|  |  |
| --- | --- |
|  |  |

Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

SISTEMA INFORMATIVO DI UN CIRCUITO DI BIBLIOTECHE

Matricola: 0269082

Matteo Fanfarillo

**Indice**

[1. Descrizione del Minimondo 2](#_Toc606296459)

[2. Analisi dei Requisiti 4](#_Toc1289394997)

[3. Progettazione concettuale 9](#_Toc2081466291)

[4. Progettazione logica 25](#_Toc2147004904)

[5. Progettazione fisica 50](#_Toc518560220)

[Appendice: Implementazione 122](#_Toc403811585)

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | Un circuito di biblioteche mette a disposizione un servizio di prenotazione libri. Si vuole realizzare un sistema informativo di gestione per tali biblioteche, ciascuna caratterizzata da indirizzo, numero di telefono, nome del responsabile ed orario di apertura settimanale.  Il sistema gestisce libri, le cui copie sono disponibili in un sottoinsieme delle biblioteche, in numero differente. Ciascun libro è associato allo stato “in prestito” / “disponibile”. Inoltre, il sistema permette agli amministratori di gestire i turni di lavoro, fino ad un massimo di 8 ore, dei bibliotecari che operano nelle biblioteche del circuito. I turni sono realizzati su base mensile. Un report ad hoc consente agli amministratori di sapere se alcune delle biblioteche del circuito sono scoperte. Nel caso in cui un bibliotecario faccia richiesta di malattia, il gestore del sistema deve indicare che quel turno non è stato svolto per tale motivo e identificare un bibliotecario sostituto. Di ciascun bibliotecario è di interesse il codice fiscale, il nome, il cognome, la data di nascita, il luogo di nascita, il titolo di studio.  Gli utenti del circuito di biblioteche possono registrarsi fornendo tutte le loro informazioni anagrafiche ed un numero arbitrario di contatti (telefono, cellulare, email), specificando qual è il mezzo di comunicazione preferito con cui vogliono essere contattati. All’atto di effettuare un prestito, i bibliotecari possono recuperare la disponibilità delle copie del libro presso la biblioteca in cui essi lavorano. Se il libro è disponibile, il sistema restituisce lo scaffale ed il ripiano in cui può essere prelevato il libro. Se non ne è disponibile alcuna copia, il bibliotecario può verificare in quali altre biblioteche del circuito è presente tale libro ed effettuare una richiesta di trasferimento. In questo caso, il libro viene segnalato come “prestato ad altra biblioteca” e viene tenuto traccia di qual è la biblioteca di partenza e di destinazione. All’atto della consegna della copia, l’utente del servizio può chiedere di trattenere in consultazione il libro per 1, 2, o 3 mesi.  I bibliotecari hanno la possibilità di generare un report indicante quali libri, prestati dalla biblioteca in cui stanno svolgendo il turno, non sono ancora stati restituiti e le informazioni dell’utente che possiede attualmente la copia del libro. I recapiti consentono al bibliotecario di mettersi in contatto per sollecitare la restituzione.  Il sistema calcola automaticamente la penale da versare per un prestito riconsegnato in ritardo, ammontante a 0.10 € per ciascun giorno di ritardo (fino ad un massimo di 10 giorni), 0.50€ per ciascun giorno di ritardo superiore al decimo. Questa tariffa viene richiesta al cliente (e registrata nel sistema) alla riconsegna del titolo.   Un libro che non è stato prestato nei passati 10 anni viene dismesso dalla biblioteca. Tale operazione viene effettuata dagli amministratori del circuito. Il record ad esso associato non viene eliminato, ma questo non potrà più essere prestato agli utenti. I prestiti in corso restano validi fino alla riconsegna. |

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

|  |  |  |  |
| --- | --- | --- | --- |
| **Linea** | **Termine** | **Nuovo termine** | **Motivo correzione** |
| 6 | Libro | Copia | Si ha un’omonimia col termine “libro”: in riga 5 è inteso come “pubblicazione”, mentre in altri punti del testo (come in riga 6) è inteso come “copia”, “oggetto materiale”. |
| 11 | Gestore del sistema | Amministratore | Anche “gestore del sistema” e “amministratore” sono sinonimi, per cui è bene utilizzare un unico termine. |
| 15-16 | Informazioni anagrafiche | Codice fiscale, nome, cognome, data di nascita e indirizzo di residenza | “Informazioni anagrafiche” è un’espressione troppo generica che potrebbe dare luogo ad ambiguità, per cui è bene sostituirla con un elenco completo delle informazioni che dovranno caratterizzare gli utenti. |
| 16 | Contatto | Recapito | “Contatto” e “recapito” sono sinonimi, per cui è bene utilizzare un unico termine. |
| 19 | Libro | Copia | Qui “libro” è inteso come “copia”. |
| 20 | Libro | Copia | Qui “libro” è inteso come “copia”. |
| 22 | Presente | Disponibile | “Presente” e “disponibile sono sinonimi, per cui è bene utilizzare un unico termine. |
| 22 | Libro | Copia | Qui “libro” è inteso come “copia”. |
| 23 | Prestato ad altra biblioteca | Trasferito | Il prestito a un’altra biblioteca e il trasferimento sono lo stesso concetto; si preferiscono i termini “trasferimento” e “trasferito” per evitare di confondere il prestito di una copia di un libro tra biblioteche col prestito di una copia di un libro a un utente. |
| 24 | Consegna | Prestito | “Consegna” e “prestito” sono sinonimi, per cui è bene utilizzare un unico termine. |
| 25 | Consultazione | Prestito | La consultazione di una copia da parte di un’utente avviene nell’ambito di un prestito: si può assumere che i termini “consultazione” e “prestito” si riferiscano al medesimo concetto e siano quindi sinonimi. |
| 25 | Libro | Copia | Qui “libro” è inteso come “copia”. |
| 27 | Libri | Copie | Qui “libri” è inteso come “copie”. |
| 31 | Prestito | Copia in prestito | Questa sostituzione serve principalmente a rendere ancora più chiara la specifica. |
| 31 | Riconsegnato | Restituito | La riconsegna di una copia e la sua restituzione sono lo stesso concetto, per cui i termini “riconsegnato” e “restituito” possono essere considerati sinonimi. |
| 33 | Tariffa | Penale | “Tariffa” e “penale” indicano lo stesso concetto, per cui possono essere considerati sinonimi. |
| 34 | Cliente | Utente | “Cliente” e “utente” indicano lo stesso concetto, per cui possono essere considerati sinonimi. |
| 34 | Riconsegna | Restituzione | “Riconsegna” e “restituzione” sono sinonimi. |
| 34 | Titolo | Copia | “Titolo” e “copia” indicano lo stesso concetto, per cui possono essere considerati sinonimi. |
| 39 | Riconsegna | Restituzione | “Riconsegna” e “restituzione” sono sinonimi. |

### Specifica disambiguata

|  |
| --- |
| Si vuole realizzare un sistema informativo di gestione per un circuito di biblioteche, ciascuna delle quali è caratterizzata da indirizzo, numero di telefono, nome del responsabile e orario di apertura settimanale.   Il sistema gestisce libri, le cui copie sono disponibili in un sottoinsieme delle biblioteche in numero differente. Ciascuna copia è associata allo stato “in prestito” / “disponibile”. Inoltre, il sistema permette agli amministratori di gestire i turni di lavoro, fino a un massimo di 8 ore, dei bibliotecari che operano nelle biblioteche. I turni sono realizzati su base mensile. Un report ad hoc consente agli amministratori di sapere se alcune biblioteche sono scoperte. Nel caso in cui un bibliotecario faccia richiesta di malattia, l’amministratore deve indicare il turno che non è stato svolto e il motivo, e deve identificare un bibliotecario sostituto. Di ciascun bibliotecario sono di interesse il codice fiscale, il nome, il cognome, la data di nascita, il luogo di nascita e il titolo di studio.  Gli utenti del circuito di biblioteche possono registrarsi fornendo codice fiscale, nome, cognome, data di nascita, indirizzo di residenza e un numero arbitrario di recapiti (telefono, cellulare, email), specificando qual è il mezzo di comunicazione preferito con cui vogliono essere contattati. All’atto di effettuare un prestito, i bibliotecari possono recuperare la disponibilità delle copie del libro presso la biblioteca in cui essi lavorano. Se almeno una copia è disponibile, il sistema restituisce lo scaffale e il ripiano in cui tale copia può essere prelevata. In caso contrario, il bibliotecario può verificare in quali altre biblioteche la copia è disponibile ed effettuare una richiesta di trasferimento. In questo caso, la copia viene segnalata come “trasferita” e viene tenuto traccia di quali sono le biblioteche di partenza e di destinazione. L’utente può chiedere di trattenere in prestito la copia per un mese, due mesi, oppure tre mesi.   I bibliotecari hanno la possibilità di generare un report indicante quali copie, prestate dalla biblioteca in cui stanno svolgendo il turno, non sono ancora state restituite, e le informazioni dell’utente che possiede attualmente la copia del libro. I recapiti consentono al bibliotecario di mettersi in contatto con l’utente per sollecitare la restituzione. Il sistema calcola automaticamente la penale da versare per una copia in prestito restituita in ritardo, ammontante a 0.10€ per ciascun giorno di ritardo (fino a un massimo di 10 giorni), 0.50€ per ciascun giorno di ritardo superiore al decimo. La penale viene richiesta all’utente (e registrata nel sistema) all’atto della restituzione della copia.   Un libro di cui non è stata prestata alcuna copia nei passati 10 anni viene dismesso dalla biblioteca. Tale operazione viene effettuata dagli amministratori. Il record associato a questo libro non viene eliminato ma le relative copie non potranno più essere prestate agli utenti. I prestiti in corso restano validi fino alla restituzione della copia. |

## Glossario dei Termini

|  |  |  |  |
| --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Biblioteca | Biblioteca appartenente al circuito di biblioteche |  | Bibliotecario, copia, prestito, trasferimento |
| Bibliotecario | Persona che lavora in una biblioteca |  | Biblioteca, malattia, turno |
| Copia | Copia di un libro presente in una biblioteca. Può essere prestata a un utente. | Libro, titolo | Biblioteca, libro, prestito, trasferimento, utente |
| Libro | Pubblicazione presente in una o più biblioteche. Ciascuna biblioteca ha zero o più copie dello stesso libro. |  | Copia |
| Malattia | Richiesta da parte di un bibliotecario di saltare un turno di lavoro |  | Bibliotecario, turno |
| Penale | Tariffa che un utente deve pagare nel caso in cui restituisca in ritardo una copia che aveva preso in prestito. | Tariffa | Prestito, utente |
| Prestito | Prestito fatto a un utente; ogni prestito si riferisce a una copia di un libro. | Consegna, consultazione | Biblioteca, copia, penale, utente |
| Trasferimento | Prestito fatto a un’altra biblioteca; ogni trasferimento si riferisce a una copia di un libro. | Prestito ad altra biblioteca | Biblioteca, copia |
| Turno | Turno di lavoro di un bibliotecario; può essere di al massimo 8 ore. |  | Bibliotecario, malattia |
| Utente | Persona che prende in prestito copie di libri da una biblioteca | Cliente | Copia, penale, prestito |

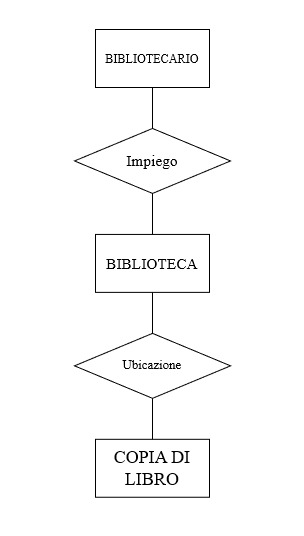
## Raggruppamento dei requisiti in insiemi omogenei

|  |
| --- |
| **Frasi di carattere generale** |
| Si vuole realizzare un sistema informativo di gestione per un circuito di biblioteche. |
| **Frasi relative alle biblioteche** |
| Per le biblioteche rappresentiamo l’indirizzo, il numero di telefono, il nome del responsabile e l’orario di apertura settimanale. |
| **Frasi relative ai bibliotecari** |
| Per i bibliotecari rappresentiamo il codice fiscale, il nome, il cognome, la data di nascita, il luogo di nascita e il titolo di studio. |
| **Frasi relative ai libri e alle copie dei libri** |
| Il sistema gestisce libri, le cui copie sono disponibili in un sottoinsieme delle biblioteche in numero differente. Ciascuna copia è associata allo stato “in prestito” / “disponibile”. All’atto di effettuare un prestito, i bibliotecari possono recuperare la disponibilità delle copie del libro presso la biblioteca in cui essi lavorano. Se almeno una copia è disponibile, il sistema restituisce lo scaffale e il ripiano in cui tale copia può essere prelevata. In caso contrario, il bibliotecario può verificare in quali biblioteche la copia è disponibile ed effettuare una richiesta di trasferimento. Un libro di cui non è stata prestata alcuna copia nei passati 10 anni viene dismesso dalla biblioteca. Il record associato a questo libro non viene eliminato ma le relative copie non potranno più essere prestate agli utenti. |
| **Frasi relative ai turni** |
| Il sistema permette agli amministratori di gestire i turni di lavoro, fino a un massimo di 8 ore, dei bibliotecari che operano nelle biblioteche. I turni sono realizzati su base mensile. |
| **Frasi relative alle richieste di malattia** |
| Nel caso in cui un bibliotecario faccia richiesta di malattia, l’amministratore deve indicare il turno che non è stato svolto e il motivo, e deve identificare un bibliotecario sostituto. |
| **Frasi relative agli utenti** |
| Per gli utenti del circuito di biblioteche rappresentiamo il codice fiscale, il nome, il cognome, la data di nascita, l’indirizzo di residenza, un numero arbitrario di recapiti (telefono, cellulare, email) e il mezzo di comunicazione preferito con cui gli utenti vogliono essere contattati. |
| **Frasi relative ai prestiti** |
| L’utente può chiedere di trattenere in prestito una copia per un mese, due mesi, oppure tre mesi. I bibliotecari hanno la possibilità di generare un report indicante quali copie, prestate dalla biblioteca in cui stanno svolgendo il turno, non sono ancora state restituite, e le informazioni dell’utente che possiede attualmente la copia del libro. I prestiti in corso restano validi fino alla restituzione della copia. |
| **Frasi relative ai trasferimenti** |
| All’atto di effettuare un prestito, se nella biblioteca non è disponibile alcuna copia, il bibliotecario può verificare in quali biblioteche la copia è disponibile ed effettuare una richiesta di trasferimento. In questo caso, la copia viene segnalata come “trasferita” e viene tenuto traccia di quali sono le biblioteche di partenza e di destinazione. |
| **Frasi relative alle penali** |
| Il sistema calcola automaticamente la penale da versare per una copia in prestito restituita in ritardo, ammontante a 0.10€ per ciascun giorno di ritardo (fino a un massimo di 10 giorni), 0.50€ per ciascun giorno di ritardo superiore al decimo. La penale viene richiesta all’utente (e registrata nel sistema) all’atto della restituzione della copia. |

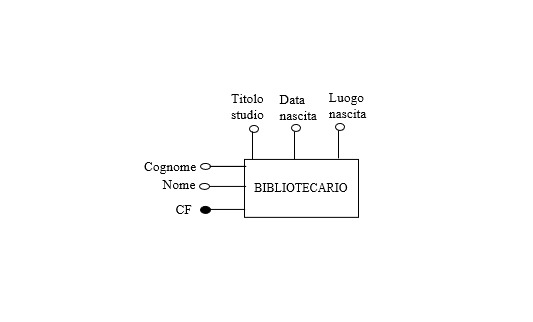
# Progettazione concettuale

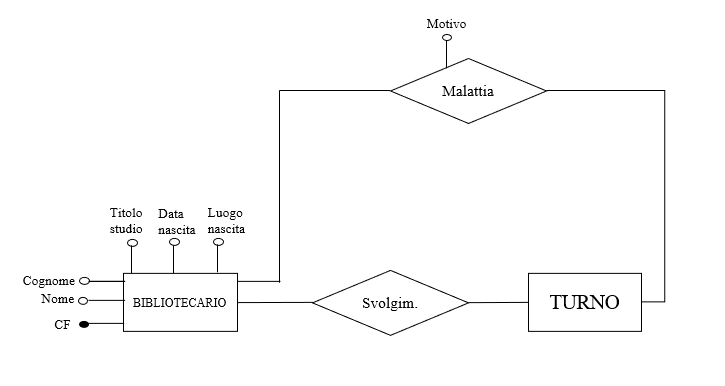
## Costruzione dello schema E-R

I tre concetti principali del mini-mondo di interesse sono le copie dei libri, le biblioteche e i bibliotecari, che possono essere rappresentati come entità dello schema scheletro di partenza. In una prima analisi, è possibile supporre che esista un’associazione *impiego* che lega i bibliotecari con le rispettive biblioteche in cui lavorano, e un’associazione *ubicazione* tra le copie dei libri e le rispettive biblioteche che le possiedono.



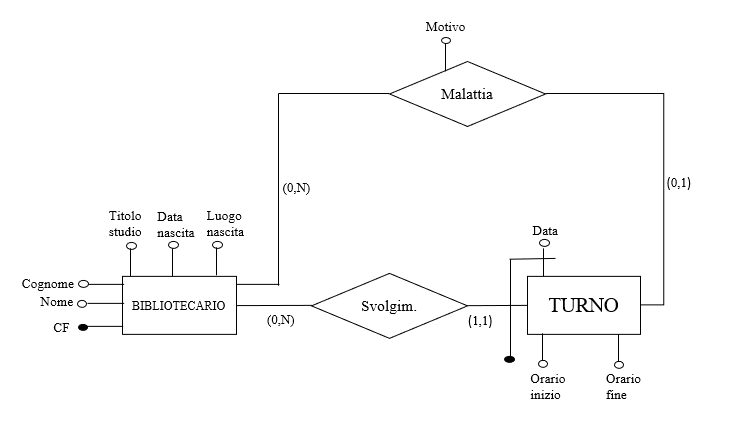
Analizziamo dapprima le specifiche che coinvolgono i bibliotecari. Possiamo innanzitutto osservare che all’entità *bibliotecario* possono essere assegnati gli attributi *codice fiscale* (che farà da identificatore), *nome*, *cognome*, *data di nascita*, *luogo di nascita* e *titolo di studio*.



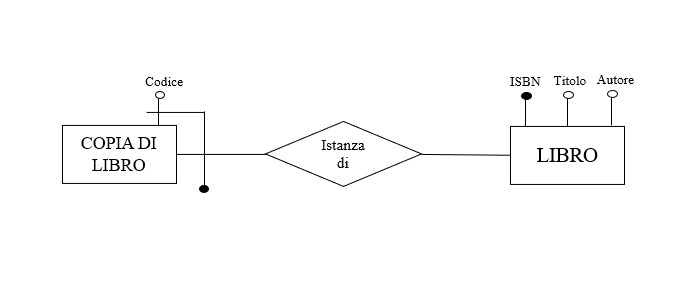
I bibliotecari lavorano seguendo determinati turni di lavoro, per cui deve essere introdotta l’entità *turno*, che sarà relazionata con l’entità *bibliotecario* mediante l’associazione *svolgimento*. Alcuni turni di lavoro potrebbero essere non svolti a seguito di una richiesta di malattia da parte di un bibliotecario; per le richieste di malattia bisogna tener traccia, oltre al turno non svolto, anche il relativo motivo e il bibliotecario **sostituto**, per cui è opportuno introdurre un’associazione denominata *malattia*, che avrà *motivo* come attributo e legherà l’entità *turno* con l’entità *bibliotecario.* Tale relazione permetterà appunto di determinare qual è il bibliotecario **sostituto**. 

È facile intuire che i turni di lavoro dei bibliotecari possono essere caratterizzati dall’orario di inizio e dall’orario di fine. Poiché le specifiche sottolineano che i turni sono organizzati su base mensile, si può intendere che i turni non siano univocamente definiti dal giorno della settimana, per cui è opportuno aggiungere piuttosto l’attributo *data* che, insieme all’entità *bibliotecario*, è in grado di identificare in modo univoco il turno.

