOR\_ID)} EDITOR$$
$$\pi_{(FIRST\_NAME,LAST\_NAME,DATE\_BIRTH,PLACE\_BIRTH,NAME)}\sigma(ISBN = INPUT)$$

Questa interrogazione ci stampa le informazioni legate agli autori di un certo libro. La funzionalità completa verrà vista nel paragrafo sulla scrittura in MySQL, visto che aggiungerà dei pezzi di codice PHP per prendere tutta una serie di informazioni legate ad un libro, e stamparle a video per l'utente, non solo le informazioni legate ai vari autori. Completate le scritture in algebra relazionale, passiamo alla traduzione in MySQL.

4.2 Scrittura in MySQL

Adesso traduciamo le scritture in algebra relazionale in scritture nel linguaggio MySQL, cosi' da poterle implementare nell'applicativo web. In questa sezione si troveranno anche delle parti scritte con il PHP, per assicurarsi la corretta funzionalità degli script. Visto che l'idea generale e' già stata ampiamente spiegata nelle scritture riguardanti l'algebra relazionale, ci limiteremo ad aggiungere dettagli solo applicabili all'SQL.

4.2.1 Statistiche

Cinque lingue più comuni:

```
SELECT LANGUAGE,COUNT(LANGUAGE) AS N_of_times
FROM BOOK GROUP BY LANGUAGE
ORDER BY N_of_times DESC LIMIT 5;
```

Per questa prima interrogazione ci basta contare il numero di volte che ogni lingua appare, poi ordiniamo in ordine decrescente per il valore e stampiamo solo le prime cinque a video. Abbiamo scelto le prime 5 lingue per mostrare l'alta disponibilità linguistica del sistema bibliotecario.

Autore che ha scritto più libri:

```
SELECT a.FIRST_NAME, a.LAST_NAME,
COUNT(a.SSN) AS N_of_books_written
FROM AUTHOR as a INNER JOIN WRITTEN_BY as w
WHERE a.SSN = w.AUTHOR_ID
GROUP BY a.SSN
ORDER BY N_of_books_written DESC LIMIT 1;
```

Per trovare l'autore che ha scritto più libri disponibili nel nostro sistema bibliotecario, ci basta fare una JOIN della tabella *AUTHOR* e della tabella *WRITTEN_BY* sulla base dei loro valori identificatori. Andiamo poi a contare quante volte ogni singolo identificatore dell'autore appare poi, come nelle interrogazioni precedenti, ordiniamo la tabella e stampiamo.

Editore che ha pubblicato più libri:

```
SELECT e.EDITOR_ID, e.NAME,
COUNT(b.ID_EDITOR) AS N_of_books_published
FROM BOOK as b INNER JOIN EDITOR as e
WHERE e.EDITOR_ID = b.ID_EDITOR
GROUP BY b.ID_EDITOR ORDER BY N_of_books_published
DESC LIMIT 1;
```

Troviamo l'editore che ha pubblicato più libri utilizzando la chiave esterna in *BOOK* di nome *ID_EDITOR*. Questa chiave ci lascia relazionare ogni libro all'editore che lo ha pubblicato, per quanto basterebbe questo per avere l'identificatore dell'editore che ha pubblicato più libri. Ricordiamo che noi siamo interessati al suo nome quindi, utilizziamo di nuovo la JOIN e poi contiamo per ID stampando anche il suo nome.

Succursale con più libri:

```
SELECT l.NAME, COUNT(f.LIBRARY_ID) AS N_of_books_in_library
FROM LIBRARY as l INNER JOIN FOUND_IN as f
WHERE f.LIBRARY_ID = l.LIB_ID
GROUP BY f.LIBRARY_ID
ORDER BY N_of_books_in_library
DESC LIMIT 1;
```

La succursale con più libri la troviamo eseguendo una JOIN tra la tabella *LIBRARY* e la tabella *FOUND_IN*. Queste due tabelle sono rispettivamente la tabella delle succursali e la tabella che

attribuisce ai libri una locazione in una libreria. Contando quante volte troviamo ogni identificativo per le succursali, troviamo quale contiene più libri. Abbiamo scelto come gruppo di stampare la succursale con più libri, per avere un modo facile di tenere sotto controllo sempre la succursale più fornita.

Cinque libri con più copie:

```
SELECT b.TITLE, COUNT(c.ISBN) AS N_of_copies
FROM BOOK as b INNER JOIN COPY as c
WHERE b.ISBN = c.ISBN
GROUP BY c.ISBN ORDER BY N_of_copies
DESC LIMIT 5;
```

L'interrogazione alla base dati che ci calcola i cinque libri con più copie opera eseguendo una JOIN tra la tabella dei libri e la tabella delle copie per l'identificativo che le accomuna, ordinando poi i risultati e contando quante volte si ripete un singolo ISBN nella tabella copie. Abbiamo scelto questa interrogazione aggiuntiva per avere un modo di controllare quali sono i libri con la maggiore disponibilità in libreria, abbiamo scelto di stampare a video i cinque libri visto che alla creazione della base dati, c'erano cinque libri tutti con lo stesso numero di copie più alto.

Tutti i libri in prestito ad un utente:

```
SELECT b.TITLE, r.ID_COPY, r.ISBN, r.LIBRARY_ID,
r.STARTDATE, r.ENDDATE FROM
STUDENT AS s INNER JOIN RENTED_BY AS r
ON s.STUDENT_N = r.USER_ID
INNER JOIN BOOK AS b ON b.ISBN = r.ISBN
WHERE s.STUDENT_N = '$STUDENT_N'
```

Per stampare tutti i libri in prestito ad un utente a video, utilizziamo la tabella *RENTED_BY*, ed eseguiamo una JOIN per entrambe le sue chiavi esterne, solo che invece che utilizzare l'ISBN della tabella *COPY*, utilizziamo quello della tabella *BOOK*, che ci lascia accesso ai titoli dei libri. Usiamo come condizione ulteriore il fatto che abbiamo ricevuto in input l'identificatore per lo studente cosi' da stampare i risultati solo per lui.

Ogni utente a cui e' in prestito un libro:

```
SELECT s.F_NAME, s.L_NAME, s.STUDENT_N, r.ID_COPY, r.LIBRARY_ID,
r.STARTDATE, r.ENDDATE FROM STUDENT AS s
INNER JOIN RENTED_BY AS r ON s.STUDENT_N = r.USER_ID
INNER JOIN BOOK AS b ON b.ISBN = r.ISBN WHERE b.ISBN = '$ISBN'
```

Applichiamo qui la stessa logica applicata per l'interrogazione "Tutti i libri in prestito ad un utente", l'unica differenza e' che in questo caso la condizione ulteriore e' comandata dall'identificativo del libro, non dall'identificativo dell'utente.

4.2.2 Funzionalità

Informazioni relative al libro:

```
$query = "SELECT a.FIRST_NAME, a.LAST_NAME,
a.DATE_BIRTH, a.PLACE_BIRTH, e.NAME, COUNT(c.COPY_N)
```

```
FROM BOOK as b
INNER JOIN WRITTEN_BY AS w ON b.ISBN = w.BOOK_ID
INNER JOIN AUTHOR AS a ON w.AUTHOR_ID = a.SSN
INNER JOIN EDITOR AS e ON b.ID_EDITOR = e.EDITOR_ID
INNER JOIN COPY AS c ON c.ISBN = b.ISBN
WHERE b.ISBN = '$ISBN'
GROUP BY a.SSN;";

$query = mysqli_query($link, $sql);
$FIRST_NAME = "";
$LAST_NAME = "";
$PLACE_BIRTH = "";
$DATE_BIRTH = "";

while ($row = mysqli_fetch_array($query)) {
$FIRST = $row['FIRST_NAME'];
$FIRST_NAME = $FIRST_NAME . $FIRST . ";" . " ";
$LAST = $row['LAST_NAME'];
$LAST_NAME = $LAST_NAME . $LAST . ";" . " ";
$DATE = $row['DATE_BIRTH'];
$DATE_BIRTH = $DATE_BIRTH . $DATE . ";" . " ";
$PLACE = $row['PLACE_BIRTH'];
$PLACE_BIRTH = $PLACE_BIRTH . $PLACE . ";" . " ";
$NAME = $row['NAME'];
$COPY_N = $row[5];
}
```