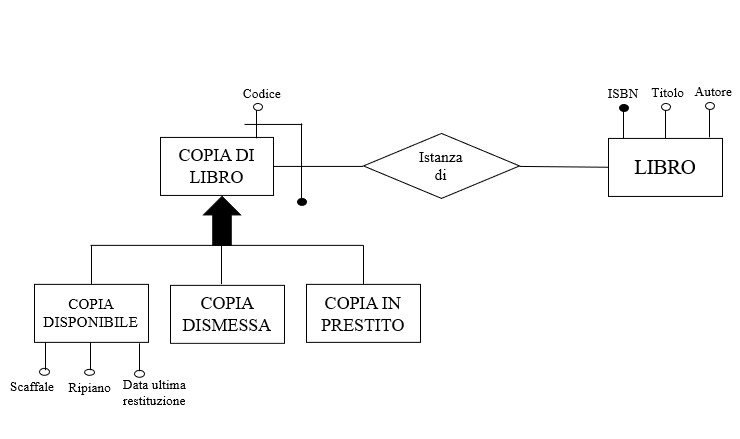
A valle di queste considerazioni e dopo aver aggiunto le cardinalità opportune alle associazioni, si ottiene il seguente sotto-diagramma E-R:



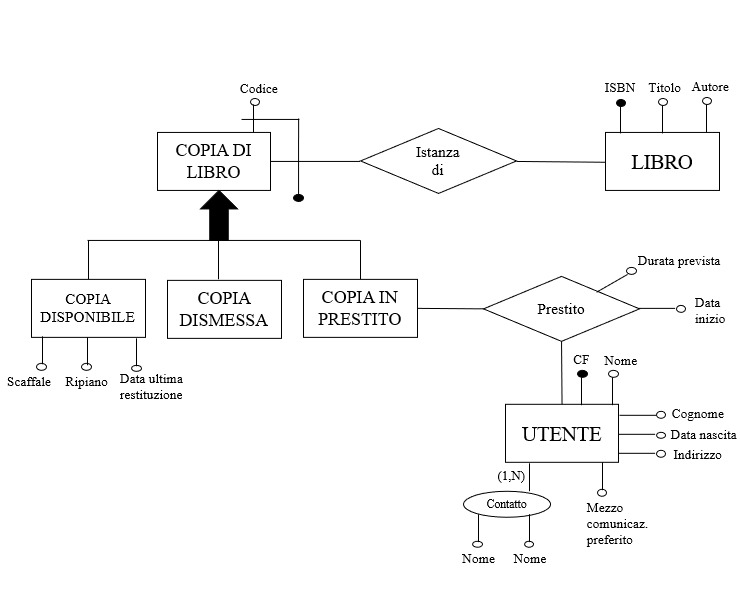
Analizziamo ora le specifiche che coinvolgono le copie dei libri. Possiamo subito osservare che le copie sono delle istanze dell’entità *libro*, la quale indica un concetto più astratto (ovvero le pubblicazioni). Sebbene le specifiche non ne parlino esplicitamente, è utile definire degli attributi per i libri e le relative copie. Ad esempio, può risultare comodo caratterizzare un libro attraverso il codice ISBN (che ha il compito di identificare il libro), il titolo e l’autore. D’altra parte, un modo intuitivo per identificare la singola copia di un libro non c’è, per cui le attribuiremo come attributo un indice numerico (un codice) che sarà univoco nell’ambito del libro.



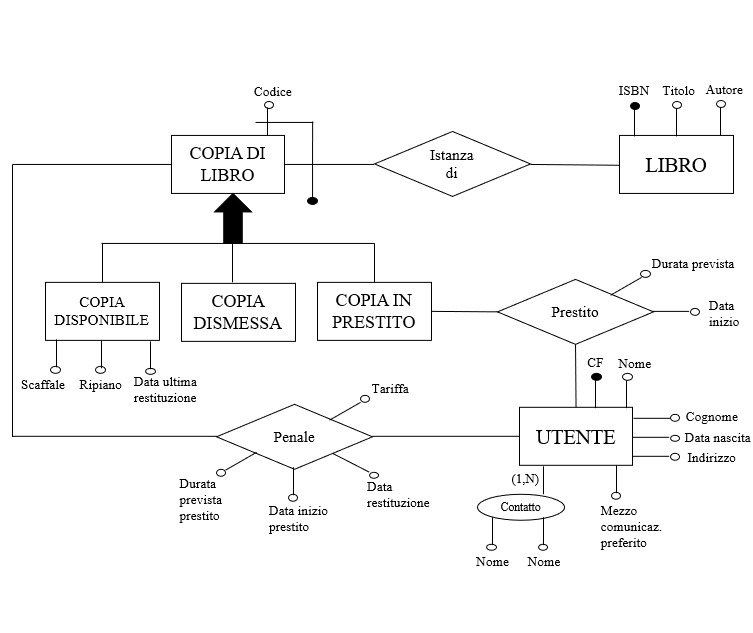
Le copie dei libri sono inoltre caratterizzate dallo stato “in prestito” / “disponibile”. Pocihé i libri disponibili hanno come ulteriori attributi lo scaffale e il ripiano, conviene considerare *copia in prestito* e *copia disponibile* come entità figlie di *copia di libro*. In realtà, dalle ultime righe delle specifiche emerge un terzo stato per le copie dei libri e, quindi, una terza entità figlia, che sarebbe *copia dismessa*. Affinché una copia di libro all’interno di una particolare biblioteca entri nello stato “dismessa”, è necessario che tutte le copie di quello stesso libro all’interno di quella stessa biblioteca siano attualmente disponibili e siano state restituite per l’ultima volta da un prestito almeno dieci anni fa. Per questo motivo, è opportuno tenere traccia anche della data dell’ultima restituzione per le copie disponibili.



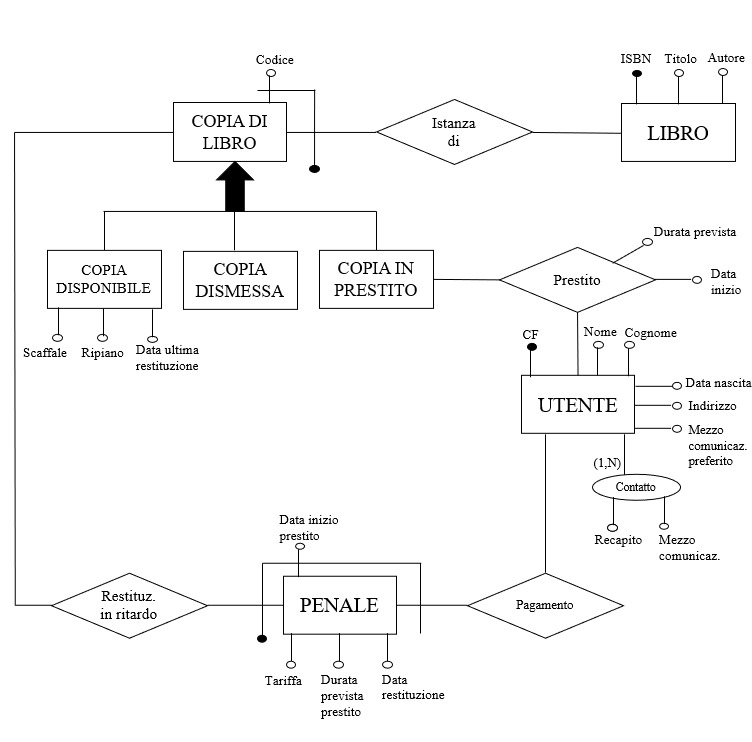
Poiché è prevista la possibilità di risalire all’utente che attualmente ha una determinata copia di libro in prestito, è necessario introdurre l’entità *utente* che sarà relazionata con l’entità *copia in prestito* mediante l’associazione *prestito*. Gli utenti sono caratterizzati dagli attributi *codice fiscale* (che farà da identificatore), *nome*, *cognome*, *data di nascita* e *indirizzo*. A questi verranno aggiunti anche l’attributo multivalore composto *contatto* (di cui si specificano il recapito e il relativo mezzo di comunicazione) e l’attributo *mezzo di comunicazione preferito*. Poiché per ogni prestito in corso si deve conoscere la durata pattuita e si deve verificare se la scadenza verrà rispettata o meno, bisogna aggiungere all’associazione *prestito* gli attributi *durata prevista* e *data di inizio*.



Le specifiche prevedono anche che saranno registrate le penali che gli utenti dovranno pagare dopo aver restituito una copia in ritardo. Per questo motivo, per un primo momento, aggiungeremo un’associazione che lega l’entità *copia di libro* con l’entità *utente*: *penale*. Questa associazione sarà caratterizzata dall’attributo *tariffa*, che indica l’ammontare della penale, ma anche dagli attributi *durata prevista del prestito*, *data di inizio del prestito* e *data di restituzione*, che permettono di calcolare il valore di *tariffa*.



Tuttavia, appare evidente che la costruzione appena descitta sia proibitiva: infatti, non permette di registrare più di una penale relativa a prestiti che hanno coinvolto la stessa copia di libro e il medesimo utente. Per questo motivo, è preferibile applicare il pattern della reificazione e trasformare così l’associazione *penale* in un’entità, che identificheremo mediante la data di inizio del prestito, l’utente e la copia di libro coinvolti.



Talvolta, una biblioteca, per effettuare un prestito, può aver bisogno di richiedere a un’altra biblioteca il trasferimento di una copia del libro da prestare. È comunque possibile assumere che ogni copia appartenga sempre alla medesima biblioteca, per cui, nel momento in cui un prestito si conclude, è necessario restituire la copia alla biblioteca da cui è partito l’eventuale trasferimento. Per questo motivo, introduciamo una nuova entità, *copia trasferita*, e consideriamo l’insieme delle copie trasferite come un sottoinsieme delle copie prestate.

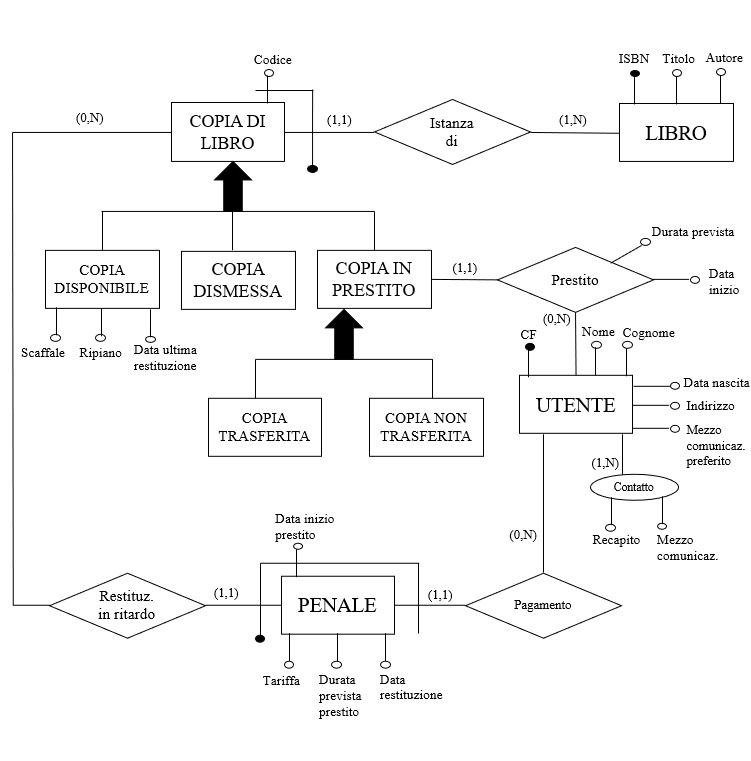
Osserviamo anche che i bibliotecari dovranno poter tenere traccia di tutte le copie attualmente prestate dalla biblioteca in cui essi lavorano. Queste copie sono divise in due gruppi:

- Le copie appartenenti alla biblioteca in questione e direttamente prestate agli utenti: si tratta di copie prestate ma non trasferite.

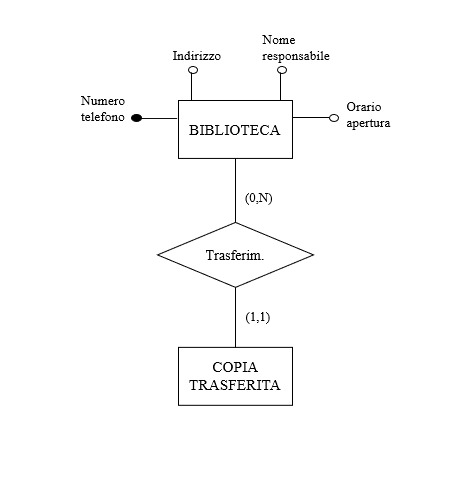
- Le copie appartenenti a un’altra biblioteca, trasferite alla biblioteca in questione e poi prestate agli utenti; si tratta di copie prestate e trasferite.

Da qui nasce la necessità di conoscere quali sono le copie prestate ma non trasferite, per cui è opportuno introdurre una seconda generalizzazione totale all’entità *copia in prestito*, che avrà così due entità figlie: *copia trasferita* e *copia non trasferita*.

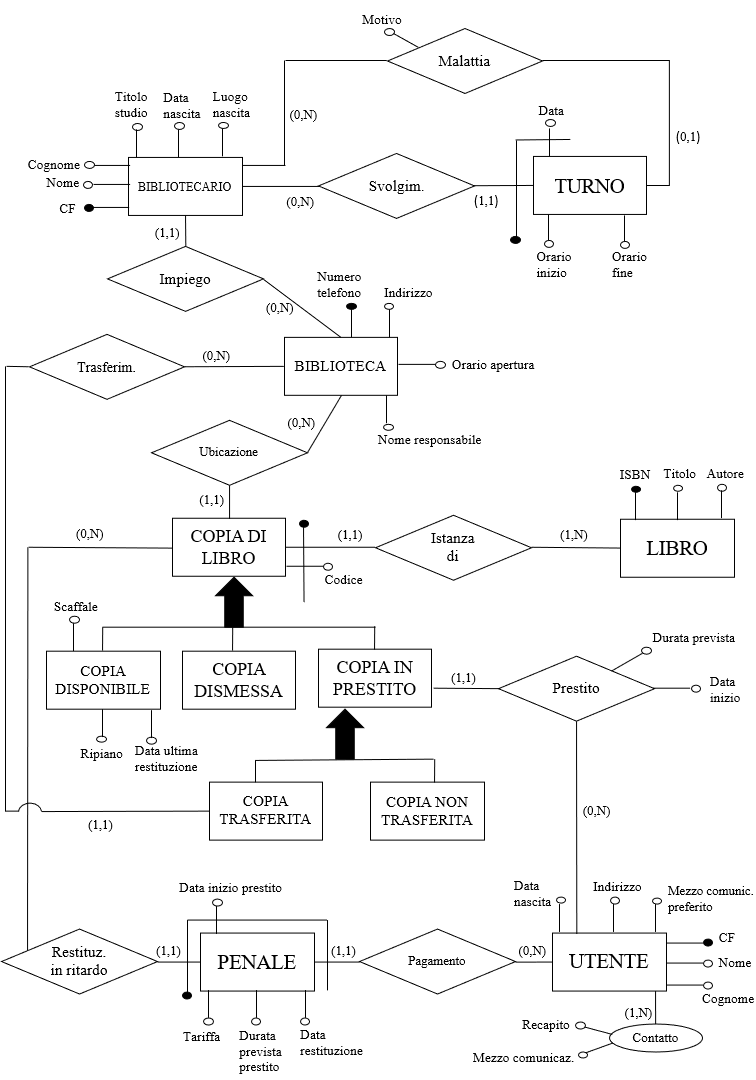
A valle di queste considerazioni e dopo aver aggiunto le cardinalità opportune alle associazioni, si ottiene il seguente sotto-diagramma E-R:



A proprosito dei trasferimenti, si vuole tenere traccia della biblioteca di partenza e quella di destinazione; da una parte, all’interno del diagramma E-R, siamo già in grado di individuare la biblioteca di partenza, dato che è relazionata all’entità *copia di libro* mediante l’associazione *ubicazione*. Per poter catturare anche la biblioteca di destinazione, invece, è necessario introdurre un’associazione chiamata *trasferimento* che lega l’entità *copia trasferita* con l’entità *biblioteca*.  
Inoltre, dobbiamo ancora aggiungere gli attributi dell’entità biblioteca che sono: numero di telefono (che farà da identificatore), indirizzo, nome del responsabile e orario di apertura.



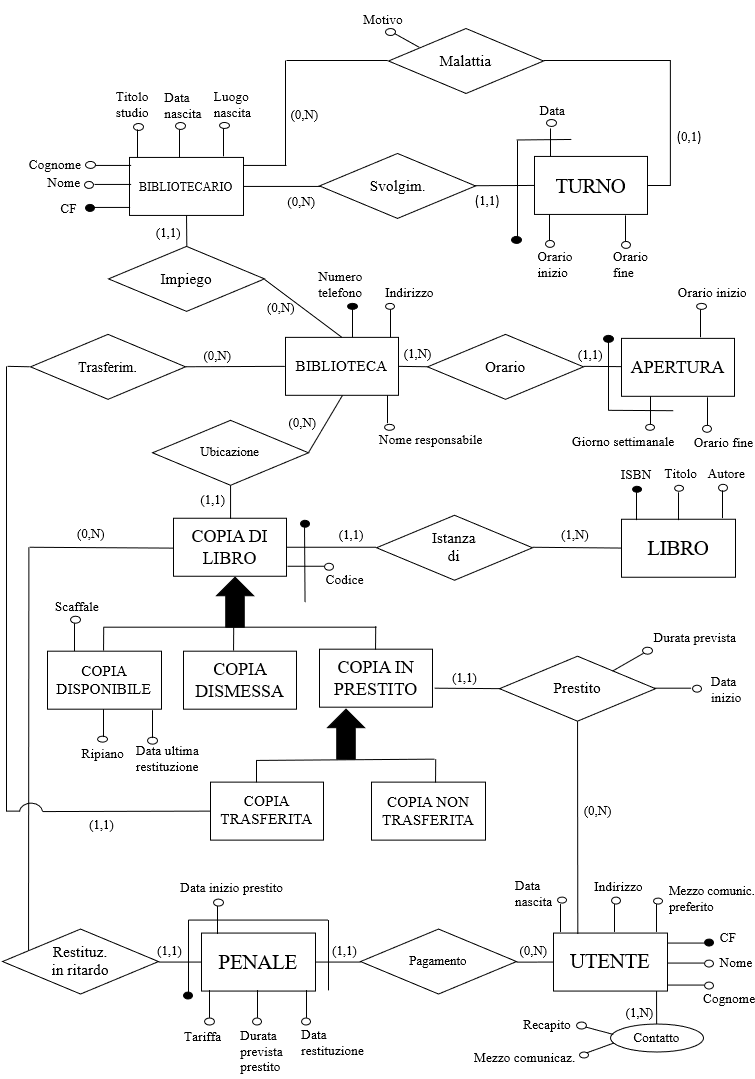
Ora, integrando assieme gli schemi ottenuti fino a questo punto, possiamo ottenere una prima versione del diagramma E-R completo:



### Integrazione finale

Il diagramma E-R così ottenuto non presenta entità o associazioni con lo stesso nome, né conflitti strutturali in senso stretto. Tuttavia, possiamo notare che esistono due concetti del tutto analoghi che sono stati rappresentati in modo differente all’interno del diagramma: da una parte si ha l’entità *turno* che indica gli orari di lavoro dei bibliotecari; dall’altra si ha l’attributo *orario di apertura* (appartenente all’entità biblioteca) che, similmente, indica gli orari di apertura delle biblioteche. Poiché questi due concetti dovranno essere confrontati tra loro e sarà quindi necessario avere una descizione dettagliata degli orari di apertura delle biblioteche, risulta conveniente reificare l’attributo *orario di apertura* dell’entità *biblioteca*, ottenendo così una nuova entità (*apertura)*, che sarà relazionata all’entità *biblioteca* tramite l’associazione uno-a-molti *orario*. Questa entità, per essere caratterizzata in modo esaustivo, ha bisogno degli attributi *giorno settimanale*, *orario di inizio* e *orario di fine*. Inoltre, è possibile identificarla univocamente tramite la biblioteca e il giorno settimanale di riferimento, dopo aver assunto che ciascuna biblioteca effettui al più un’apertura al giorno (mantenendo così una coerenza col discorso fatto sui turni di lavoro dei bibliotecari).

A valle di queste considerazioni, otteniamo il seguente schema E-R (che, per motivi di spazio, si trova nella pagina seguente):



## Regole aziendali

**Regole di vincolo**

(RV01) I turni svolti dai bibliotecari devono durare al più 8 ore.

(RV02) L’orario di fine dei turni svolti dai bibliotecari deve venire dopo il relativo orario di inizio.

(RV03) L’orario di fine dell’apertura delle biblioteche deve venire dopo il relativo orario di inizio.

(RV04) L’intervallo temporale compreso tra l’orario di inizio e l’orario di fine del turno svolto da un bibliotecario deve essere contenuto nell’intervallo temporale compreso tra l’orario di inizio e l’orario di fine dell’apertura della biblioteca in cui il bibliotecario ha impiego.

(RV05) Il bibliotecario supplente deve avere impiego nella medesima biblioteca in cui ha impiego il bibliotecario che ha effettuato una richiesta di malattia.

(RV06) Un bibliotecario non deve effettuare una supplenza mentre è già impegnato a svolgere un suo turno di lavoro.

(RV07) Il mezzo di comunicazione preferito di un utente deve essere presente tra i contatti di tale utente.

(RV08) Gli attributi *mezzo di comunicazione* e *mezzo di comunicazione preferito* dell’entità *utente* devono assumere uno dei seguenti tre valori: *telefono*, *cellulare*, *email*.

(RV09) L’attributo *durata prevista* dell’associazione *prestito* e l’attributo *durata prevista prestito* dell’entità *penale* devono assumere uno dei seguenti tre valori: *1 mese*, *2 mesi*, *3 mesi*.

(RV10) La tariffa di una penale deve assumere un valore strettamente maggiore di zero.

**Regole di derivazione**

(RD01) La durata di un turno si ottiene sottraendo il relativo orario di fine con il relativo orario di inizio.

(RD02) La tariffa di una penale si ottiene:

- Moltiplicando 0,10€ per il numero di giorni di ritardo se il numero di giorni di ritardo è minore o uguale a 10.

- Moltiplicando 0,50€ al numero di giorni di ritardo sottratto di 10, e sommando al risultato così ottenuto 1€ se il numero di giorni di ritardo è maggiore di 10.

(RD03) La data dell’ultima restituzione di un libro all’interno di una biblioteca si ottiene selezionando la più recente data dell’ultima restituzione delle copie di tale libro ubicate in quella stessa biblioteca.

## Dizionario dei dati

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatori** |
| Apertura | Apertura delle biblioteche. | Giorno settimanale, Orario inizio, Orario fine | Giorno settimanale, Biblioteca |
| Biblioteca | Biblioteca appartenente al circuito di biblioteche in esame. | Numero di telefono, Indirizzo, Nome responsabile | Numero di telefono |
| Bibliotecario | Bibliotecario che lavora in una biblioteca del circuito. | Codice fiscale, Nome, Cognome, Data di nascita, Luogo di nascita, Titolo di studio | Codice fiscale |
| Copia di libro | Copia concreta di un libro. | Codice | Codice, Libro |
| Copia dismessa | Copia di un libro che non può essere più prestata agli utenti. | Codice | Codice, Libro |
| Copia disponibile | Copia di un libro attualmente presente in una biblioteca; può essere prestata a un utente. | Codice, Data ultima restituzione, Ripiano, Scaffale | Codice, Libro |
| Copia in prestito | Copia di un libro attualmente posseduta da un utente. | Codice | Codice, Libro |
| Copia non trasferita | Copia di un libro attualmente posseduta da un utente; è stata prestata dalla biblioteca di appartenenza. | Codice | Codice, Libro |
| Copia trasferita | Copia di un libro attualmente posseduta da un utente; è stata prestata da una biblioteca diversa di quella di appartenenza a seguito di un trasferimento. | Codice | Codice, Libro |
| Libro | Pubblicazione | ISBN, Autore, Titolo | ISBN |
| Penale | Penale che un utente deve pagare a seguito di una restituzione di una copia di libro avvenuta in ritardo. | Data inizio prestito, Data restituzione, Durata prevista prestito, Tariffa | Data inizio prestito, Copia di libro, Utente |
| Turno | Turno di lavoro svolto da un bibliotecario. | Data, Orario inizio, Orario fine | Data, Bibliotecario |
| Utente | Utente che può richiedere prestiti delle copie di libro alle biblioteche del circuito. | Codice fiscale, Nome, Cognome, Data di nascita, Indirizzo, Mezzo di comunicazione preferito, Contatto (Recapito+Mezzo comunicazione) | Codice fiscale |
| **Associazione** | **Descrizione** | **Entità coinvolte** | **Attributi** |
| Impiego | Associa un bibliotecario alla biblioteca in cui lavora. | Bibliotecario (1,1), Biblioteca (1,N) |  |
| Istanza di | Associa una copia di libro al libro corrispondente. | Copia di libro (1,1), Libro (1,N) |  |
| Malattia | Permesso di malattia richiesto da un bibliotecario. | Turno (0,1), Bibliotecario (0,N) | Motivo |
| Orario | Associa una biblioteca alle relative aperture. | Biblioteca (1,N), Apertura (1,1) |  |
| Pagamento | Associa una penale all’utente che l’ha pagata. | Penale (1,1), Utente (0,N) |  |
| Prestito | Associa una copia di libro in prestito all’utente a cui è stata prestata. | Copia in prestito (1,1), Utente (0,N) | Data inizio, Durata prevista |
| Restituzione in ritardo | Associa una penale alla relativa copia di libro restituita in ritardo. | Penale (1,1), Copia di libro (0,N) |  |
| Svolgimento | Associa un bibliotecario ai relativi turni di lavoro. | Bibliotecario (1,N), Turno (1,1) |  |
| Trasferimento | Associa una copia di libro trasferita alla relativa biblioteca di destinazione. | Copia trasferita (1,1), Biblioteca (0,N) |  |
| Ubicazione | Associa una copia di libro alla biblioteca di appartenenza. | Copia di libro (1,1), Biblioteca (1,N) |  |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| **Concetto nello schema** | **Tipo[[1]](#footnote-1)** | **Volume atteso** |
| Apertura | E | 120 |
| Biblioteca | E | 20 |
| Bibliotecario | E | 200 |
| Copia di libro | E | 1.500.000 |
| Copia dismessa | E | 100.000 |
| Copia disponibile | E | 1.100.000 |
| Copia in prestito | E | 300.000 |
| Copia non trasferita | E | 200.000 |
| Copia trasferita | E | 100.000 |
| Libro | E | 35.000 |
| Penale | E | 250.000 |
| Turno | E | 4500 |
| Utente | E | 600.000 |
| Impiego | R | 200 |
| Istanza di | R | 1.500.000 |
| Malattia | R | 300 |
| Orario | R | 120 |
| Pagamento | R | 250.000 |
| Prestito | R | 300.000 |
| Restituzione in ritardo | R | 250.000 |
| Svolgimento | R | 4500 |
| Trasferimento | R | 100.000 |
| Ubicazione | R | 1.500.000 |

## Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| **Cod.** | **Descrizione** | **Frequenza attesa** |
| Op 1 | Aggiorna i turni dei bibliotecari. | 1 al mese (per 4500 turni differenti) |
| Op 2 | Inserisci un permesso di malattia. | 10 al giorno |
| Op 3 | Trova i bibliotecari di una biblioteca disponibili in un intervallo di tempo. | 10 al giorno |
| Op 4 | Inserisci una nuova penale. | 330 al giorno |
| Op 5 | Trova le informazioni sui turni dei bibliotecari che lavorano in una specifica biblioteca. | 1 al mese |
| Op 6 | Trova le informazioni sulle biblioteche e sulle loro aperture. | 1 al mese |
| Op 7 | Inserisci un nuovo utente. | 200 al giorno |
| Op 8 | Trova le copie disponibili all’interno di una biblioteca, indicando la data della loro ultima restituzione. | 3 al giorno (= 1 alla settimana per ciascuna biblioteca) |
| Op 9 | Contrassegna come “dismessa” una copia disponibile. | 210 alla settimana |
| Op 10 | Inserisci una nuova copia di libro in una biblioteca, contrassegnandola come disponibile. | 400 al giorno |
| Op 11 | Inserisci un nuovo libro. | 10 al giorno |
| Op 12 | Trova le copie disponibili di un particolare libro all’interno di una biblioteca, indicando il loro ripiano e il loro scaffale. | 5000 al giorno |
| Op 13 | Trova le biblioteche che hanno almeno una copia di un determinato libro nello stato “disponibile”. | 1700 al giorno |
| Op 14 | Contrassegna come “in prestito” (ma non trasferita) una copia, indicando i dati dell’utente coinvolto nel prestito. | 3300 al giorno |
| Op 15 | Contrassegna come “in prestito e trasferita” una copia, indicando i dati dell’utente coinvolto nel prestito e la biblioteca di destinazione del trasferimento. | 1700 al giorno |
| Op 16 | Contrassegna nuovamente come “disponibile” una copia in prestito ma non trasferita, eliminando il relativo prestito e aggiornando la data della sua ultima restituzione. | 3300 al giorno |
| Op 17 | Contrassegna nuovamente come “disponibile” una copia in prestito e trasferita, eliminando il relativo prestito e il relativo trasferimento, e aggiornando la data della sua ultima restituzione. | 1700 al giorno |
| Op 18 | Trova le copie contrassegnate come “in prestito” (ma non trasferite) di una biblioteca e i dati dei relativi utenti coinvolti nei prestiti. | 1 al giorno (per ciascuna biblioteca) |
| Op 19 | Trova le copie contrassegnate come “in prestito e trasferite” della biblioteca destinataria del trasferimento e i dati dei relativi utenti coinvolti nei prestiti. | 1 al giorno (per ciascuna biblioteca) |

## Costo delle operazioni

- Op 1: 1 accesso in scrittura nell’entità *turno* + 1 accesso in scrittura nell’associazione *svolgimento*; l’operazione è da ripetersi una volta al mese per circa 4500 turni differenti:  
Costo (Op 1) = (2\*4500+2\*4500) / 1 mese = 18.000 / mese = 600 / gg

- Op 2: 1 accesso in scrittura nell’associazione *malattia*; l’operazione è da ripetersi per circa 10 volte al giorno (assumendo che mediamente ognuno dei 200 bibliotecari richieda un permesso di malattia ogni tre settimane): Costo (Op 2) = (2\*200) / 3 sett = 19 / gg

- Op 3: 10 accessi in lettura nell’entità *bibliotecario* + 10 accessi in lettura nell’associazione *impiego* + 6\*10 accessi in lettura nell’associazione *svolgimento* + 6\*10 accessi in lettura nell’entità *turno*; l’operazione è da ripetersi una volta ogni tre settimane per circa 200 bibliotecari differenti:  
Costo (Op 3) = (10\*200+10\*200+60\*200+60\*200) / 3 sett = 1333 / gg

- Op 4: 1 accesso in scrittura nell’entità *penale* + 1 accesso in scrittura nell’associazione *pagamento* + 1 accesso in scrittura nell’associazione *restituzione in ritardo*; l’operazione è da ripetersi 330 volte al giorno: Costo (Op 20) = (2+2+2) \* 330 / gg = 1980 / gg

- Op 5: 1 accesso in lettura nell’entità *turno* + 1 accesso in lettura nell’associazione *svolgimento* + 1 accesso in lettura nell’entità *bibliotecario* + 1 accesso in lettura nell’associazione *impiego*; l’operazione è da ripetersi una volta al mese per circa 4500 turni differenti:  
Costo (Op 5) = (1\*4500+1\*4500+1\*4500+1\*4500) / mese = 18.000 / mese = 600 / gg

- Op 6: 1 accesso in lettura nell’entità *apertura* + 1 accesso in lettura nell’associazione *orario* + 1 accesso nell’entità *biblioteca*; l’operazione è da ripetersi una volta al mese per circa 120 aperture differenti e per 20 biblioteche differenti: Costo (Op 6) = (1\*120+1\*120+1\*20) / mese = 9 / gg

- Op 7: 1 accesso in scrittura nell’entità *utente*; l’operazione è da ripetersi 200 volte al giorno:  
Costo (Op 7) = 2\*200 / gg = 400 / gg

- Op 8: 55.000 accessi in lettura nell’associazione *ubicazione* + 55.000 accessi in lettura nell’entità *copia di libro* + 55.000 accessi in lettura nell’entità *copia disponibile*; l’operazione è da ripetersi 3 volte al giorno (assumendo che venga effettuata una volta alla settimana per ciascuna biblioteca):  
Costo (Op 8) = (55.000\*20+55.000\*20+55.000\*20) / sett = 3.290.000 / sett = 470.000 / gg

- Op 9: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia dismessa*; l’operazione è da ripetersi 210 volte alla settimana: Costo (Op 9) = (1+2+2) \* 210 / sett = 150 / gg

- Op 10: 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *ubicazione* + 1 accesso in scrittura nell’associazione *istanza di*; l’operazione è da ripetersi 400 volte al giorno:  
Costo (Op 10) = (2+2+2+2) \* 400 / gg = 3200 / gg

- Op 11: 1 accesso in scrittura nell’entità *libro*; l’operazione è da ripetersi 10 volte al giorno:  
Costo (Op 11) = 2\*10 / gg = 20 / gg

- Op 12: 2 accessi in lettura nell’entità *copia disponibile* + 2 accessi in lettura nell’entità *copia di libro* + 2 accessi in lettura nell’associazione *ubicazione* + 2 accessi in lettura nell’associazione *istanza di*; l’operazione è da ripetersi mediamente 5000 volte al giorno:  
Costo (Op 12) = (2+2+2+2) \* 5000 / gg = 40.000 / gg

- Op 13: 1 accesso in lettura nell’entità *biblioteca* + 2 accessi in lettura nell’associazione *ubicazione* + 2 accessi in lettura nell’entità *copia di libro* + 2 accessi in lettura nell’entità *copia disponibile* + 2 accessi in lettura nell’associazione *istanza di*; l’operazione è da ripetersi 1700 volte al giorno per massimo 20 biblioteche: Costo (Op 13) = (1\*20+2\*20+2\*20+2\*20+2\*20) \* 1700 / gg = 306.000 / gg

- Op 14: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’entità *copia non trasferita* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi 3300 volte al giorno: Costo (Op 14) = (1+2+2+2+2) \* 3300 / gg = 29.700 / gg

- Op 15: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’entità *copia trasferita* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno:  
Costo (Op 15): (1+2+2+2+2+2) \* 1700 / gg = 18.700 / gg

- Op 16: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’entità *copia non trasferita* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi 3300 volte al giorno: Costo (Op 16) = (1+2+2+2+2) \* 3300 / gg = 29.700 / gg

- Op 17: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’entità *copia trasferita* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno:  
Costo (Op 17): (1+2+2+2+2+2) \* 1700 / gg = 18.700 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia di libro* + 200.000 accessi in lettura nell’entità *copia in prestito* + 200.000 accessi in lettura nell’entità *copia non trasferita* + 200.000 accessi in lettura nell’associazione *ubicazione* + 200.000 accessi in lettura nell’associazione *prestito* + 200.000 accessi in lettura nell’entità *utente*; l’operazione è da ripetersi 1 volta al giorno:  
Costo (Op 18) = (200.000+200.000+200.000+200.000+200.000+200.000) / gg = 1.200.000 / gg

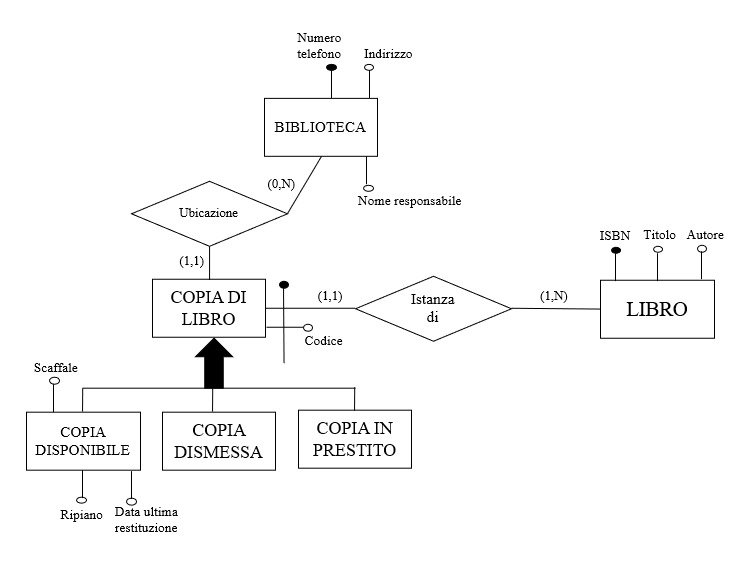
- Op 19: 100.000 accessi in lettura nell’entità *copia di libro* + 100.000 accessi in lettura nell’entità *copia in prestito* + 100.000 accessi in lettura nell’entità *copia trasferita* + 100.000 accessi in lettura nell’assocazione *trasferimento* + 100.000 accessi in lettura nell’associazione *prestito* + 100.000 accessi in lettura nell’entità *utente*; l’operazione è da ripetersi 1 volta al giorno:  
Costo (Op 19) = (100.000+100.000+100.000+100.000+100.000+100.000) / gg = 600.000 / gg

## Ristrutturazione dello schema E-R

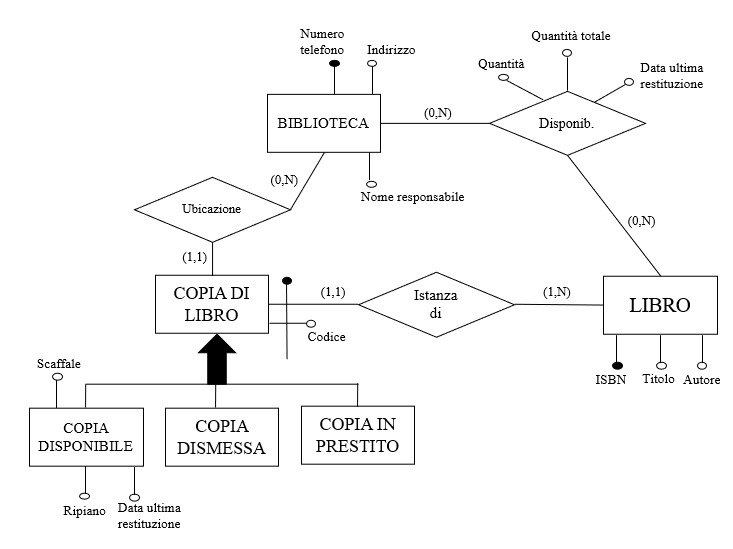
**Analisi delle ridondanze**

All’interno dello schema E-R precedentemente costruito è possibile aggiungere un’associazione molti-a-molti aggiuntiva, denominata *disponibilità*, che connette l’entità *biblioteca* con l’entità *libro.* Lo scopo di tale associazione è tenere traccia in modo diretto di quali libri (e in quale quantità) sono correntemente disponibili in ciascuna biblioteca, oltre a verificare più agevolmente se ci sono libri (assieme alle relative copie) da dismettere in qualche biblioteca.

Assenza del dato ridondante:



Presenza del dato ridondante:



Le operazioni coinvolte nella possibile aggiunta di tale ridondanza sono: Op 8, Op 9, Op 10, Op 13, Op 14, Op 15, Op 16, Op 17. Analizziamo il costo di tali operazioni nel caso in cui ci fosse la ridondanza:

- Op 8: l’operazione può essere riformulata in: “Trova i libri di cui almeno una copia è disponibile all’interno di una biblioteca, indicando la data della loro ultima restituzione”; si hanno dunque al massimo 35.000 accessi in lettura nell’associazione *disponibilità* + 35.000 accessi in lettura nell’entità *libro* (gli accessi sono meno di 55.000 poiché in una biblioteca si hanno più copie di libri che libri stessi); l’operazione è da ripetersi una volta alla settimana per 20 biblioteche differenti:  
Costo’ (Op 8) = (35.000\*20+35.000\*20) / sett = 200.000 / gg = Costo (Op 8) – 270.000 / gg

- Op 9: il costo aggiuntivo è dato da 1 accesso in scrittura nell’associazione *disponibilità*:  
Costo’ (Op 9) = Costo (Op 9) + 2\*210 / sett = Costo (Op 9) + 60 / gg

- Op 10: il costo aggiuntivo è dato da 1 accesso in scrittura nell’associazione *disponibilità*:  
Costo’ (Op 10) = Costo (Op 10) + 2\*400 / gg = Costo (Op 10) + 800 / gg

- Op 13: 1 accesso in lettura nell’entità *biblioteca* + 1 accesso in lettura nell’associaz. *disponibilità* + 1 accesso in lettura nell’entità *libro*; l’operazione è da ripetersi 1700 volte al giorno per 20 biblioteche:  
Costo’ (Op 13) = (1\*20+1\*20+1\*20) \* 1700 / gg = 102.000 / gg = Costo (Op 13) – 204.000 / gg

- Op 14: il costo aggiuntivo è dato da 1 accesso in scrittura nell’associazione *disponibilità*:  
Costo’ (Op 14) = Costo (Op 14) + 2\*3300 / gg = Costo (Op 14) + 6600 / gg

- Op 15: il costo aggiuntivo è dato da 1 accesso in scrittura nell’associazione *disponibilità*:  
Costo’ (Op 15) = Costo (Op 15) + 2\*1700 / gg = Costo (Op 15) + 3400 / gg

- Op 16: il costo aggiuntivo è dato da 1 accesso in scrittura nell’associazione *disponibilità*:  
Costo’ (Op 16) = Costo (Op 16) + 2\*3300 / gg = Costo (Op 16) + 6600 / gg

- Op 17: il costo aggiuntivo è dato da 1 accesso in scrittura nell’associazione *disponibilità*:  
Costo’ (Op 17) = Costo (Op 17) + 2\*1700 / gg = Costo (Op 17) + 3400 / gg

Complessivamente, con l’aggiunta della ridondanza, il risparmio del costo temporale ammonta mediamente a 453.140 accessi giornalieri, che corrisponde al 16,6% rispetto al totale.

D’altra parte, la ridondanza richiede presumibilmente l’aggiunta di una relazione con 5 colonne all’interno del modello logico:

- Numero di telefono della biblioteca, che può occupare fino a 8 byte per ogni tupla.

- ISBN del libro, che può occupare fino a 8 byte per ogni tupla.

- Data dell’ultima restituzione, che, ad esempio, col tipo di dato DATE, può occupare 3 byte per ogni tupla.

- Quantità, che può occupare fino a 4 byte per ogni tupla.

- Quantità totale, che può occupare fino a 4 byte per ogni tupla.

In base ai volumi attesi precedentemente dichiarati, è possibile avere fino a 20\*35.000 = 700.000 tuple all’interno della relazione aggiuntiva, che può comportare un impiego di fino a 18.900.000 byte = 19 MB in più.