Questo e' un esempio di codice SQL e PHP mischiati per assicurarsi un funzionamento corretto dell'interrogazione. Con questa interrogazione inizialmente volevamo stampare a video tutte le informazioni utili per un libro, e grazie a questa scrittura abbiamo come output una tabella dove ad ogni riga abbiamo le informazioni legate ad ogni singolo autore, seguite dalle informazioni sul libro. Grazie al ciclo while utilizzato nella seconda parte del codice, concateniamo i valori riguardanti gli autori che partono con l'essere vuoti, cosi' da poterli stampare a schermo come stringhe uniche.

Aggiunta utente:

```
INSERT INTO STUDENT(F_NAME, L_NAME, STREET, NUM, CITY, PS, PHONE_N)
VALUES (‘$F_NAME’, ‘$L_NAME’, ’$STREET’, ’$NUM’, ’$CITY’, ’$PS’, ‘$PHONE_N’);
```

Per la aggiunta utente, utilizziamo il tipo di dato AUTOINCREMENT, cosi' da poter assegnare ad ogni nuovo utente un identificativo differente, questo ci aiuta ad assicurarci che i vincoli sulle chiavi siano rispettati.

Modifica utente:

```
$sql="";
if(!empty($STREET)) {
    $sql.= "UPDATE STUDENT
    SET STREET = '$STREET'
    WHERE STUDENT_N = '$STUDENT_N';";
```

```
}
if(!empty($CITY) {
    $sql.= "UPDATE STUDENT
    SET CITY = '$CITY'
    WHERE STUDENT_N = '$STUDENT_N';";
}
if(!empty($NUM)) {
    $sql.= "UPDATE STUDENT
    SET NUM = '$NUM'
    WHERE STUDENT_N = '$STUDENT_N';";
}
if(!empty($PS)) {
    $sql.= "UPDATE STUDENT
    SET PS = '$PS'
    WHERE STUDENT_N = '$STUDENT_N';";
}
if (!empty($PHONE_N)) {
    $sql.= "UPDATE STUDENT
    SET PHONE_N = '$PHONE_N'
    WHERE STUDENT_N = '$STUDENT_N';";
}
if (!empty($F_NAME)){
    $sql.= "UPDATE STUDENT
    SET F_NAME = '$F_NAME'
    WHERE STUDENT_N = '$STUDENT_N';";
}
if (!empty($L_NAME)) {
    $sql.= "UPDATE STUDENT
    SET L_NAME = '$L_NAME'
    WHERE STUDENT_N = '$STUDENT_N';";
}
```

La modifica utente sfrutta la possibilità' di concatenare più interrogazioni MySQL usando l'operatore .=, andando a concatenare soltanto le interrogazioni di aggiornamento per cui ci viene fornito qualcosa. Questo lascia all'utente finale più libertà di andare ad aggiornare il minimo indispensabile.

Cancellazione utente:

```
DELETE FROM STUDENT WHERE STUDENT_N = '$STUDENT_N';
```

Aggiunta prestito:

```
$sql = "SELECT MAX(f.ID_COPY) INTO @copy FROM FOUND_IN AS f
LEFT JOIN RENTED_BY AS r ON f.ID_BOOK = r.ISBN AND
f.ID_COPY = r.ID_COPY AND f.LIBRARY_ID = r.LIBRARY_ID
WHERE r.ID_COPY IS NULL AND f.ID_BOOK = '$ISBN'
AND f.LIBRARY_ID = '$LIBRARY_ID';";
$sql.= "SELECT DATE_ADD('$STARTDATE', INTERVAL 30 DAY) INTO @date;";
$sql.= "INSERT INTO RENTED_BY VALUES
('$STUDENT_N', @copy, '$ISBN', '$LIBRARY_ID', '$STARTDATE', @date);";
```

```
$sql.= "SELECT * FROM RENTED_BY WHERE
(USER_ID = '$STUDENT_N' AND ID_COPY = @copy
AND ISBN = '$ISBN' AND LIBRARY_ID = '$LIBRARY_ID'
AND STARTDATE = '$STARTDATE'
AND ENDDATE = @date);";
```

Anche in questo caso sfruttiamo la possibilità di poter concatenare interrogazioni e di utilizzare la funzione mysqli_multi_query() per eseguirle in massima correttezza. Cosi' facendo andiamo a trovarci tutti i valori necessari per l'inizio di un nuovo prestito in modo ordinato. Facciamo la SELECT iniziale con la JOIN per andare a trovare se c'è' una copia disponibile del libro che vogliamo prendere in prestito nella nostra libreria di interesse, non dovesse esserci, @copy sarebbe nullo, e l'interrogazione ritornerebbe un errore. Qua vediamo anche che abbiamo deciso 30 giorni essere il tempo per il prestito. Come controllo sull'errore, se non dovesse esserci una copia disponibile del libro scelto nella libreria scelta, sul sito web apparirà un messaggio di errore che spiegherà che il libro non e' presente.

Modifica prestito:

```
UPDATE RENTED_BY SET ENDDATE = '$ENDDATE'
WHERE ISBN = '$ISBN' AND
USER_ID = '$STUDENT_N' AND
ID_COPY = '$ID_COPY' AND
LIBRARY_ID = '$LIBRARY_ID';
```

Per il prestito abbiamo deciso di alterare soltanto la fine di esso, questa scelta e' pensata sulla base del funzionamento solito di una libreria. Dati gli elementi singoli del prestito, si e' soliti al massimo richiedere una proroga su un libro già dato in prestito, se mai si dovesse cambiare libro, di solito si preferisce iniziare un nuovo prestito con due nuove date di inizio e fine.

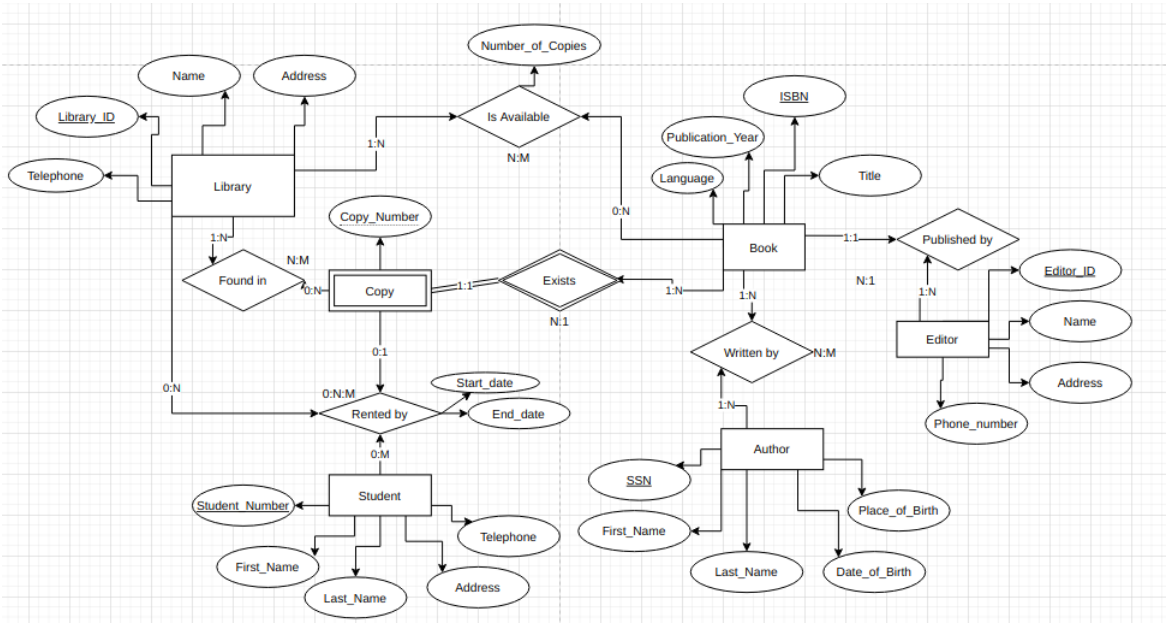
Cancella prestito:

```
DELETE FROM RENTED_BY WHERE
(USER_ID = '$STUDENT_N' AND
ISBN = '$ISBN' AND
ID_COPY = '$ID_COPY' AND
LIBRARY_ID = '$LIBRARY_ID');
```