In definitiva, possiamo concludere che può risultare conveniente risparmiare circa 450.000 accessi giornalieri al costo di utilizzare 19 MB di memoria in più: ciò porterà a un discreto miglioramento delle prestazioni temporali (il che è sempre auspicabile) e, dall’altra parte, un utilizzo di qualche megabyte di memoria in più generalmente non rappresenta un grosso problema. Per questo motivo, sceglieremo di inserire la ridondanza e, quindi, l’associazione *disponibilità* che lega l’entità *biblioteca* con l’entità *libro*.

**Eliminazione delle generalizzazioni**

All’interno del diagramma E-R si hanno due generalizzazioni: vediamo qual è la modalità con cui conviene eliminarle.

Generalizzazione che ha come parent l’entità *copia in prestito*:

Le operazioni coinvolte in tale generalizzazione sono: Op 14, Op 15, Op 16, Op 17, Op 18, Op 19. Analizziamo il costo di tali operazioni (considerando solo le entità e le associazioni coinvolte nella generalizzazione) nel caso in cui accorpassimo le entità figlie nel parent, nel caso in cui accorpassimo l’entità parent nelle figlie e nel caso in cui sostituissimo semplicemente la generalizzazione con delle associazioni.

🡪 Accorpamento delle entità figlie nel parent

- Op 14: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi mediamente 3300 volte al giorno:  
Costo1 (Op 14) = (2+2) \* 3300 / gg = 13.200 / gg

- Op 15: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno: Costo1 (Op 15) = (2+2+2) \* 1700 / gg = 10.200 / gg

- Op 16: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi mediamente 3300 volte al giorno:  
Costo1 (Op 16) = (2+2) \* 3300 / gg = 13.200 / gg

- Op 17: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno: Costo1 (Op 17) = (2+2+2) \* 1700 / gg = 10.200 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia in prestito* + 200.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi 1 volta al giorno:  
Costo1 (Op 18) = (200.000+200.000) / gg = 400.000 / gg

- Op 19: 100.000 accessi in lettura nell’entità *copia in prestito* + 100.000 accessi in lettura nell’associazione *trasferimento* + 100.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi 1 volta al giorno: Costo1 (Op 19) = (100.000+100.000+100.000) / gg = 300.000 / gg

🡪 Accorpamento dell’entità parent nelle figlie

- Op 14: 1 accesso in scrittura nell’entità *copia non trasferita* + 1 accesso in scrittura nell’associazione *prestito1*; l’operazione è da ripetersi mediamente 3300 volte al giorno:  
Costo2 (Op 14) = (2+2) \* 3300 / gg = 13.200 / gg

- Op 15: 1 accesso in scrittura nell’entità *copia trasferita* + 1 accesso in scrittura nell’associazione *prestito2* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno: Costo2 (Op 15) = (2+2+2) \* 1700 / gg = 10.200 / gg

- Op 16: 1 accesso in scrittura nell’entità *copia non trasferita* + 1 accesso in scrittura nell’associazione *prestito1*; l’operazione è da ripetersi mediamente 3300 volte al giorno:  
Costo2 (Op 16) = (2+2) \* 3300 / gg = 13.200 / gg

- Op 17: 1 accesso in scrittura nell’entità *copia trasferita* + 1 accesso in scrittura nell’associazione *prestito2* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno: Costo2 (Op 17) = (2+2+2) \* 1700 / gg = 10.200 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia non trasferita* + 200.000 accessi in lettura nell’associazione *prestito1*; l’operazione è da ripetersi 1 volta al giorno:  
Costo2 (Op 18) = (200.000+200.000) / gg = 400.000 / gg

- Op 19: 100.000 accessi in lettura nell’entità *copia trasferita* + 100.000 accessi in lettura nell’associazione *trasferimento* + 100.000 accessi in lettura nell’associazione *prestito2*; l’operazione è da ripetersi 1 volta al giorno: Costo2 (Op 19) = (100.000+100.000+100.000) / gg = 300.000 / gg

🡪 Sostituzione della generalizzazione con delle associazioni

- Op 14: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *in prestito & non trasferita* + 1 accesso in scrittura nell’entità *copia non trasferita* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi 3300 volte al giorno:  
Costo3 (Op 14) = (2+2+2+2) \* 3300 / gg = 26.400 / gg

- Op 15: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *in prestito & trasferita* + 1 accesso in scrittura nell’entità *copia trasferita* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno: Costo3 (Op 15) = (2+2+2+2+2) \* 1700 / gg = 17.000 / gg

- Op 16: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *in prestito & non trasferita* + 1 accesso in scrittura nell’entità *copia non trasferita* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi 3300 volte al giorno:  
Costo3 (Op 16) = (2+2+2+2) \* 3300 / gg = 26.400 / gg

- Op 17: 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *in prestito & trasferita* + 1 accesso in scrittura nell’entità *copia trasferita* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno: Costo3 (Op 17) = (2+2+2+2+2) \* 1700 / gg = 17.000 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia in prestito* + 200.000 accessi in lettura nell’associazione *in prestito & non trasferita* + 200.000 accessi in lettura nell’entità *copia non trasferita* + 200.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi 1 volta al giorno: Costo3 (Op 18) = (200.000+200.000+200.000+200.000) / gg = 800.000 / gg

- Op 19: 100.000 accessi in lettura nell’entità *copia in prestito* + 100.000 accessi in lettura nell’associazione *in prestito & trasferita* + 100.000 accessi in lettura nell’entità *copia trasferita* + 100.000 accessi in lettura nell’associazione *prestito* + 100.000 accessi in lettura nell’associazione *trasferimento*; l’operazione è da ripetersi una volta al giorno:  
Costo3 (Op 19) = (100.000+100.000+100.000+100.000+100.000) / gg = 500.000 / gg

Dalle considerazioni appena fatte, possiamo dedurre che, dal punto di vista delle prestazioni, non c’è molta differenza tra l’accorpare le entità figlie nel parent e l’accorpare l’entità parent nelle figlie, mentre sostituire le generalizzazione con delle associazioni potrebbe risultare sconveniente.  
  
  
  
Generalizzazione che ha come parent l’entità *copia di libro*:

Le operazioni coinvolte nella generalizzazione sono: Op 4, Op 9, Op 10, Op 12, Op 14, Op 15, Op 16, Op 17, Op 18, Op 19. Analizziamo il costo di tali operazioni (considerando solo le entità e le associazioni coinvolte nella generalizzazione e assumendo momentaneamente che le entità figlie siano solo *copia disponibile*, *copia dismessa* e *copia in prestito*) nel caso in cui accorpassimo le entità figlie nel parent, nel caso in cui accorpassimo l’entità parent nelle figlie e nel caso in cui sostituissimo semplicemente la generalizzazione con delle associazioni.

🡪 Accorpamento delle entità figlie nel parent

- Op 4: 1 accesso in scrittura nell’associazione *restituzione in ritardo*; l’operazione è da ripetersi 330 volte al giorno: CostoA (Op 20) = 2\*330 / gg = 660 / gg

- Op 9: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia di libro*; l’operazione è da ripetersi 210 volte alla settimana: CostoA (Op 9) = (1+2) \* 210 / sett = 90 / gg

- Op 10: 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’associazione *ubicazione* + 1 accesso in scrittura nell’associazione *istanza di*; l’operazione è da ripetersi 400 volte al giorno: CostoA (Op 10) = (2+2+2) \* 400 / gg = 2400 / gg

- Op 12: 2 accessi in lettura nell’entità *copia di libro* + 2 accessi in lettura nell’associazione *ubicazione* + 2 accessi in lettura nell’associazione *istanza di*; l’operazione è da ripetersi 5000 volte al giorno: CostoA (Op 12) = (2+2+2) \* 5000 / gg = 30.000 / gg

- Op 14: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi mediamente 3300 volte al giorno: CostoA (Op 14) = (1+2+2) \* 3300 / gg = 16.500 / gg

- Op 15: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno:  
CostoA (Op 15) = (1+2+2+2) \* 1700 / gg = 11.900 / gg

- Op 16: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi mediamente 3300 volte al giorno: CostoA (Op 16) = (1+2+2) \* 3300 / gg = 16.500 / gg

- Op 17: 1 accesso in lettura nell’entità *copia di libro* + 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno:  
CostoA (Op 17) = (1+2+2+2) \* 1700 / gg = 11.900 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia di libro* + 200.000 accessi in lettura nell’associazione *ubicazione* + 200.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi una volta al giorno: CostoA (Op 18) = (200.000+200.000+200.000) / gg = 600.000 / gg

- Op 19: 100.000 accessi in lettura nell’entità *copia di libro* + 100.000 accessi in lettura nell’associazione *trasferimento* + 100.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi una volta al giorno: CostoA (Op 19) = (100.000+100.000+100.000) / gg = 300.000 / gg

🡪 Accorpamento dell’entità parent nelle figlie

- Op 4: 1 accesso in scrittura nell’associazione *restituzione in ritardo1*; l’operazione è da ripetersi 330 volte al giorno: CostoB (Op 20) = 2\*330 / gg = 660 / gg

- Op 9: 1 accesso in lettura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia dismessa* + 1 accesso in scrittura nell’associazione *ubicazione1* + 1 accesso in scrittura nell’associazione *ubicazione2* + 1 accesso in scrittura nell’associazione *istanza di1* + 1 accesso in scrittura nell’associazione *istanza di2* + 1 accesso in scrittura nell’associazione *restituzione in ritardo1* + 1 accesso in scrittura nell’associazione *restituzione in ritardo2*; l’operazione è da ripetersi 210 volte alla settimana:  
CostoB (Op 9) = (1+2+2+2+2+2+2+2+2) \* 210 / sett = 510 / gg

- Op 10: 1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *ubicazione1* + 1 accesso in scrittura nell’associazione *istanza di1*; l’operazione è da ripetersi 400 volte al giorno: CostoB (Op 10) = (2+2+2) \* 400 / gg = 2400 / gg

- Op 12: 2 accessi in lettura nell’entità *copia disponibile* + 2 accessi in lettura nell’associazione *ubicazione1* + 2 accessi in lettura nell’associazione *istanza di1*; l’operazione è da ripetersi 5000 volte al giorno: CostoB (Op 12) = (2+2+2) \* 5000 / gg = 30.000 / gg

- Op 14: 1 accesso in lettura nell’entità *copia disponibile* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *ubicazione1* + 1 accesso in scrittura nell’associazione *ubicazione3* + 1 accesso in scrittura nell’associazione *istanza di1* + 1 accesso in scrittura nell’associazione *istanza di3* + 1 accesso in scrittura nell’associazione *restituzione in ritardo1* + 1 accesso in scrittura nell’associazione *restituzione in ritardo3*; l’operazione è da ripetersi 3300 volte al giorno:  
CostoB (Op 14) = (1+2+2+2+2+2+2+2+2+2) \* 3300 / gg = 62.700 / gg

- Op 15: 1 accesso in lettura nell’entità *copia disponibile* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento* + 1 accesso in scrittura nell’associazione *ubicazione1* + 1 accesso in scrittura nell’associazione *ubicazione3* + 1 accesso in scrittura nell’associazione *istanza di1* + 1 accesso in scrittura nell’associazione *istanza di3* + 1 accesso in scrittura nell’associazione *restituzione in ritardo1* + 1 accesso in scrittura nell’associazione *restituzione in ritardo3*; l’operazione è da ripetersi 1700 volte al giorno:  
CostoB (Op 15) = (1+2+2+2+2+2+2+2+2+2+2) \* 1700 / gg = 35.700 / gg

- Op 16: 1 accesso in lettura nell’entità *copia in prestito* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *ubicazione1* + 1 accesso in scrittura nell’associazione *ubicazione3* + 1 accesso in scrittura nell’associazione *istanza di1* + 1 accesso in scrittura nell’associazione *istanza di3* + 1 accesso in scrittura nell’associazione *restituzione in ritardo1* + 1 accesso in scrittura nell’associazione *restituzione in ritardo3*; l’operazione è da ripetersi 3300 volte al giorno:  
CostoB (Op 16) = (1+2+2+2+2+2+2+2+2+2) \* 3300 / gg = 62.700 / gg

- Op 17: 1 accesso in lettura nell’entità *copia in prestito* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento* + 1 accesso in scrittura nell’associazione *ubicazione1* + 1 accesso in scrittura nell’associazione *ubicazione3* + 1 accesso in scrittura nell’associazione *istanza di1* + 1 accesso in scrittura nell’associazione *istanza di3* + 1 accesso in scrittura nell’associazione *restituzione in ritardo1* + 1 accesso in scrittura nell’associazione *restituzione in ritardo3*; l’operazione è da ripetersi 1700 volte al giorno:  
CostoB (Op 17) = (1+2+2+2+2+2+2+2+2+2+2) \* 1700 / gg = 35.700 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia in prestito* + 200.000 accessi in lettura nell’associazione *ubicazione3* + 200.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi una volta al giorno: CostoB (Op 18) = (200.000+200.000+200.000) / gg = 600.000 / gg

- Op 19: 100.000 accessi in lettura nell’entità *copia in prestito* + 100.000 accessi in lettura nell’associazione *trasferimento* + 100.000 accessi in lettura nell’associazione *prestito*: l’operazione è da ripetersi una volta al giorno: CostoB (Op 19) = (100.000+100.000+100.000) / gg = 300.000 / gg

🡪 Sostituzione della generalizzazione con delle associazioni

- Op 4: 1 accesso in scrittura nell’associazione *restituzione in ritardo*; l’operazione è da ripetersi 330 volte al giorno: CostoC (Op 20) = 2\*330 / gg = 660 / gg

- Op 9: 1 accesso in lettura nell’entità *copia disponibile* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *sottoinsieme copia disponibile* + 1 accesso in scrittura nell’entità *copia dismessa* + 1 accesso in scrittura nell’associazione *sottoinsieme copia dismessa*; l’operazione è da ripetersi 210 volte alla settimana:  
CostoC (Op 9) = (1+2+2+2+2) \* 210 / sett = 270 / gg

- Op 10: 1 accesso in scrittura nell’entità *copia di libro* + 1 accesso in scrittura nell’associazione *ubicazione* + 1 accesso in scrittura nell’associazione *istanza di* + 1 accesso in scrittura nell’associazione *sottoinsieme copia disponibile* + 1 accesso in scrittura nell’entità *copia disponibile*; l’operazione è da ripetersi 400 volte al giorno: CostoC (Op 10) = (2+2+2+2+2) \* 400 / gg = 4000 / gg

- Op 12: 2 accessi in lettura nell’entità *copia disponibile* + 2 accessi in lettura nell’associazione *sottoinsieme copia disponibile* + 2 accessi in lettura nell’entità *copia di libro* + 2 accessi in lettura nell’associazione *ubicazione* + 2 accessi in lettura nell’associazione *istanza di*; l’operazione è da ripetersi 5000 volte al giorno: CostoC (Op 12) = (2+2+2+2+2) \* 5000 / g = 50.000 / gg

- Op 14: 1 accesso in lettura nell’entità *copia disponibile* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *sottoinsieme copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *sottoinsieme copia in prestito* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi mediamente 3300 volte al giorno: CostoC (Op 14) = (1+2+2+2+2+2) \* 3300 / gg = 36.300 / gg

- Op 15: 1 accesso in lettura nell’entità *copia disponibile* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *sottoinsieme copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *sottoinsieme copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno:  
CostoC (Op 15) = (1+2+2+2+2+2+2) \* 1700 / gg = 22.100 / gg

- Op 16: 1 accesso in lettura nell’entità *copia in prestito* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *sottoinsieme copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *sottoinsieme copia in prestito* + 1 accesso in scrittura nell’associazione *prestito*; l’operazione è da ripetersi mediamente 3300 volte al giorno: CostoC (Op 16) = (1+2+2+2+2+2) \* 3300 / gg = 36.300 / gg

- Op 17: 1 accesso in lettura nell’entità *copia in prestito* +1 accesso in scrittura nell’entità *copia disponibile* + 1 accesso in scrittura nell’associazione *sottoinsieme copia disponibile* + 1 accesso in scrittura nell’entità *copia in prestito* + 1 accesso in scrittura nell’associazione *sottoinsieme copia in prestito* + 1 accesso in scrittura nell’associazione *prestito* + 1 accesso in scrittura nell’associazione *trasferimento*; l’operazione è da ripetersi 1700 volte al giorno:  
CostoC (Op 17) = (1+2+2+2+2+2+2) \* 1700 / gg = 22.100 / gg

- Op 18: 200.000 accessi in lettura nell’entità *copia di libro* + 200.000 accessi in lettura nell’associazione *sottoinsieme copia in prestito* + 200.000 accessi in lettura nell’entità *copia in prestito* + 200.000 accessi in lettura nell’associazione *ubicazione* + 200.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi una volta al giorno:  
CostoC (Op 18) = (200.000+200.000+200.000+200.000+200.000) / gg = 1.000.000 / gg

- Op 19: 100.000 accessi in lettura nell’entità *copia in prestito* + 100.000 accessi in lettura nell’associazione *trasferimento* + 100.000 accessi in lettura nell’associazione *prestito*; l’operazione è da ripetersi una volta al giorno: CostoC (Op 19) = (100.000+100.000+100.000) / gg = 300.000 / gg

In definitiva:

CostoA (tot) = 990.000 / gg (circa)

CostoB (tot) = 1.130.000 / gg (circa)

CostoC (tot) = 1.470.000 / gg (circa)

Da questa analisi, possiamo dedurre che sostituire la generalizzazione con delle associazioni potrebbe risultare sconveniente perché può gravare sulle prestazioni temporali del database.  
D’altra parte, possiamo osservare che, accorpando le entità figlie nel parent, andremmo incontro a uno spreco di memoria dovuto sia alla necessità di aggiungere un attributo aggiuntivo (*stato*) che indichi se la copia in questione è disponibile, dismessa oppure in prestito, sia alla presenza di eventuali valori nulli aggiuntivi in corrispondenza degli attributi relativi alla sola entità *copia disponibile*. Quantifichiamo tale spreco di memoria:

a) L’attributo aggiuntivo *stato* (se trattato come stringa) può occupare al più 45 byte per ogni tupla relativa all’entità *copia di libro*.

b) L’attributo *scaffale* può occupare indicativamente 8 byte per ogni tupla.

c) L’attributo *ripiano* può occupare indicativamente 8 byte per ogni tupla.

d) L’attributo *data ultima restituzione* (se trattato come DATE) può occupare 3 byte per ogni tupla.

Il numero di tuple relative all’entità copia di libro che mediamente avrebbero gli attributi *scaffale*, *ripiano* e *data ultima restituzione* impostati a *null*, in base al volume atteso dei dati precedentemente analizzato, è pari a 400.000. Ne consegue che, in caso di accorpamento delle entità figlie nel parent, rispetto al caso di accorpamento dell’entità parent nelle figlie, richiede un impiego di memoria aggiuntiva pari a: 45\*1.500.000+(8+8+3) \* 400.000 byte = 75.100.000 byte = 75 MB.

D’altro lato, l’accorpamento dell’entità parent nelle figlie porta a un aumento del costo temporale giornaliero delle operazioni pari a circa 14,1%.

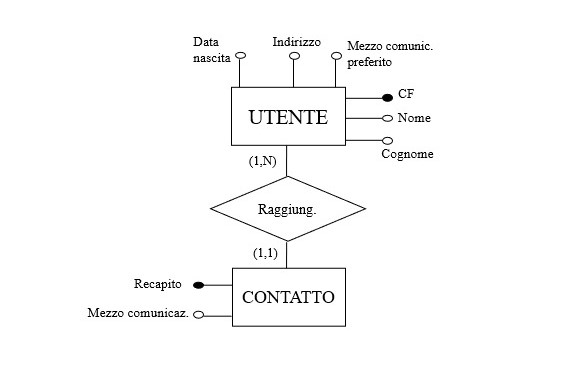
Come abbiamo già osservato durante l’analisi sull’aggiunta della ridondanza, un risparmio di decine (o centinaia) di migliaia di accessi giornalieri al database è generalmente preferibile a un risparmio di qualche decina di megabyte di memoria: perciò, la soluzione che adotteremo consiste nell’accorpare le entità figlie nel parent; per semplicità, effettueremo la medesima scelta per la generalizzazione analizzata precedentemente (che ha l’entità *copia in prestito* come parent), ottenendo così un’unica entità *copia di libro* al posto della gerarchia di generalizzazioni di partenza.

**Scelta degli identificatori primari**

Tutte le entità all’interno del nostro diagramma E-R hanno già un unico identificatore: sarà lui a svolgere il ruolo di identificatore primario.

## Trasformazione di attributi e identificatori

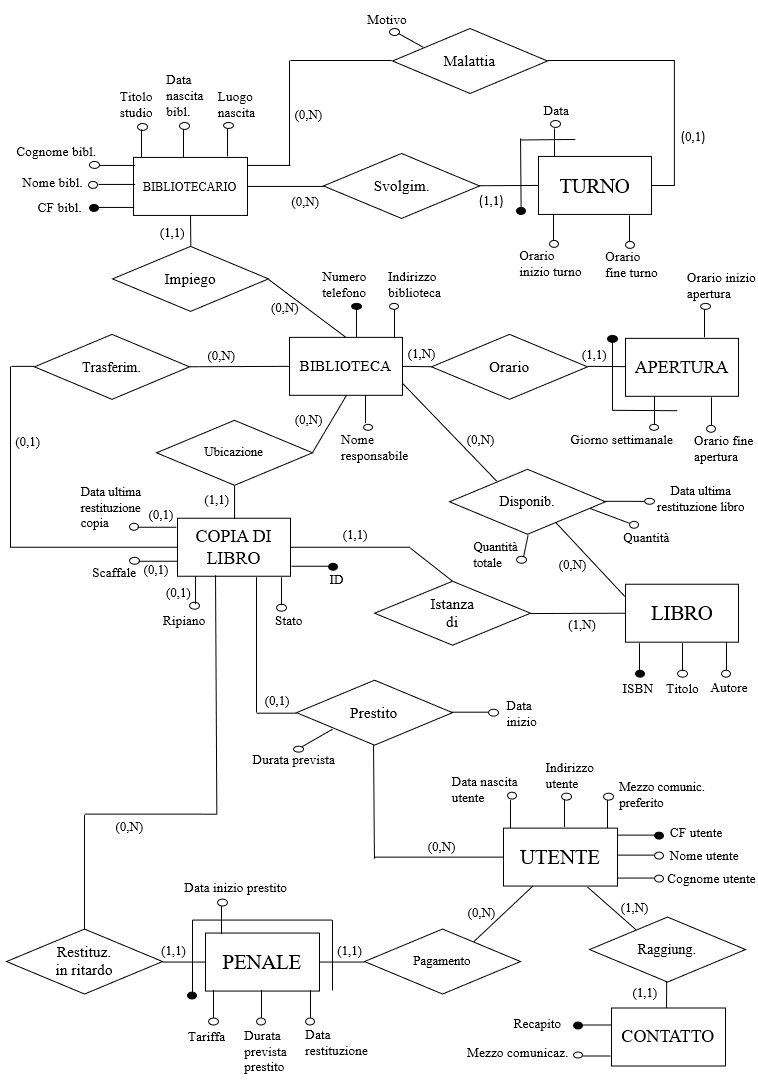
L’entità *utente* ha un attributo composto multivalore (*contatto*) che è difficile da tradurre direttamente in schema relazionale, per cui richiede un aggiustamento preliminare. In particolare, andremo a trasformare tale attributo in un’entità forte (denominata sempre *contatto*), che sarà caratterizzata dagli attributi *recapito*, *mezzo di comunicazione* e sarà relazionata all’entità *utente* tramite un’associazione *raggiungimento* uno-a-molti; inoltre, la nuova entità *contatto* avrà l’attributo *recapito* come identificatore primario.



Per quanto riguarda gli identificatori, notiamo che quello dell’entità *copia di libro* può essere alleggerito: in effetti, è possibile attribuire a tale entità un ID che la identifichi del tutto, al posto di un codice che deve essere accompaganto dall’ISBN del relativo libro. Ne conseguono così l’eliminazione dell’identificatore esterno e la trasformazione di *copia di libro* in entità forte.

Infine, disambiguiamo gli attributi omonimi di entità diverse specificando, appunto, a quale entità si riferiscono.

Con questo, abbiamo concluso la fase di ristrutturazione dello schema E-R. Nella pagina seguente viene riportato il diagramma ristrutturato.



## Traduzione di entità e associazioni

Fase 1 – trasformazione delle entità forti che partecipano con cardinalità massima pari a N a tutte le associazioni ad esse collegate:

- BIBLIOTECA (Numero telefono, Indirizzo biblioteca, Nome responsabile)

- LIBRO (ISBN, Titolo, Autore)

- UTENTE (CF utente, Nome utente, Cognome utente, Data nascita utente, Indirizzo utente, Mezzo comunicazione preferito)

Fase 2 – trasformazione delle entità forti che partecipano con cardinalità (1,1) ad almeno una delle associazioni ad esse collegate:

- BIBLIOTECARIO (CF bibliotecario, Nome bibliotecario, Cognome bibliotecario, Data nascita bibliotecario, Luogo nascita, Titolo studio, Biblioteca impiego)

- CONTATTO (Recapito, Mezzo comunicazione, Utente)

- COPIA DI LIBRO (ID, Biblioteca ubicazione, Libro, Stato, Scaffale\*, Ripiano\*, Data ultima restituzione copia\*)

Dovremo aggiungere:

🡪 Un vincolo di integrità referenziale che lega l’attributo *Biblioteca impiego* dello schema relazionale *Bibliotecario* con l’attributo *Numero telefono* dello schema relazionale *Biblioteca*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Utente* dello schema relazionale *Contatto* con l’attributo *CF utente* dello schema relazionale *Utente*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Libro* dello schema relazionale *Copia di libro* con l’attributo *ISBN* dello schema relazionale *Libro*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Biblioteca ubicazione* dello schema relazionale *Copia di libro* con l’attributo *Numero telefono* dello schema relazionale *Biblioteca*.

Fase 3 – trasformazione delle entità forti che partecipano con cardinalità (0,1) ad almeno una delle associazioni ad esse collegate:

L’entità *copia di libro* partecipa a due associazioni con cardinalità (0,1) (*prestito*, *trasferimento*); per queste due associazioni, a differenza di quelle precedenti, effettueremo una trasformazione non ottima, dato che presumibilmente la maggior parte delle tuple relative all’entità *copia di libro* non parteciperà a queste due associazioni. Perciò, aggiungeremo i seguenti schemi relazionali:

- PRESTITO (Copia prestata, Utente destinatario, Data inizio, Durata prevista)

- TRASFERIMENTO (Copia trasferita, Biblioteca destinazione)

Dovremo aggiungere:

🡪 Un vincolo di integrità referenziale che lega l’attributo *Copia prestata* dello schema relazionale *Prestito* con l’attributo *ID* dello schema relazionale *Copia di libro*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Utente destinatario* dello schema relazionale *Prestito* con l’attributo *CF utente* dello schema relazionale *Utente*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Copia trasferita* dello schema relazionale *Trasferimento* con l’attributo *ID* dello schema relazionale *Copia di libro*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Biblioteca destinazione* dello schema relazionale *Trasferimento* con l’attributo *Numero telefono* dello schema relazionale *Biblioteca*.

Fase 4 – trasformazione delle entità deboli:

- TURNO (Bibliotecario, Data, Orario inizio turno, Orario fine turno)

- APERTURA (Biblioteca, Giorno settimanale, Orario inizio apertura, Orario fine apertura)

- PENALE (Copia, Utente pagante, Data inizio prestito, Data restituzione, Durata prevista prestito, Tariffa)

Dovremo aggiungere:

🡪 Un vincolo di integrità referenziale che lega l’attributo *Bibliotecario* dello schema relazionale *Turno* con l’attributo *CF bibliotecario* dello schema relazionale *Bibliotecario*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Biblioteca* dello schema relazionale *Apertura* con l’attributo *Numero telefono* dello schema relazionale *Biblioteca*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Copia* dello schema relazionale *Penale* con l’attributo *ID* dello schema relazionale *Copia di libro*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Utente pagante* dello schema relazionale *Penale* con l’attributo *CF utente* dello schema relazionale *Utente*.

Fase 5 – trasformazione delle associazioni rimanenti:

- DISPONIBILITÀ (Numero biblioteca, ISBN libro, Quantità, Quantità totale, Data ultima restituzione libro)

- MALATTIA (Bibliotecario malato, Data malattia, Motivo, Bibliotecario sostituto)

Dovremo aggiungere:

🡪 Un vincolo di integrità referenziale che lega l’attributo *Numero biblioteca* dello schema relazionale *Disponibilità* con l’attributo *Numero telefono* dello schema relazionale *Biblioteca*.

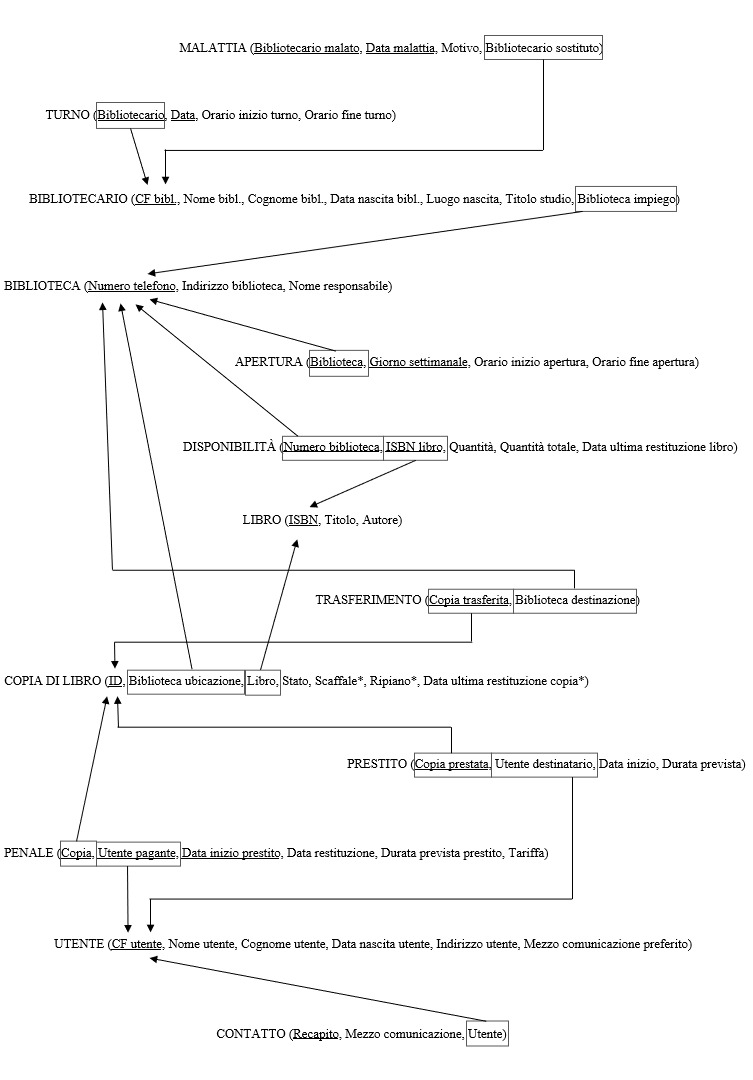
🡪 Un vincolo di integrità referenziale che lega l’attributo *ISBN libro* dello schema relazionale *Disponibilità* con l’attributo *ISBN* dello schema relazionale *Libro*.

🡪 Un vincolo di integrità referenziale che lega l’attributo *Bibliotecario sostituto* dello schema relazionale *Malattia* con l’attributo *CF bibliotecario* dello schema relazionale *Bibliotecario*.

Per imporre che il bibliotecario malato e la data di malattia all’interno della relazione *Malattia* debbano già essere presenti nella relazione *Turno*, non sono sufficienti dei vincoli di integrità referenziale la cui tabella referente è *Malattia* e la cui tabella riferita è *Turno*, poiché si tratta di un’asserzione che coinvolge più di un attributo. Perciò, esprimeremo tale asserzione direttamente tramite un controllo a software da aggiungere manualmente durante la progettazione fisica.

Inoltre, è opportuno osservare anche che si è scelto di effettuare una trasformazione non ottima anche per l’associazione *Malattia*, che è dunque divenuta una relazione separata da *Turno*. Questa scelta è dovuta al fatto che, di 4500 turni previsti, solo 300 sarebbero coinvolti nell’associazione *Malattia*, per cui una trasformazione ottima avrebbe portato ad avere la stragrande maggioranza delle tuple di *Turno* con dei valori nulli in corrispondenza degli attributi *Motivo* e *Bibliotecario sostituto*.

Nella pagina seguente viene riportata una rappresentazione grafica del modello relazionale completo:



## Normalizzazione del modello relazionale

- 1NF: Una base di dati è in 1NF (prima forma normale) se ogni relazione contenuta nella base di dati è in 1NF; una relazione è in 1NF se e solo se contiene una chiave primaria, ogni suo attributo è definito su un dominio con valori atomici (indivisibili) e ogni suo attributo contiene un singolo valore da quel dominio.

In base a questa definizione, possiamo concludere che il modello relazionale precedentemente descritto è già in forma 1NF.

- 2NF: Una base di dati è in 2NF (seconda forma normale) quando è in 1NF e per ogni relazione tutti gli attributi non-chiave dipendono funzionalmente dall’intera chiave composta.

In base a questa definizione, il nostro modello relazionale non è in forma 2NF poiché, all’interno della relazione *Penale*, c’è l’attributo non-chiave *Tariffa* che dipende dall’attributo *Data inizio prestito*, ma non dagli attributi *Copia*, *Utente pagante*. Per far sì che la nostra base di dati sia in 2NF, è necessario scomporre la relazione *Penale* nel seguente modo:

- TARIFFA (Inizio prestito, Restituzione, Durata pattuita prestito, Valore tariffa)

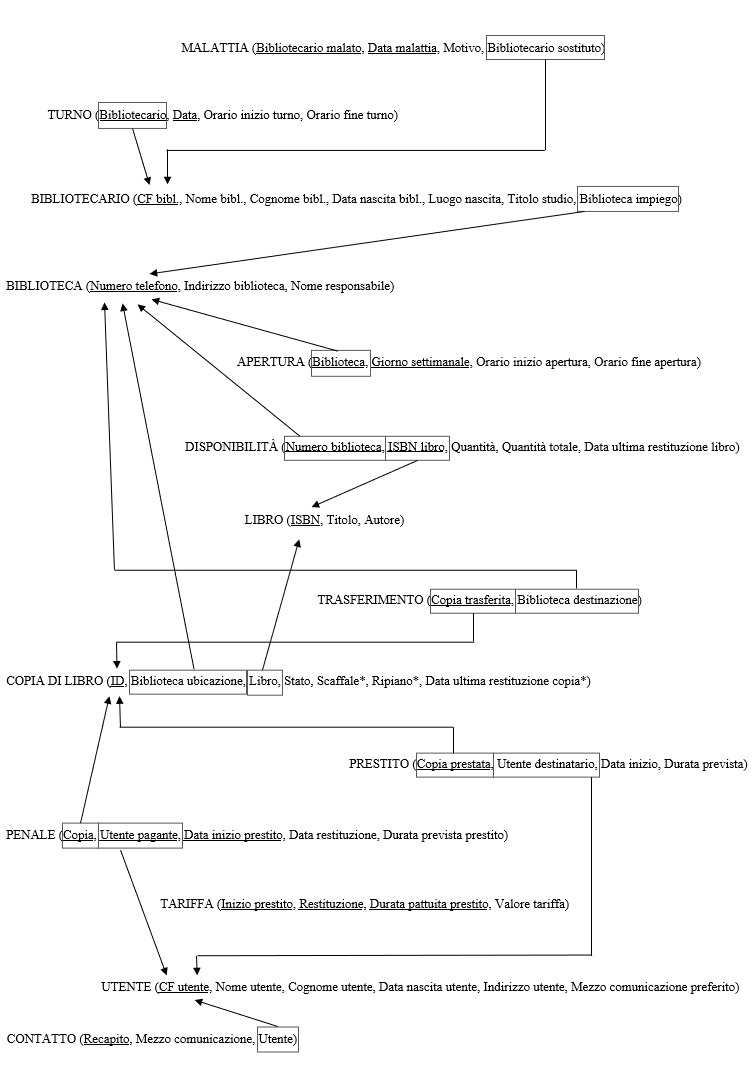
- PENALE (Copia, Utente pagante, Data inizio prestito, Data restituzione, Durata prevista prestito)

Notiamo che, per garantire coerenza tra le relazioni *Penale* e *Tariffa* (in modo tale che in *Penale* può esistere una tupla in cui *Data inizio prestito* assume un valore A, *Data restituzione* assume un valore B e *Durata prevista prestito* assume un valore C se e solo se in *Tariffa* esiste già una tupla in cui *Inizio prestito* assume il valore A, *Restituzione* assume il valore B e *Durata pattuita prestito* assume il valore C), non è sufficiente esprimere dei vincoli di integrità referenziale la cui tabella referente è *Penale* e la cui tabella riferita è *Tariffa*. Perciò, anche in questo caso, effettueremo manualmente un controllo a software durante la progettazione fisica.

- 3NF: Una base di dati è in 3NF (terza forma normale) se è in 2NF e, per ogni relazione, non esistono attributi non-chiave che dipendono da altri attributi non-chiave. In altre parole, non devono esserci dipendenze funzionali transitive.

In base a questa definizione, a seguito della decomposizione effettuata sulla relazione *Penale*, possiamo dire che il modello relazionale è già anche in forma 3NF.

Di seguito viene riportata una rappresentazione grafica del modello relazionale in forma 2NF (che, nel nostro caso specifico, coincide con la forma 3NF):



# Progettazione fisica

Nella fase di progettazione fisica, per motivi pratici sono state inserite due ulteriori relazioni:

- *Contatto Preferito*, che contiene gli attributi *Codice fiscale utente* e *Mezzo di comunicazione preferito*; questa informazione è stata separata dalla tabella *Utente* in modo da agevolare la realizzazione di asserzioni come, ad esempio, vincoli di integrità referenziali.

- *User*, che contiene gli attributi *Username*, *Password* e *Ruolo* e ha un’utilità nell’ambito dell’autenticazione alla base di dati.

## Utenti e privilegi

Gli utenti (che d’ora in poi chiameremo ***user*** per non confonderli con gli utenti del circuito di biblioteche) previsti per l’applicazione sono:

- *Login*: è associato al ruolo *login*, che comprende i seguenti privilegi:

1) Execute della stored procedure *login*: permette di effettuare l’autenticazione al database, tramite cui verrà riconosciuto l’effettivo user che sta per utilizzare l’applicazione, in modo tale che lo user possa accedere a tutti e soli i servizi di sua competenza.

Lo user *Login*, in effetti, è un utente fittizio, ovvero un account di sistema che viene utilizzato dal client per connettersi al database.

- *Bibliotecario*: è associato al ruolo *bibliotecario*, che comprende i seguenti privilegi:

1. Execute della stored procedure *aggiungi\_contatto*: permette di inserire un nuovo contatto di un utente del circuito di biblioteche all’interno della relazione *Contatto*.
2. Execute della stored procedure *aggiungi\_contatto\_preferito*: permette di specificare qual è il mezzo di comunicazione preferito di un utente del circuito.
3. Execute della stored procedure *aggiungi\_utente*: permette di inserire un nuovo utente del circuito all’interno della relazione *Utente*; questa procedura, insieme ad *aggiungi\_contatto* e *aggiungi\_contatto\_preferito*, consente la registrazione di un nuovo utente a una biblioteca con il conseguente salvataggio di tutte le sue informazioni di interesse all’interno del database.
4. Execute della stored procedure *inizia\_prestito*: all’atto di effettuare un prestito, permette al bibliotecario di verificare se c’è almeno una copia di un determinato libro nello stato ‘disponibile’ all’interno della biblioteca in cui egli lavora. Nel caso in cui la copia c’è, la procedura restituisce lo scaffale e il ripiano in cui la copia è posizionata, e registra il prestito; altrimenti, restituisce la lista di biblioteche in cui è disponibile almeno una copia del libro specificato.
5. Execute della stored procedure *inizia\_trasferimento*: permette al bibliotecario di registrare una richiesta di trasferimento all’interno del database dopo aver appurato che all’interno della biblioteca in cui lavora non c’è alcuna copia del libro di interesse nello stato ‘disponibile’.
6. Execute della stored procedure *ottieni\_biblioteca\_impiego*: permette di conoscere la biblioteca in cui il bibliotecario lavora.
7. Execute della stored procedure *ottieni\_ISBN*: permette di conoscere il codice ISBN dei libri che hanno un determinato titolo e un determinato autore; è una procedura che dà un valore aggiunto agli attributi *Titolo* e *Autore* della relazione *Libro*.
8. Execute della stored procedure *report\_prestiti*: permette di generare un report contenente le informazioni riguardanti i prestiti (tuttora in corso) che sono partiti dalla biblioteca in cui il bibliotecario lavora, e le informazioni riguardanti gli utenti della biblioteca che sono coinvolti nei suddetti prestiti. È proprio questa procedura che consente al bibliotecario di recuperare i recapiti degli utenti per sollecitarli nella restituzione delle copie prestate.
9. Execute della stored procedure *termina\_prestito*: permette al bibliotecario di contrassegnare nuovamente una copia di libro come ‘disponibile’, eliminando così le relative tuple dalle relazioni *Prestito* e/o *Trasferimento*; inoltre, questa procedura verifica se la suddetta copia è stata restituita in ritardo e, in tal caso, ne calcola automaticamente la penale, registrandola all’interno del database.