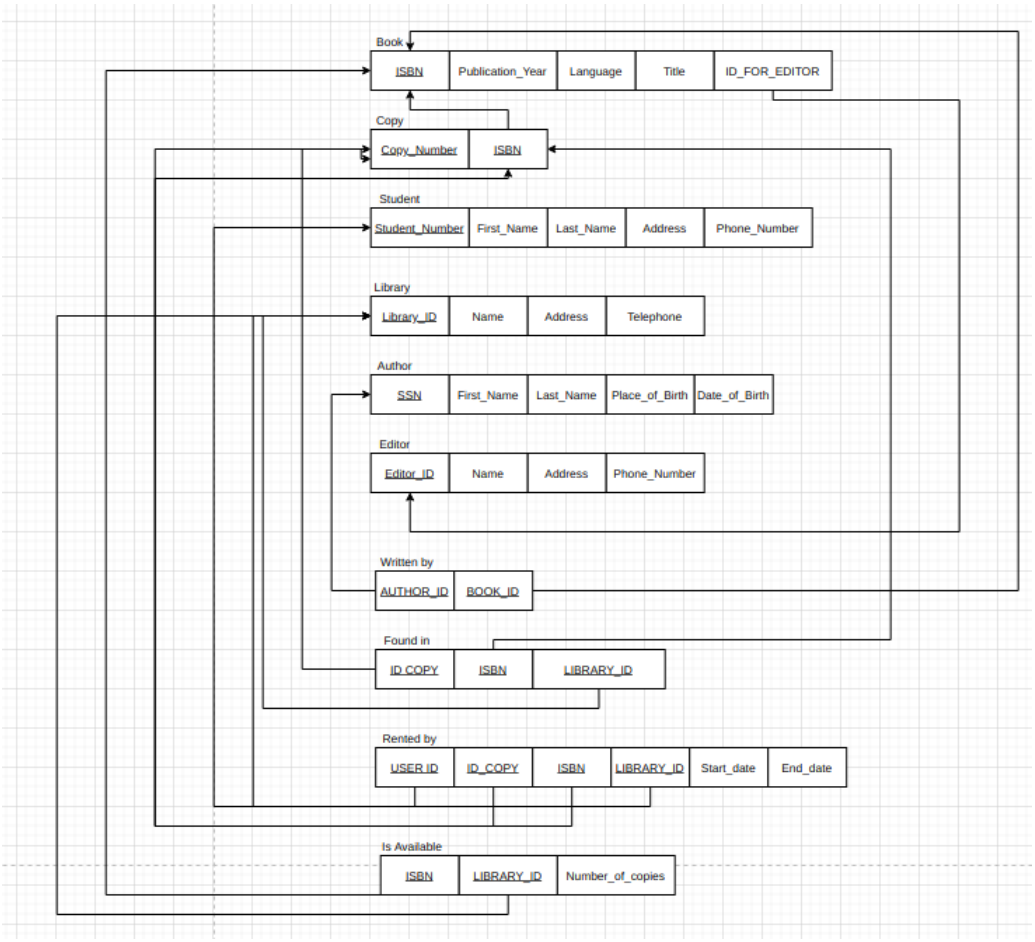
In conclusione queste sono le funzionalità di maggiore importanza per il funzionamento dell'applicativo web che si interfaccia con la base dati.