- *Amministratore*: è associato al ruolo *amministratore*, che comprende i seguenti privilegi:

1. Execute della stored procedure *aggiungi\_bibliotecario*: permette di inserire un nuovo bibliotecario all’interno della relazione *Bibliotecario*.
2. Execute della stored procedure *aggiungi\_copia*: permette di inserire una nuova copia e, se non c’è già, il relativo libro all’interno del database.
3. Execute della stored procedure *aggiungi\_malattia*: permette di registrare un permesso di malattia di un bibliotecario all’interno della relazione *Malattia*, assegnando il turno del bibliotecario malato a un altro bibliotecario che sia libero e che abbia impiego nella medesima biblioteca.
4. Execute della stored procedure *aggiungi\_turno*: permette all’amministratore di gestire i turni di lavoro dei bibliotecari, inserendoli all’interno della relazione *Turno*.
5. Execute della stored procedure *crea\_user*: permette all’amministratore di registrare un nuovo user all’interno del database con i relativi username, password e ruolo (quest’ultimo può essere ‘bibliotecario’ oppure ‘amministratore’).
6. Execute della stored procedure *dismetti\_copie*: permette di contrassegnare come dismessi i libri che, all’interno di ciascuna biblioteca, non vengono prestati da oltre 10 anni.
7. Execute della stored procedure *report\_biblioteche\_scoperte*: permette di generare un report contenente la lista delle biblioteche che sono scoperte per almeno un giorno del periodo specificato; ciascuna biblioteca è accompagnata dai giorni in cui è scoperta e i relativi orari, in modo da facilitare il compito di assegnare i turni di lavoro ai bibliotecari in modo da coprire per intero tutti gli orari di apertura delle biblioteche del circuito.
8. Execute della stored procedure *trova\_sostituti*: nel momento in cui un bibliotecario effettua una richiesta di malattia, viene chiamata questa procedura subito prima di *aggiungi\_malattia* in modo tale che l’amministratore possa conoscere la lista dei bibliotecari che sono in grado di sostituire il malato.

L’assegnazione dei privilegi agli user è avvenuta seguendo il **principio del privilegio minimo** in modo da garantire che ciascuno user possa effettuare solo le operazioni strettamente necessarie al suo scopo (che, appunto, consistono nell’esecuzione di alcune stored procedures che incapsulano i comandi SQL che implementano i servizi offerti al client).

## Strutture di memorizzazione

|  |  |  |
| --- | --- | --- |
| **Tabella <apertura>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[2]](#footnote-2)** |
| **Biblioteca** | VARCHAR (15) | PK, NN |
| **Giorno\_settimanale** | INT | PK, NN, UN |
| **Orario\_inizio\_apertura** | TIME | NN |
| **Orario\_fine\_apertura** | TIME | NN |
| **Tabella <biblioteca>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Numero\_telefono** | VARCHAR (15) | PK, NN |
| **Indirizzo\_biblioteca** | VARCHAR (45) | NN |
| **Responsabile** | VARCHAR (45) | NN |
| **Tabella <bibliotecario>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **CF\_bibliotecario** | VARCHAR (16) | PK, NN |
| **Nome\_bibliotecario** | VARCHAR (45) | NN |
| **Cognome\_bibliotecario** | VARCHAR (45) | NN |
| **Data\_nascita\_bibliotecario** | DATE | NN |
| **Luogo\_nascita** | VARCHAR (45) | NN |
| **Titolo\_studio** | VARCHAR (45) | NN |
| **Biblioteca\_impiego** | VARCHAR (15) | NN |
| **Username\_bibliotecario** | VARCHAR (45) | NN, UQ |
| **Tabella <contatto>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Recapito** | VARCHAR (45) | PK, NN |
| **Mezzo\_comunicazione** | ENUM (…) | NN |
| **Utente** | VARCHAR (16) | NN |
| **Tabella <contatto\_preferito>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Codice\_fiscale\_utente** | VARCHAR (16) | PK, NN |
| **Mezzo\_comunicazione\_preferito** | ENUM (…) | NN |
| **Tabella <copia\_di\_libro>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **ID** | INT | PK, NN, UN, AI |
| **Biblioteca\_ubicazione** | VARCHAR (15) | NN |
| **Libro** | VARCHAR (13) | NN |
| **Stato** | ENUM (…) | NN |
| **Scaffale** | INT | UN |
| **Ripiano** | INT | UN |
| **Data\_ultima\_restituzione** | DATE |  |
| **Tabella <disponibilità>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Numero\_biblioteca** | VARCHAR (15) | PK, NN |
| **ISBN\_libro** | VARCHAR (13) | PK, NN |
| **Quantità** | INT | NN, UN |
| **Quantità\_totale** | INT | NN, UN |
| **Data\_ultima\_restituzione\_libro** | DATE |  |
| **Tabella <libro>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **ISBN** | VARCHAR (13) | PK, NN |
| **Titolo** | VARCHAR (45) | NN |
| **Autore** | VARCHAR (45) | NN |
| **Tabella <malattia>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Bibliotecario\_malato** | VARCHAR (16) | PK, NN |
| **Data\_malattia** | DATE | PK, NN |
| **Motivo** | VARCHAR (45) | NN |
| **Bibliotecario\_sostituto** | VARCHAR (16) | NN |
| **Tabella <penale>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Copia** | INT | PK, NN, UN |
| **Utente\_pagante** | VARCHAR (16) | PK, NN |
| **Data\_inizio\_prestito** | DATE | PK, NN |
| **Data\_restituzione** | DATE | NN |
| **Durata\_prevista\_previsto** | ENUM (…) | NN |
| **Tabella <prestito>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Copia\_prestata** | INT | PK, NN, UN |
| **Utente\_destinatario** | VARCHAR (16) | NN |
| **Data\_inizio** | DATE | NN |
| **Durata\_prevista** | ENUM (…) | NN |
| **Tabella <tariffa>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Inizio\_prestito** | DATE | PK, NN |
| **Restituzione** | DATE | PK, NN |
| **Durata\_prevista\_prestito** | ENUM (…) | PK, NN |
| **Valore\_tariffa** | FLOAT | NN, UN |
| **Tabella <trasferimento>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Copia\_trasferita** | INT | PK, NN, UN |
| **Biblioteca\_destinazione** | VARCHAR (15) | NN |
| **Tabella <turno>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Bibliotecario** | VARCHAR (16) | PK, NN |
| **Data** | DATE | PK, NN |
| **Orario\_inizio\_turno** | TIME | NN |
| **Orario\_fine\_turno** | TIME | NN |
| **Tabella <user>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Username** | VARCHAR (45) | PK, NN |
| **Password** | CHAR (32) | NN |
| **Ruolo** | ENUM (…) | NN |
| **Tabella <utente>** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **CF\_utente** | VARCHAR (16) | PK, NN |
| **Nome\_utente** | VARCHAR (45) | NN |
| **Cognome\_utente** | VARCHAR (45) | NN |
| **Data\_nascita\_utente** | DATE | NN |
| **Indirizzo\_utente** | VARCHAR (45) | NN |

## Indici

|  |  |
| --- | --- |
| **Tabella <apertura>** | |
| **Indice <PRIMARY>** | **Tipo[[3]](#footnote-3):** |
| Biblioteca | <PR> |
| Giorno\_settimanale | <PR> |
| **Tabella <biblioteca>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Numero\_telefono | <PR> |
| **Tabella <bibliotecario>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| CF\_bibliotecario | <PR> |
| **Indice <biblioteca\_impiego\_idx>** | **Tipo:** |
| Biblioteca\_impiego | <IDX> |
| **Indice <username\_bibliotecario\_UNIQUE>** | **Tipo:** |
| Username\_bibliotecario | <UQ> |
| **Tabella <contatto>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Recapito | <PR> |
| **Indice <utente\_idx>** | **Tipo:** |
| Utente | <IDX> |
| **Tabella <contatto\_preferito>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Codice\_fiscale\_utente | <PR> |
| **Tabella <copia\_di\_libro>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| ID | <PR> |
| **Indice <biblioteca\_ubicazione\_idx>** | **Tipo:** |
| Biblioteca\_ubicazione | <IDX> |
| **Indice <libro\_idx>** | **Tipo:** |
| Libro | <IDX> |
| **Indice <stato\_idx>** | **Tipo:** |
| Stato | <IDX> |
| **Tabella <disponibilità>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Numero\_biblioteca | <PR> |
| ISBN\_libro | <PR> |
| **Indice <ISBN\_libro\_idx>** | **Tipo:** |
| ISBN\_libro | <IDX> |
| **Indice <data\_ultima\_restituzione\_libro\_idx>** | **Tipo:** |
| Data\_ultima\_restituzione\_libro | <IDX> |
| **Tabella <libro>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| ISBN | <PR> |
| **Tabella <malattia>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Bibliotecario\_malato | <PR> |
| Data\_malattia | <PR> |
| **Indice <bibliotecario\_sostituto\_idx>** | **Tipo:** |
| Bibliotecario\_sostituto | <IDX> |
| **Tabella <penale>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Copia | <PR> |
| Utente\_pagante | <PR> |
| Data\_inizio\_prestito | <PR> |
| **Indice <utente\_pagante\_idx>** | **Tipo:** |
| Utente\_pagante | <IDX> |
| **Tabella <prestito>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Copia\_prestata | <PR> |
| **Indice <utente\_destinatario\_idx>** | **Tipo:** |
| Utente\_destinatario | <IDX> |
| **Tabella <tariffa>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Inizio\_prestito | <PR> |
| Restituzione | <PR> |
| Durate\_pattuita\_prestito | <PR> |
| **Tabella <trasferimento>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Copia\_trasferita | <PR> |
| **Indice <biblioteca\_destinazione\_idx>** | **Tipo:** |
| Biblioteca\_destinazione | <IDX> |
| **Tabella <turno>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Bibliotecario | <PR> |
| Data | <PR> |
| **Tabella <user>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| Username | <PR> |
| **Tabella <utente>** | |
| **Indice <PRIMARY>** | **Tipo:** |
| CF\_utente | <PR> |

- Gli indici <PRIMARY> sono relativi alle chiavi primarie di tutte le relazioni del database; sono dunque molto utili in quanto la chiave primaria di una relazione comprende gli attributi che, più degli altri, sono coinvolti nelle operazioni di select e di join.

- Gli indici <biblioteca\_impiego\_idx>, <username\_bibliotecario\_UNIQUE>, <utente\_idx>, <biblioteca\_ubicazione\_idx>, <libro\_idx>, <ISBN\_libro\_idx>, <bibliotecario\_sostituto\_idx>, <utente\_pagante\_idx>, <utente\_destinatario\_idx>, <biblioteca\_destinazione\_idx> sono relativi agli attributi delle foreign key di tutte le relazioni del database; anch’essi sono utili in quanto permettono di aumentare notevolmente l’efficienza delle operazioni di join che, in generale, risultano essere le più critiche dal punto di vista delle prestazioni.

- Gli indici <stato\_idx>, <data\_ultima\_restituzione\_libro\_idx> sono stati aggiunti per rendere più efficienti le operazioni di select (che nel caso della nostra applicazione richiedono molti accessi in lettura giornalieri) che hanno rispettivamente come predicato l’attributo *Stato* della relazione *Copia di libro* e l’attributo *Data ultima* restituzione libro della relazione *Disponibilità*.

## Trigger

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`biblioteca\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`biblioteca\_BEFORE\_INSERT` BEFORE INSERT ON `biblioteca` FOR EACH ROW

BEGIN

if not NEW.numero\_telefono regexp '^[0-9]{4,15}$' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`biblioteca\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`biblioteca\_BEFORE\_UPDATE` BEFORE UPDATE ON `biblioteca` FOR EACH ROW

BEGIN

if not NEW.numero\_telefono regexp '^[0-9]{4,15}$' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`libro\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`libro\_BEFORE\_INSERT` BEFORE INSERT ON `libro` FOR EACH ROW

BEGIN

if not (NEW.ISBN regexp '^[0-9]{9,9}X$' or NEW.ISBN regexp '^[0-9]{10,10}$' or NEW.ISBN regexp '^[0-9]{13,13}$') then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`libro\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`libro\_BEFORE\_UPDATE` BEFORE UPDATE ON `libro` FOR EACH ROW

BEGIN

if not (NEW.ISBN regexp '^[0-9]{9,9}X$' or NEW.ISBN regexp '^[0-9]{10,10}$' or NEW.ISBN regexp '^[0-9]{13,13}$') then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`utente\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`utente\_BEFORE\_INSERT` BEFORE INSERT ON `utente` FOR EACH ROW

BEGIN

if not NEW.CF\_utente regexp '^[A-Z]{6,6}[0-9]{2,2}[A-Z][0-9]{2,2}[A-Z][0-9]{3,3}  
[A-Z]$' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`utente\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`utente\_BEFORE\_UPDATE` BEFORE UPDATE ON `utente` FOR EACH ROW

BEGIN

if not NEW.CF\_utente regexp '^[A-Z]{6,6}[0-9]{2,2}[A-Z][0-9]{2,2}[A-Z][0-9]{3,3}  
[A-Z]$' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`bibliotecario\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`bibliotecario\_BEFORE\_INSERT` BEFORE INSERT ON `bibliotecario` FOR EACH ROW

BEGIN

if not NEW.CF\_bibliotecario regexp '^[A-Z]{6,6}[0-9]{2,2}[A-Z][0-9]{2,2}[A-Z]  
[0-9]{3,3}[A-Z]$' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`bibliotecario\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`bibliotecario\_BEFORE\_UPDATE` BEFORE UPDATE ON `bibliotecario` FOR EACH ROW

BEGIN

if not NEW.CF\_bibliotecario regexp '^[A-Z]{6,6}[0-9]{2,2}[A-Z][0-9]{2,2}[A-Z]  
[0-9]{3,3}[A-Z]$' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`turno\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`turno\_BEFORE\_INSERT` BEFORE INSERT ON `turno` FOR EACH ROW

BEGIN

declare diff time;

declare seconds int;

select timediff(NEW.orario\_fine\_turno, NEW.orario\_inizio\_turno) into diff;

select time\_to\_sec(diff) into seconds;

if seconds > 28800 or seconds <= 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`turno\_BEFORE\_INSERT\_1` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`turno\_BEFORE\_INSERT\_1` BEFORE INSERT ON `turno` FOR EACH ROW

BEGIN

declare giorno\_sett\_turno int;

declare inizio\_apertura time;

declare fine\_apertura time;

declare diff\_inizio time;

declare diff\_fine time;

declare seconds\_inizio int;

declare seconds\_fine int;

declare var\_count int;

select weekday(NEW.`data`) into giorno\_sett\_turno;

select count(\*) from `bibliotecario` join `apertura` on `biblioteca\_impiego` = `biblioteca`

where `giorno\_settimanale` = giorno\_sett\_turno and `CF\_bibliotecario` = NEW.bibliotecario

into var\_count;

if var\_count = 0 then

signal sqlstate '45000';

end if;

select `orario\_inizio\_apertura`, `orario\_fine\_apertura`

from `bibliotecario` join `apertura` on `biblioteca\_impiego` = `biblioteca`

where `giorno\_settimanale` = giorno\_sett\_turno and `CF\_bibliotecario` = NEW.bibliotecario

into inizio\_apertura, fine\_apertura;

select timediff(NEW.orario\_inizio\_turno, inizio\_apertura) into diff\_inizio;

select time\_to\_sec(diff\_inizio) into seconds\_inizio;

select timediff(fine\_apertura, NEW.orario\_fine\_turno) into diff\_fine;

select time\_to\_sec(diff\_fine) into seconds\_fine;

if seconds\_inizio < 0 or seconds\_fine < 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`turno\_BEFORE\_INSERT\_2` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`turno\_BEFORE\_INSERT\_2` BEFORE INSERT ON `turno` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `malattia`

where `bibliotecario\_sostituto` = NEW.bibliotecario and `data\_malattia` = NEW.data

into count;

if count > 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`turno\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`turno\_BEFORE\_UPDATE` BEFORE UPDATE ON `turno` FOR EACH ROW

BEGIN

declare diff time;

declare seconds int;

select timediff(NEW.orario\_fine\_turno, NEW.orario\_inizio\_turno) into diff;

select time\_to\_sec(diff) into seconds;

if seconds > 28800 or seconds <= 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`turno\_BEFORE\_UPDATE\_1` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`turno\_BEFORE\_UPDATE\_1` BEFORE UPDATE ON `turno` FOR EACH ROW

BEGIN

declare giorno\_sett\_turno int;

declare inizio\_apertura time;

declare fine\_apertura time;

declare diff\_inizio time;

declare diff\_fine time;

declare seconds\_inizio int;

declare seconds\_fine int;

declare var\_count int;

select weekday(NEW.`data`) into giorno\_sett\_turno;

select count(\*) from `bibliotecario` join `apertura` on `biblioteca\_impiego` = `biblioteca`

where `giorno\_settimanale` = giorno\_sett\_turno and `CF\_bibliotecario` = NEW.bibliotecario

into var\_count;

if var\_count = 0 then

signal sqlstate '45000';

end if;

select `orario\_inizio\_apertura`, `orario\_fine\_apertura`

from `bibliotecario` join `apertura` on `biblioteca\_impiego` = `biblioteca`

where `giorno\_settimanale` = giorno\_sett\_turno and `CF\_bibliotecario` = NEW.bibliotecario

into inizio\_apertura, fine\_apertura;

select timediff(NEW.orario\_inizio\_turno, inizio\_apertura) into diff\_inizio;

select time\_to\_sec(diff\_inizio) into seconds\_inizio;

select timediff(fine\_apertura, NEW.orario\_fine\_turno) into diff\_fine;

select time\_to\_sec(diff\_fine) into seconds\_fine;

if seconds\_inizio < 0 or seconds\_fine < 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`turno\_BEFORE\_UPDATE\_2` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`turno\_BEFORE\_UPDATE\_2` BEFORE UPDATE ON `turno` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `malattia`

where `bibliotecario\_sostituto` = NEW.bibliotecario and `data\_malattia` = NEW.data

into count;

if count > 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`malattia\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`malattia\_BEFORE\_INSERT` BEFORE INSERT ON `malattia` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `turno`

where `bibliotecario` = NEW.bibliotecario\_malato and `data` = NEW.data\_malattia

into count;

if count = 0 then

signal sqlstate '45000';

end if;

select count(\*) from `turno`

where `bibliotecario` = NEW.bibliotecario\_sostituto and `data` = NEW.data\_malattia

into count;

if count > 0 then

signal sqlstate '45000';

end if;

select count(\*) from `bibliotecario` b1 join `bibliotecario` b2 on b1.`biblioteca\_impiego` = b2.`biblioteca\_impiego`

where b1.`CF\_bibliotecario` = NEW.bibliotecario\_malato and b2.`CF\_bibliotecario` = NEW.bibliotecario\_sostituto

into count;

if count = 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`malattia\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`malattia\_BEFORE\_UPDATE` BEFORE UPDATE ON `malattia` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `turno`

where `bibliotecario` = NEW.bibliotecario\_malato and `data` = NEW.data\_malattia

into count;

if count = 0 then

signal sqlstate '45000';

end if;

select count(\*) from `turno`

where `bibliotecario` = NEW.bibliotecario\_sostituto and `data` = NEW.data\_malattia

into count;

if count > 0 then

signal sqlstate '45000';

end if;

select count(\*) from `bibliotecario` b1 join `bibliotecario` b2 on b1.`biblioteca\_impiego` = b2.`biblioteca\_impiego`

where b1.`CF\_bibliotecario` = NEW.bibliotecario\_malato and b2.`CF\_bibliotecario` = NEW.bibliotecario\_sostituto

into count;

if count = 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`apertura\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`apertura\_BEFORE\_INSERT` BEFORE INSERT ON `apertura` FOR EACH ROW

BEGIN

declare diff time;

declare seconds int;

select timediff(NEW.orario\_fine\_apertura, NEW.orario\_inizio\_apertura) into diff;

select time\_to\_sec(diff) into seconds;

if seconds <= 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`apertura\_BEFORE\_INSERT\_1` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`apertura\_BEFORE\_INSERT\_1` BEFORE INSERT ON `apertura` FOR EACH ROW

BEGIN

if NEW.giorno\_settimanale > 6 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`apertura\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`apertura\_BEFORE\_UPDATE` BEFORE UPDATE ON `apertura` FOR EACH ROW

BEGIN

declare diff time;

declare seconds int;

select timediff(NEW.orario\_fine\_apertura, NEW.orario\_inizio\_apertura) into diff;

select time\_to\_sec(diff) into seconds;

if seconds <= 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`apertura\_BEFORE\_UPDATE\_1` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`apertura\_BEFORE\_UPDATE\_1` BEFORE UPDATE ON `apertura` FOR EACH ROW

BEGIN

if NEW.giorno\_settimanale > 6 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`disponibilità\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`disponibilità\_BEFORE\_INSERT` BEFORE INSERT ON `disponibilità` FOR EACH ROW

BEGIN

if NEW.quantità > NEW.quantità\_totale then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`disponibilità\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`disponibilità\_BEFORE\_UPDATE` BEFORE UPDATE ON `disponibilità` FOR EACH ROW

BEGIN

if NEW.quantità > NEW.quantità\_totale then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`copia\_di\_libro\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`copia\_di\_libro\_BEFORE\_INSERT` BEFORE INSERT ON `copia\_di\_libro` FOR EACH ROW

BEGIN

declare dieci\_anni\_fa date;

declare giorni\_alla\_dismissione int;

select date\_sub(curdate(), interval 3652 day) into dieci\_anni\_fa;

select datediff(NEW.data\_ultima\_restituzione\_copia, dieci\_anni\_fa) into giorni\_alla\_dismissione;

if giorni\_alla\_dismissione >= 0 and NEW.stato = 'dismessa' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`copia\_di\_libro\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`copia\_di\_libro\_BEFORE\_UPDATE` BEFORE UPDATE ON `copia\_di\_libro` FOR EACH ROW