5 Appendice

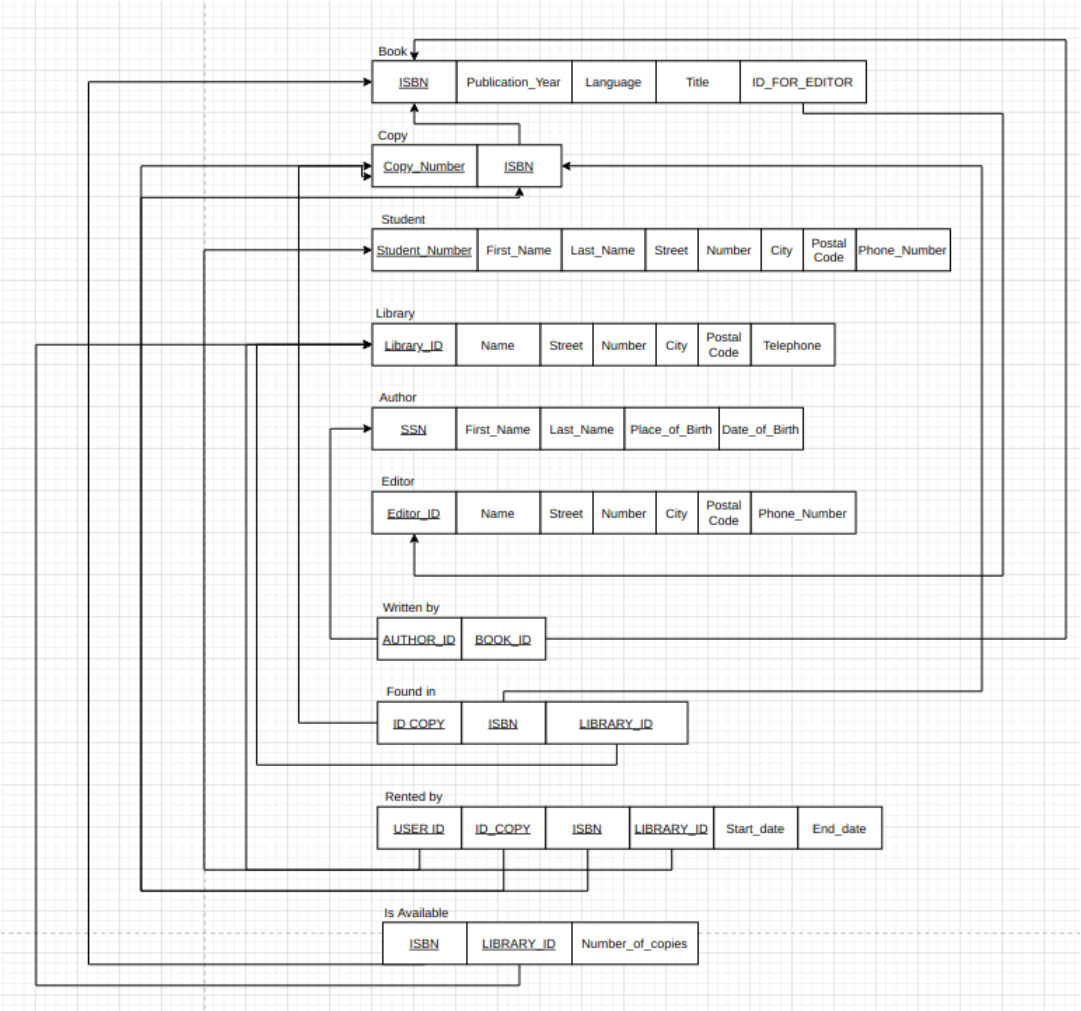
5.1 Appendice elemento 1: diagramma ER



5.2 Appendice elemento 2: diagramma relazionale



5.3 Appendice elemento 3: diagramma relazionale normalizzato



5.4 Appendice elemento 4: spiegazione file .zip

Nel file di consegna del progetto abbiamo tutti i file legati all'applicativo web, in più il file Biblioteca.sql, che dovrebbe bastare per avere una copia identica della nostra base dati. In caso si dovesse avere bisogno dei file .csv per importare i dati di una singola tabella, si possono trovare nella cartella project_data e si possono poi importare. Infine l'identificativo che viene usato per stabilire una connessione con la base dati nel codice e' ada10, con password Juventus.1992.