BEGIN

declare dieci\_anni\_fa date;

declare giorni\_alla\_dismissione int;

select date\_sub(curdate(), interval 3652 day) into dieci\_anni\_fa;

select datediff(NEW.data\_ultima\_restituzione\_copia, dieci\_anni\_fa) into giorni\_alla\_dismissione;

if giorni\_alla\_dismissione >= 0 and NEW.stato = 'dismessa' then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`penale\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`penale\_BEFORE\_INSERT` BEFORE INSERT ON `penale` FOR EACH ROW

BEGIN

declare scad date;

declare num\_giorni\_ritardo int;

if NEW.durata\_prevista\_prestito = '1 mese' then

select date\_add(NEW.data\_inizio\_prestito, INTERVAL 30 DAY) into scad;

elseif NEW.durata\_prevista\_prestito = '2 mesi' then

select date\_add(NEW.data\_inizio\_prestito, INTERVAL 60 DAY) into scad;

elseif NEW.durata\_prevista\_prestito = '3 mesi' then

select date\_add(NEW.data\_inizio\_prestito, INTERVAL 90 DAY) into scad;

end if;

select datediff(NEW.data\_restituzione, scad) into num\_giorni\_ritardo;

if num\_giorni\_ritardo <= 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`penale\_BEFORE\_INSERT\_1` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`penale\_BEFORE\_INSERT\_1` BEFORE INSERT ON `penale` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `tariffa`

where `inizio\_prestito` = NEW.data\_inizio\_prestito and `restituzione` = NEW.data\_restituzione and `durata\_pattuita\_prestito` = NEW.durata\_prevista\_prestito

into count;

if count = 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`penale\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`penale\_BEFORE\_UPDATE` BEFORE UPDATE ON `penale` FOR EACH ROW

BEGIN

declare scad date;

declare num\_giorni\_ritardo int;

if NEW.durata\_prevista\_prestito = '1 mese' then

select date\_add(NEW.data\_inizio\_prestito, INTERVAL 30 DAY) into scad;

elseif NEW.durata\_prevista\_prestito = '2 mesi' then

select date\_add(NEW.data\_inizio\_prestito, INTERVAL 60 DAY) into scad;

elseif NEW.durata\_prevista\_prestito = '3 mesi' then

select date\_add(NEW.data\_inizio\_prestito, INTERVAL 90 DAY) into scad;

end if;

select datediff(NEW.data\_restituzione, scad) into num\_giorni\_ritardo;

if num\_giorni\_ritardo <= 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`penale\_BEFORE\_UPDATE\_1` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`penale\_BEFORE\_UPDATE\_1` BEFORE UPDATE ON `penale` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `tariffa` where `inizio\_prestito` = NEW.data\_inizio\_prestito and `restituzione` = NEW.data\_restituzione and `durata\_pattuita\_prestito` = NEW.durata\_prevista\_prestito into count;

if count = 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`tariffa\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`tariffa\_BEFORE\_INSERT` BEFORE INSERT ON `tariffa` FOR EACH ROW

BEGIN

declare scad date;

declare num\_giorni\_ritardo int;

declare tariffa\_esatta float;

if NEW.durata\_pattuita\_prestito = '1 mese' then

select date\_add(NEW.inizio\_prestito, INTERVAL 30 DAY) into scad;

elseif NEW.durata\_pattuita\_prestito = '2 mesi' then

select date\_add(NEW.inizio\_prestito, INTERVAL 60 DAY) into scad;

elseif NEW.durata\_pattuita\_prestito = '3 mesi' then

select date\_add(NEW.inizio\_prestito, INTERVAL 90 DAY) into scad;

end if;

select datediff(NEW.restituzione, scad) into num\_giorni\_ritardo;

if num\_giorni\_ritardo <= 0 then

signal sqlstate '45000';

elseif num\_giorni\_ritardo <= 10 then

set tariffa\_esatta = num\_giorni\_ritardo\*0.10;

if NEW.valore\_tariffa <> tariffa\_esatta then

signal sqlstate '45000';

end if;

else

set tariffa\_esatta = 1.00 + (num\_giorni\_ritardo - 10)\*0.50;

if NEW.valore\_tariffa <> tariffa\_esatta then

signal sqlstate '45000';

end if;

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`tariffa\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`tariffa\_BEFORE\_UPDATE` BEFORE UPDATE ON `tariffa` FOR EACH ROW

BEGIN

declare scad date;

declare num\_giorni\_ritardo int;

declare tariffa\_esatta float;

if NEW.durata\_pattuita\_prestito = '1 mese' then

select date\_add(NEW.inizio\_prestito, INTERVAL 30 DAY) into scad;

elseif NEW.durata\_pattuita\_prestito = '2 mesi' then

select date\_add(NEW.inizio\_prestito, INTERVAL 60 DAY) into scad;

elseif NEW.durata\_pattuita\_prestito = '3 mesi' then

select date\_add(NEW.inizio\_prestito, INTERVAL 90 DAY) into scad;

end if;

select datediff(NEW.restituzione, scad) into num\_giorni\_ritardo;

if num\_giorni\_ritardo <= 0 then

signal sqlstate '45000';

elseif num\_giorni\_ritardo <= 10 then

set tariffa\_esatta = num\_giorni\_ritardo\*0.10;

if NEW.valore\_tariffa <> tariffa\_esatta then

signal sqlstate '45000';

end if;

else

set tariffa\_esatta = 1.00 + (num\_giorni\_ritardo - 10)\*0.50;

if NEW.valore\_tariffa <> tariffa\_esatta then

signal sqlstate '45000';

end if;

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`contatto\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`contatto\_BEFORE\_INSERT` BEFORE INSERT ON `contatto` FOR EACH ROW

BEGIN

if NEW.mezzo\_comunicazione = 'telefono' or NEW.mezzo\_comunicazione = 'cellulare' then

if not NEW.recapito regexp '^[0-9]{4,15}$' then

signal sqlstate '45000';

end if;

elseif NEW.mezzo\_comunicazione = 'email' then

if not NEW.recapito regexp '^[a-zA-Z0-9][a-zA-Z0-9.\_-]\*[a-zA-Z0-9.\_-]@  
[a-zA-Z0-9][a-zA-Z0-9.\_-]\*[a-zA-Z0-9]\\.[a-zA-Z]{2,63}$' then

signal sqlstate '45000';

end if;

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`contatto\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`contatto\_BEFORE\_UPDATE` BEFORE UPDATE ON `contatto` FOR EACH ROW

BEGIN

if NEW.mezzo\_comunicazione = 'telefono' or NEW.mezzo\_comunicazione = 'cellulare' then

if not NEW.recapito regexp '^[0-9]{4,15}$' then

signal sqlstate '45000';

end if;

elseif NEW.mezzo\_comunicazione = 'email' then

if not NEW.recapito regexp '^[a-zA-Z0-9][a-zA-Z0-9.\_-]\*[a-zA-Z0-9.\_-]@  
[a-zA-Z0-9][a-zA-Z0-9.\_-]\*[a-zA-Z0-9]\\.[a-zA-Z]{2,63}$' then

signal sqlstate '45000';

end if;

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`trasferimento\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`trasferimento\_BEFORE\_INSERT` BEFORE INSERT ON `trasferimento` FOR EACH ROW

BEGIN

declare bibl\_part varchar(45);

select `biblioteca\_ubicazione` from `copia\_di\_libro` where `ID` = NEW.copia\_trasferita  
into bibl\_part;

if bibl\_part = NEW.biblioteca\_destinazione then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`trasferimento\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`trasferimento\_BEFORE\_UPDATE` BEFORE UPDATE ON `trasferimento` FOR EACH ROW

BEGIN

declare bibl\_part varchar(45);

select `biblioteca\_ubicazione` from `copia\_di\_libro` where `ID` = NEW.copia\_trasferita  
into bibl\_part;

if bibl\_part = NEW.biblioteca\_destinazione then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`contatto\_preferito\_BEFORE\_INSERT` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`contatto\_preferito\_BEFORE\_INSERT` BEFORE INSERT ON `contatto\_preferito` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `contatto`

where `utente` = NEW.codice\_fiscale\_utente and `mezzo\_comunicazione` = NEW.mezzo\_comunicazione\_preferito into count;

if count = 0 then

signal sqlstate '45000';

end if;

END$$

USE `CircuitoDiBiblioteche`$$

DROP TRIGGER IF EXISTS `CircuitoDiBiblioteche`.`contatto\_preferito\_BEFORE\_UPDATE` $$

USE `CircuitoDiBiblioteche`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `CircuitoDiBiblioteche`.`contatto\_preferito\_BEFORE\_UPDATE` BEFORE UPDATE ON `contatto\_preferito` FOR EACH ROW

BEGIN

declare count int;

select count(\*) from `contatto`

where `utente` = NEW.codice\_fiscale\_utente and `mezzo\_comunicazione` = NEW.mezzo\_comunicazione\_preferito into count;

if count = 0 then

signal sqlstate '45000';

end if;

END$$

DELIMITER ;

## Eventi

set global event\_scheduler = on;

CREATE event IF NOT EXISTS `cleanup\_penali\_tariffe`

on schedule

every 730 day

on completion preserve

do

delete from `penale` where `data\_restituzione` < (NOW() - interval 730 day);

delete from `tariffa` where `restituzione` < (NOW() - interval 730 day);

set global event\_scheduler = on;

CREATE event IF NOT EXISTS `cleanup\_turni\_malattie`

on schedule

every 30 day

on completion preserve

do

delete from `malattia` where `data\_malattia` < (NOW() - interval 30 day);

delete from `turno` where `data` < (NOW() - interval 30 day);

Entrambi gli eventi sopra riportati sono istanziati solo in fase di configurazione del sistema, dopodiché eseguono periodicamente i loro comandi in automatico.

## Viste

Non sono presenti particolari viste all’interno dell’applicazione.

## Stored Procedures e transazioni

-- -----------------------------------------------------

-- procedure login

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`login`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `login` (in var\_username varchar(45), in var\_pass varchar(45), out var\_role int)

BEGIN

declare var\_user\_role enum('amministratore', 'bibliotecario');

select `ruolo` from `user`

where `username` = var\_username and `password` = md5(var\_pass)

into var\_user\_role;

if var\_user\_role = 'amministratore' then

set var\_role = 1;

elseif var\_user\_role = 'bibliotecario' then

set var\_role = 2;

else

set var\_role = 3;

end if;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_utente

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_utente`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_utente` (in var\_CF varchar(16), in var\_nome varchar(45), in var\_cognome varchar(45), in var\_indirizzo varchar(45), in var\_data\_nascita date)

BEGIN

insert into `utente` values (var\_CF, var\_nome, var\_cognome, var\_data\_nascita, var\_indirizzo);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_contatto

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_contatto`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_contatto` (in var\_recapito varchar(45), in var\_mezzo\_comunicazione varchar(45), in var\_CF varchar(16))

BEGIN

insert into `contatto` values (var\_recapito, var\_mezzo\_comunicazione, var\_CF);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_contatto\_preferito

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_contatto\_preferito`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_contatto\_preferito` (in var\_mezzo\_comunicazione\_preferito varchar(45), in var\_CF varchar(16))

BEGIN

insert into `contatto\_preferito` values (var\_CF, var\_mezzo\_comunicazione\_preferito);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure crea\_user

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`crea\_user`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `crea\_user` (in username varchar(45), in pass varchar(45), in ruolo varchar(45))

BEGIN

insert into `user` values (username, md5(pass), ruolo);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_copia

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_copia`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_copia` (in var\_ISBN varchar(13), in var\_titolo varchar(45), in var\_autore varchar(45), in var\_biblioteca varchar(15), in var\_scaffale int, in var\_ripiano int, out var\_ID int)

BEGIN

declare var\_count int;

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

set transaction isolation level repeatable read;

start transaction;

select count(\*) from `libro` where `ISBN` = var\_ISBN into var\_count;

if var\_count = 0 then

insert into `libro` values (var\_ISBN, var\_titolo, var\_autore);

end if;

insert into `copia\_di\_libro` (`biblioteca\_ubicazione`, `libro`, `stato`, `scaffale`, `ripiano`, `data\_ultima\_restituzione\_copia`) values (var\_biblioteca, var\_ISBN, 'disponibile', var\_scaffale, var\_ripiano, curdate());

set var\_ID = last\_insert\_id();

select count(\*) from `disponibilità` where `numero\_biblioteca` = var\_biblioteca and `ISBN\_libro` = var\_ISBN into var\_count;

if var\_count = 0 then

insert into `disponibilità` values (var\_biblioteca, var\_ISBN, 1, 1, curdate());

else

update `disponibilità` set `quantità\_totale` = `quantità\_totale`+1  
where `numero\_biblioteca` = var\_biblioteca and `ISBN\_libro` = var\_ISBN;

update `disponibilità` set `quantità` = `quantità`+1 where  
`numero\_biblioteca` = var\_biblioteca and `ISBN\_libro` = var\_ISBN;

end if;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure termina\_prestito

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`termina\_prestito`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `termina\_prestito` (in var\_copia int, out tar float)

BEGIN

declare var\_libro varchar(13);

declare var\_biblioteca\_partenza varchar(15);

declare var\_utente varchar(16);

declare var\_data\_inizio date;

declare var\_durata varchar(45);

declare scad date;

declare num\_giorni\_ritardo int;

declare var\_count int;

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

set tar = 0;

set transaction isolation level repeatable read;

start transaction;

select count(\*) from `copia\_di\_libro` where `ID` = var\_copia into var\_count;

if var\_count = 0 then

signal sqlstate '45001';

end if;

select `libro`, `biblioteca\_ubicazione` from `copia\_di\_libro` where `ID` = var\_copia into var\_libro, var\_biblioteca\_partenza;

update `copia\_di\_libro` set `stato` = 'disponibile' where `ID` = var\_copia;

update `copia\_di\_libro` set `data\_ultima\_restituzione\_copia` = curdate()  
where `ID` = var\_copia;

update `disponibilità` set `quantità` = `quantità`+1 where  
`numero\_biblioteca` = var\_biblioteca\_partenza and `ISBN\_libro` = var\_libro;

update `disponibilità` set `data\_ultima\_restituzione\_libro` = curdate()  
where `numero\_biblioteca` = var\_biblioteca\_partenza and `ISBN\_libro` = var\_libro;

select `utente\_destinatario`, `data\_inizio`, `durata\_prevista` from `prestito`  
where `copia\_prestata` = var\_copia into var\_utente, var\_data\_inizio, var\_durata;

delete from `prestito` where `copia\_prestata` = var\_copia;

delete from `trasferimento` where `copia\_trasferita` = var\_copia;

if var\_durata = '1 mese' then

select date\_add(var\_data\_inizio, interval 30 day) into scad;

elseif var\_durata = '2 mesi' then

select date\_add(var\_data\_inizio, interval 60 day) into scad;

elseif var\_durata = '3 mesi' then

select date\_add(var\_data\_inizio, interval 90 day) into scad;

end if;

select datediff(curdate(), scad) into num\_giorni\_ritardo;

select count(\*) from `tariffa`

where `inizio\_prestito` = var\_data\_inizio and `restituzione` = curdate() and `durata\_pattuita\_prestito` = var\_durata into var\_count;

if num\_giorni\_ritardo > 0 and num\_giorni\_ritardo <= 10 then

set tar = num\_giorni\_ritardo\*0.10;

if var\_count = 0 then

insert into `tariffa` values (var\_data\_inizio, curdate(), var\_durata, tar);

end if;

insert into `penale` values (var\_copia, var\_utente, var\_data\_inizio, curdate(), var\_durata);

elseif num\_giorni\_ritardo > 10 then

set tar = 1.00 + (num\_giorni\_ritardo - 10)\*0.50;

if var\_count = 0 then

insert into `tariffa` values (var\_data\_inizio, curdate(), var\_durata, tar);

end if;

insert into `penale` values (var\_copia, var\_utente, var\_data\_inizio, curdate(), var\_durata);

end if;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure report\_prestiti

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`report\_prestiti`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `report\_prestiti` (in var\_biblioteca\_corrente varchar(15))

BEGIN

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

drop temporary table if exists `info\_copie\_prestate`;

create temporary table `info\_copie\_prestate` (

`ID copia` int,

`ISBN libro` varchar(13),

`titolo libro` varchar(45),

`autore libro` varchar(45),

`data` date,

`durata prevista` varchar(45),

`CF utente` varchar(16)

);

set transaction isolation level repeatable read;

start transaction;

insert into `info\_copie\_prestate`

select `ID`, `ISBN`, `titolo`, `autore`, `data\_inizio`, `durata\_prevista`, `utente\_destinatario`

from `copia\_di\_libro` join `libro` on `libro` = `ISBN` join `prestito`  
on `ID` = `copia\_prestata`

where `stato` = 'in prestito' and `biblioteca\_ubicazione` = var\_biblioteca\_corrente;

insert into `info\_copie\_prestate`

select `ID`, `ISBN`, `titolo`, `autore`, `data\_inizio`, `durata\_prevista`, `utente\_destinatario`

from `copia\_di\_libro` join `libro` on `libro` = `ISBN` join `prestito`  
on `ID` = `copia\_prestata` join `trasferimento` on `ID` = `copia\_trasferita`

where `stato` = 'trasferita' and `biblioteca\_destinazione` = var\_biblioteca\_corrente;

select \* from `info\_copie\_prestate` order by `data`, `durata prevista`;

select distinct `CF\_utente` as `CF utente`, `nome\_utente` as `nome`, `cognome\_utente` as `cognome`, `data\_nascita\_utente` as `nascita`, `indirizzo\_utente` as `indirizzo`, `mezzo\_comunicazione\_preferito` as `mezzo comunicaz preferito`

from `info\_copie\_prestate` join `utente` on `CF utente` = `CF\_utente` join `contatto\_preferito` on `CF\_utente` = `codice\_fiscale\_utente`

order by `CF\_utente`;

select distinct `CF utente`, `recapito`, `mezzo\_comunicazione` as  
`mezzo comunicazione`

from `info\_copie\_prestate` join `contatto` on `CF utente` = `utente`

order by `CF utente`;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure dismetti\_copie

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`dismetti\_copie`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `dismetti\_copie` ()

BEGIN

declare giorni\_alla\_dismissione int;

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

drop temporary table if exists `libri\_da\_dismettere`;

create temporary table `libri\_da\_dismettere` (

`ISBN libro` varchar(13),

`numero biblioteca` varchar(15)

);

set transaction isolation level read committed;

start transaction;

insert into `libri\_da\_dismettere`

select `ISBN\_libro`, `numero\_biblioteca` from `disponibilità`

where `quantità` > 0 and `quantità` = `quantità\_totale` and `data\_ultima\_restituzione\_libro` < (now() - interval 3652 day);

update `copia\_di\_libro` set `stato` = 'dismessa' where(`libro`, `biblioteca\_ubicazione`) in (select \* from `libri\_da\_dismettere`);

delete from `disponibilità` where (`ISBN\_libro`, `numero\_biblioteca`) in  
(select \* from `libri\_da\_dismettere`);

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure ottieni\_ISBN

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`ottieni\_ISBN`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `ottieni\_ISBN` (in var\_titolo varchar(45), in var\_autore varchar(45))

BEGIN

set transaction read only;

set transaction isolation level read committed;

select `ISBN` from `libro` where `titolo` = var\_titolo and `autore` = var\_autore  
order by `ISBN`;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_turno

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_turno`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_turno` (in var\_bibliotecario varchar(16), in var\_data date, in var\_orario\_inizio time, in var\_orario\_fine time)

BEGIN

insert into `turno` values(var\_bibliotecario, var\_data, var\_orario\_inizio, var\_orario\_fine);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure report\_biblioteche\_scoperte

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`report\_biblioteche\_scoperte`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `report\_biblioteche\_scoperte` (in data1 date, in data2 date)

BEGIN

declare var\_data date;

declare var\_biblioteca varchar(15);

declare var\_indirizzo varchar(45);

declare var\_responsabile varchar(45);

declare var\_giorno\_sett int;

declare var\_inizio\_turno time;

declare var\_fine\_turno time;

declare var\_fine\_buco time;

declare var\_count int;

declare done int default false;

declare cur\_bibl cursor for select `numero\_telefono`, `indirizzo\_biblioteca`, `responsabile` from `biblioteca`;

declare cur\_turni cursor for select `orario\_inizio\_turno`, `orario\_fine\_turno`  
from `turno` join `bibliotecario` on `bibliotecario` = `CF\_bibliotecario`  
where `data` = var\_data and `biblioteca\_impiego` = var\_biblioteca;

declare continue handler for not found set done = true;

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

if datediff(data2, data1) < 0 then

signal sqlstate '45001';

end if;

drop temporary table if exists `biblioteca\_scoperta`;

create temporary table `biblioteca\_scoperta` (

`data` date,

`ora inizio buco` time,

`ora fine buco` time,

`telefono bibl` varchar(15),

`indirizzo bibl` varchar(45),

`responsabile bibl` varchar(45)

);

set var\_data = data1;

set transaction isolation level repeatable read;

start transaction;

date\_loop: loop

select weekday(var\_data) into var\_giorno\_sett;

open cur\_bibl;

bibl\_loop: loop

fetch cur\_bibl into var\_biblioteca, var\_indirizzo, var\_responsabile;

if done then

leave bibl\_loop;

end if;

insert into `biblioteca\_scoperta`

select var\_data, `orario\_inizio\_apertura`, `orario\_fine\_apertura`, var\_biblioteca, var\_indirizzo, var\_responsabile

from `apertura` where `biblioteca` = var\_biblioteca and `giorno\_settimanale` = var\_giorno\_sett;

open cur\_turni;

turni\_loop: loop

fetch cur\_turni into var\_inizio\_turno, var\_fine\_turno;

if done then

leave turni\_loop;

end if;

select count(\*) from `biblioteca\_scoperta`

where time\_to\_sec(timediff(var\_inizio\_turno, ora inizio buco)) >0 and time\_to\_sec(timediff(ora fine buco, var\_fine\_turno)) >0

and `data` = var\_data and `telefono bibl` = var\_biblioteca

into var\_count;

if var\_count > 0 then

select `ora fine buco` from `biblioteca\_scoperta`

where time\_to\_sec(timediff(var\_inizio\_turno,  
`ora inizio buco`)) > 0 and time\_to\_sec(timediff  
(`ora fine buco`, var\_fine\_turno)) > 0

and `data` = var\_data and `telefono bibl` = var\_biblioteca

into var\_fine\_buco;

insert into `biblioteca\_scoperta` values(var\_data, var\_fine\_turno, var\_fine\_buco, var\_biblioteca, var\_indirizzo, var\_responsabile);

update `biblioteca\_scoperta` set `ora fine buco` = var\_inizio\_turno

where time\_to\_sec(timediff(var\_inizio\_turno,  
`ora inizio buco`)) > 0 and time\_to\_sec(timediff  
(`ora fine buco`, var\_fine\_turno)) > 0

and `data` = var\_data and `telefono bibl` = var\_biblioteca;

end if;

delete from `biblioteca\_scoperta` where time\_to\_sec(timediff(var\_inizio\_turno, `ora inizio buco`)) <= 0 and time\_to\_sec(timediff(`ora fine buco`, var\_fine\_turno)) <= 0

and `data` = var\_data and `telefono bibl` = var\_biblioteca;

update `biblioteca\_scoperta` set `ora inizio buco` = var\_fine\_turno

where time\_to\_sec(timediff(var\_inizio\_turno, `ora inizio buco`)) <= 0 and time\_to\_sec(timediff(`ora fine buco`, var\_fine\_turno)) > 0 and time\_to\_sec(timediff(var\_fine\_turno, `ora inizio buco`)) > 0

and `data` = var\_data and `telefono bibl` = var\_biblioteca;

update `biblioteca\_scoperta` set `ora fine buco` = var\_inizio\_turno

where time\_to\_sec(timediff(var\_inizio\_turno, `ora inizio buco`)) > 0 and time\_to\_sec(timediff(`ora fine buco`, var\_fine\_turno)) <= 0 and time\_to\_sec(timediff (var\_inizio\_turno, `ora fine buco`)) < 0

and `data` = var\_data and `telefono bibl` = var\_biblioteca;

end loop;

close cur\_turni;

set done = false;

end loop;

close cur\_bibl;

set done = false;

select date\_add(var\_data, interval 1 day) into var\_data;

if datediff(data2, var\_data) < 0 then

leave date\_loop;

end if;

end loop;

select \* from `biblioteca\_scoperta`;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_bibliotecario

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_bibliotecario`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_bibliotecario` (in var\_CF varchar(16), in var\_nome varchar(45), in var\_cognome varchar(45), in var\_luogo\_nascita varchar(45), in var\_titolo\_studio varchar(45), in var\_biblioteca varchar(15), in var\_username varchar(45), in var\_data\_nascita date)

BEGIN

insert into `bibliotecario` values (var\_CF, var\_nome, var\_cognome, var\_data\_nascita, var\_luogo\_nascita, var\_titolo\_studio, var\_biblioteca, var\_username);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure inizia\_prestito

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`inizia\_prestito`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `inizia\_prestito` (in var\_libro varchar(13), in var\_biblioteca\_corrente varchar(15), in var\_utente varchar(16), in var\_durata varchar(45))

BEGIN

declare var\_ID int;

declare var\_count int;

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

drop temporary table if exists `da\_trasferire`;

create temporary table `da\_trasferire` (

`da trasferire` smallint

);

set transaction isolation level repeatable read;

start transaction;

select count(\*) from `copia\_di\_libro`

where `biblioteca\_ubicazione` = var\_biblioteca\_corrente and `libro` = var\_libro  
and `stato` = 'disponibile' into var\_count;

if var\_count > 0 then

insert into `da\_trasferire` values ('0');

select \* from `da\_trasferire`;

select min(`ID`) from `copia\_di\_libro`

where `biblioteca\_ubicazione` = var\_biblioteca\_corrente and `libro` = var\_libro and `stato` = 'disponibile' into var\_ID;

select `ID`, `scaffale`, `ripiano` from `copia\_di\_libro` where `ID` = var\_ID;

update `copia\_di\_libro` set `stato` = 'in prestito' where `ID` = var\_ID;

insert into `prestito` values (var\_ID, var\_utente, curdate(), var\_durata);

update `disponibilità` set `quantità` = `quantità`-1 where `numero\_biblioteca` = var\_biblioteca\_corrente and `ISBN\_libro` = var\_libro;

else

insert into `da\_trasferire` values ('1');

select \* from `da\_trasferire`;

select `numero\_telefono` as `telefono bibl`, `indirizzo\_biblioteca` as  
`indirizzo bibl`, `responsabile` as `responsabile bibl`

from `disponibilità` join `biblioteca` on `numero\_biblioteca` = `numero\_telefono` where `ISBN\_libro` = var\_libro and `quantità` > 0;

end if;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure inizia\_trasferimento

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`inizia\_trasferimento`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `inizia\_trasferimento` (in var\_libro varchar(13), in var\_biblioteca\_destinazione varchar(15), in var\_biblioteca\_partenza varchar(15), in var\_utente varchar(16), in var\_durata varchar(45))

BEGIN

declare var\_ID int;

declare var\_count int;

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

set transaction isolation level repeatable read;

start transaction;

select count(\*) from `copia\_di\_libro`

where `biblioteca\_ubicazione` = var\_biblioteca\_partenza and `libro` = var\_libro and `stato` = 'disponibile' into var\_count;

if var\_count > 0 then

select min(`ID`) from `copia\_di\_libro`

where `biblioteca\_ubicazione` = var\_biblioteca\_partenza and `libro` = var\_libro and `stato` = 'disponibile' into var\_ID;

select `ID`, `scaffale`, `ripiano` from `copia\_di\_libro` where `ID` = var\_ID;

update `copia\_di\_libro` set `stato` = 'trasferita' where `ID` = var\_ID;

insert into `prestito` values (var\_ID, var\_utente, curdate(), var\_durata);

insert into `trasferimento` values (var\_ID, var\_biblioteca\_destinazione);

update `disponibilità` set `quantità` = `quantità`-1

where `numero\_biblioteca` = var\_biblioteca\_partenza and  
`ISBN\_libro` = var\_libro;

else

signal sqlstate '45001';

end if;

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure trova\_sostituti

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`trova\_sostituti`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `trova\_sostituti` (in bibliotecario\_malato varchar(16), in data\_mal date)

BEGIN

declare exit handler for sqlexception

begin

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exception to the caller

end;

set transaction isolation level read committed;

start transaction;

select `CF\_bibliotecario` as `CF bibliotecario`, `nome\_bibliotecario` as `nome`, `cognome\_bibliotecario` as `cognome`, `data\_nascita\_bibliotecario` as `nascita`, `luogo\_nascita` as `luogo nascita`, `titolo\_studio` as `titolo di studio`

from `bibliotecario` where `biblioteca\_impiego` = (select b.`biblioteca\_impiego`  
from `bibliotecario` b where b.`CF\_bibliotecario` = bibliotecario\_malato)

and `CF\_bibliotecario` not in (select `bibliotecario` from `turno` where `data` = data\_mal)

and `CF\_bibliotecario` not in (select `bibliotecario\_sostituto` from `malattia`  
where `data\_malattia` = data\_mal);

commit;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure aggiungi\_malattia

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`aggiungi\_malattia`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `aggiungi\_malattia` (in bibliotecario\_malato varchar(16), in motivo varchar(45), in bibliotecario\_sostituto varchar(16), in data\_malattia date)

BEGIN

insert into `malattia` values (bibliotecario\_malato, data\_malattia, motivo, bibliotecario\_sostituto);

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure ottieni\_biblioteca\_impiego

-- -----------------------------------------------------

USE `CircuitoDiBiblioteche`;

DROP procedure IF EXISTS `CircuitoDiBiblioteche`.`ottieni\_biblioteca\_impiego`;

DELIMITER $$

USE `CircuitoDiBiblioteche`$$

CREATE PROCEDURE `ottieni\_biblioteca\_impiego` (in username varchar(45), out biblioteca varchar(15))

BEGIN

set transaction read only;

set transaction isolation level read committed;

select `biblioteca\_impiego` from `bibliotecario`

where `username\_bibliotecario` = username into biblioteca;

commit;

END$$

DELIMITER ;

USE `CircuitoDiBiblioteche`;

**Commenti sulle transazioni**

**aggiungi\_copia**

Il livello di isolamento scelto per la transazione di *aggiungi\_copia* è repeatable read per evitare che l’eventuale tupla che viene conteggiata nell’istruzione *select count (\*) from `disponibilità` where `numero\_biblioteca` = var\_biblioteca and `ISBN\_libro` = var\_ISBN* venga eliminata prima delle istruzioni di update nel ramo else.

**dismetti\_copie**

Il livello di isolamento scelto per la transazione di *dismetti\_copie* è read è read committed perché si tratta di una stored procedure che effettua un’unica lettura: perciò, da una parte può soffrire dell’anomalia della lettura sporca, ma dall’altra non deve eseguire delle letture ripetute (per cui il livello repeatable read abbasserebbe inutilmente le performance della procedura stessa).

**inizia\_prestito**

Il livello di isolamento scelto per la transazione di *inizia\_prestito* è repeatable read per evitare che le eventuali tuple conteggiate nell’istruzione *select count (\*) from `copia\_di\_libro` where `biblioteca\_ubicazione` = var\_biblioteca\_corrente and `libro` = var\_libro and `stato` = ‘disponibile’* vengano eliminate prima delle istruzioni relative al ramo if.

**inizia\_trasferimento**

Il livello di isolamento scelto per la transazione di *inizia\_trasferimento* è repeatable read per motivi del tutto analoghi alla transazione di *inizia\_prestito*.

**ottieni\_biblioteca\_impiego**

Il livello di isolamento scelto per la transazione di *ottieni\_biblioteca\_impiego* è read committed per motivi del tutto analoghi alla transazione di *dismetti\_copie*.

**ottieni\_ISBN**

Il livello di isolamento scelto per la transazione di *ottieni\_ISBN* è read committed per motivi del tutto analoghi alla transazione di *dismetti\_copie*.

**report\_biblioteche\_scoperte**

Il livello di isolamento scelto per la transazione di *report\_biblioteche\_scoperte* è repeatable read perché le biblioteche con cui è stato inizializzato il cursore *cur\_bibl* potrebbero cambiare il loro numero di telefono prima che la stored procedure entri nei cicli più interni e invochi l’operazione *insert into `biblioteca\_scoperta` select var\_data, `orario\_inizio\_apertura`, `orario\_fine\_apertura`, var\_biblioteca, var\_indirizzo, var\_responsabile from `apertura` where `biblioteca` = var\_biblioteca and `giorno\_settimanale` = var\_giorno\_sett*

Questo scenario potrebbe portare a un’inconsistenza della query in quanto, nel cursore, il numero di telefono della biblioteca modificata risulterebbe non aggiornato, a differenza di quanto avviene nelle relazioni *apertura* e *bibliotecario*.

**report\_prestiti**

Il livello di isolamento scelto per la transazione di *report\_prestiti* è repeatable read poiché, altrimenti, si avrebbe un’inconsistenza nel caso in cui le tuple relative a qualche utente del circuito di biblioteche venissero modificate o eliminate prima della seguente operazione:

*select distinct `CF\_utente` as `CF utente`, `nome\_utente` as `nome`, `cognome\_utente` as `cognome`, `data\_nascita\_utente` as `nascita`, `indirizzo\_utente` as `indirizzo`, `mezzo\_comunicazione\_preferito` as `mezzo comunicaz preferito` from `info\_copie\_prestate` join `utente` on `CF utente` = `CF\_utente` join `contatto\_preferito` on `CF\_utente` = `codice\_fiscale\_utente` order by `CF\_utente`*

e dopo la seguente operazione:

*select distinct `CF utente`, `recapito`, `mezzo\_comunicazione` as `mezzo comunicazione` from `info\_copie\_prestate` join `contatto` on `CF utente` = `utente` order by `CF utente`*

**termina\_prestito**

Il livello di isolamento scelto per la transazione di *termina\_prestito* è repeatable read perché l’ID di una copia di libro potrebbe essere modificato nell’intermezzo delle seguenti operazioni:

*select `libro`, `biblioteca\_ubicazione` from `copia\_di\_libro` where `ID` = var\_copia into var\_libro, var\_biblioteca\_partenza;*

*update `copia\_di\_libro` set `stato` = 'disponibile' where `ID` = var\_copia;*

*update `copia\_di\_libro` set `data\_ultima\_restituzione\_copia` = curdate () where `ID` = var\_copia*

Un discorso analogo è valido per la relazione *disponibilità*.

**trova\_sostituti**

Il livello di isolamento scelto per la transazione di *trova\_sostituti* è read committed per motivi del tutto analoghi alla transazione di *dismetti\_copie*.

Per concludere, osserviamo che nessuna transazione ha come livello di isolamento serializable. In effetti, non è stato identificato alcun possibile scenario caratterizzato dall’anomalia dell’inserimento fantasma. Ciò, a grandi linee, è dovuto al fatto che le uniche istruzioni SQL con operatori aggregati presenti all’interno delle procedure contengono esclusivamente l’operatore *count (\*),* non tanto con lo scopo di dare un numero esatto di elementi, bensì di constatarne la presenza o meno. Inoltre, nel caso in cui *count (\*)* restituisse zero, non verrebbero effettuate altre operazioni di lettura sulle medesime relazioni coinvolte da *count (\*),* per cui non ci sarebbero possibilità di andare incontro a un’inconsistenza causata da un inserimento fantasma.

# Appendice: Implementazione

## Codice SQL per instanziare il database

-- -----------------------------------------------------

-- Schema CircuitoDiBiblioteche

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `CircuitoDiBiblioteche` ;

-- -----------------------------------------------------

-- Schema CircuitoDiBiblioteche

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `CircuitoDiBiblioteche` DEFAULT CHARACTER SET utf8;

USE `CircuitoDiBiblioteche` ;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`biblioteca`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`biblioteca` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`biblioteca` (

`numero\_telefono` VARCHAR(15) NOT NULL,

`indirizzo\_biblioteca` VARCHAR(45) NOT NULL,

`responsabile` VARCHAR(45) NOT NULL,

PRIMARY KEY (`numero\_telefono`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`libro`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`libro` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`libro` (

`ISBN` VARCHAR(13) NOT NULL,

`titolo` VARCHAR(45) NOT NULL,

`autore` VARCHAR(45) NOT NULL,

PRIMARY KEY (`ISBN`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`utente`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`utente` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`utente` (

`CF\_utente` VARCHAR(16) NOT NULL,

`nome\_utente` VARCHAR(45) NOT NULL,

`cognome\_utente` VARCHAR(45) NOT NULL,

`data\_nascita\_utente` DATE NOT NULL,

`indirizzo\_utente` VARCHAR(45) NOT NULL,

PRIMARY KEY (`CF\_utente`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`user`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`user` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`user` (

`username` VARCHAR(45) NOT NULL,

`password` CHAR(32) NOT NULL,

`ruolo` ENUM('amministratore', 'bibliotecario') NOT NULL,

PRIMARY KEY (`username`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`bibliotecario`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`bibliotecario` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`bibliotecario` (

`CF\_bibliotecario` VARCHAR(16) NOT NULL,

`nome\_bibliotecario` VARCHAR(45) NOT NULL,

`cognome\_bibliotecario` VARCHAR(45) NOT NULL,

`data\_nascita\_bibliotecario` DATE NOT NULL,

`luogo\_nascita` VARCHAR(45) NOT NULL,

`titolo\_studio` VARCHAR(45) NOT NULL,

`biblioteca\_impiego` VARCHAR(15) NOT NULL,

`username\_bibliotecario` VARCHAR(45) NOT NULL,

PRIMARY KEY (`CF\_bibliotecario`),

CONSTRAINT `biblioteca\_impiego`

FOREIGN KEY (`biblioteca\_impiego`)

REFERENCES `CircuitoDiBiblioteche`.`biblioteca` (`numero\_telefono`)

ON DELETE NO ACTION

ON UPDATE CASCADE,

CONSTRAINT `username\_bibliotecario`

FOREIGN KEY (`username\_bibliotecario`)

REFERENCES `CircuitoDiBiblioteche`.`user` (`username`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `biblioteca\_impiego\_idx` ON `CircuitoDiBiblioteche`.`bibliotecario` (`biblioteca\_impiego` ASC) VISIBLE;

CREATE UNIQUE INDEX `username\_bibliotecario\_UNIQUE` ON `CircuitoDiBiblioteche`.`bibliotecario` (`username\_bibliotecario` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`turno`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`turno` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`turno` (

`bibliotecario` VARCHAR(16) NOT NULL,

`data` DATE NOT NULL,

`orario\_inizio\_turno` TIME NOT NULL,

`orario\_fine\_turno` TIME NOT NULL,

PRIMARY KEY (`bibliotecario`, `data`),

CONSTRAINT `bibliotecario`

FOREIGN KEY (`bibliotecario`)

REFERENCES `CircuitoDiBiblioteche`.`bibliotecario` (`CF\_bibliotecario`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`malattia`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`malattia` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`malattia` (

`bibliotecario\_malato` VARCHAR(16) NOT NULL,

`data\_malattia` DATE NOT NULL,

`motivo` VARCHAR(45) NOT NULL,

`bibliotecario\_sostituto` VARCHAR(16) NOT NULL,

PRIMARY KEY (`bibliotecario\_malato`, `data\_malattia`),

UNIQUE (`data\_malattia`, `bibliotecario\_sostituto`),

CONSTRAINT `bibliotecario\_sostituto`

FOREIGN KEY (`bibliotecario\_sostituto`)

REFERENCES `CircuitoDiBiblioteche`.`bibliotecario` (`CF\_bibliotecario`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `bibliotecario\_sostituto\_idx` ON `CircuitoDiBiblioteche`.`malattia` (`bibliotecario\_sostituto` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`apertura`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`apertura` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`apertura` (

`biblioteca` VARCHAR(15) NOT NULL,

`giorno\_settimanale` INT UNSIGNED NOT NULL,

`orario\_inizio\_apertura` TIME NOT NULL,

`orario\_fine\_apertura` TIME NOT NULL,

PRIMARY KEY (`biblioteca`, `giorno\_settimanale`),

CONSTRAINT `biblioteca`

FOREIGN KEY (`biblioteca`)

REFERENCES `CircuitoDiBiblioteche`.`biblioteca` (`numero\_telefono`)

ON DELETE NO ACTION

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`disponibilità`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`disponibilità` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`disponibilità` (

`numero\_biblioteca` VARCHAR(15) NOT NULL,

`ISBN\_libro` VARCHAR(13) NOT NULL,

`quantità` INT UNSIGNED NOT NULL,

`quantità\_totale` INT UNSIGNED NOT NULL,

`data\_ultima\_restituzione\_libro` DATE NULL,

PRIMARY KEY (`numero\_biblioteca`, `ISBN\_libro`),

CONSTRAINT `numero\_biblioteca`

FOREIGN KEY (`numero\_biblioteca`)

REFERENCES `CircuitoDiBiblioteche`.`biblioteca` (`numero\_telefono`)

ON DELETE NO ACTION

ON UPDATE CASCADE,

CONSTRAINT `ISBN\_libro`

FOREIGN KEY (`ISBN\_libro`)

REFERENCES `CircuitoDiBiblioteche`.`libro` (`ISBN`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `ISBN\_libro\_idx` ON `CircuitoDiBiblioteche`.`disponibilità` (`ISBN\_libro` ASC) VISIBLE;

CREATE INDEX `data\_ultima\_restituzione\_libro\_idx` ON `CircuitoDiBiblioteche`.`disponibilità` (`data\_ultima\_restituzione\_libro` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`copia\_di\_libro`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`copia\_di\_libro` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`copia\_di\_libro` (

`ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`biblioteca\_ubicazione` VARCHAR(15) NOT NULL,

`libro` VARCHAR(13) NOT NULL,

`stato` ENUM('disponibile', 'in prestito', 'trasferita', 'dismessa') NOT NULL,

`scaffale` INT UNSIGNED NULL,

`ripiano` INT UNSIGNED NULL,

`data\_ultima\_restituzione\_copia` DATE NULL,

PRIMARY KEY (`ID`),

CONSTRAINT `biblioteca\_ubicazione`

FOREIGN KEY (`biblioteca\_ubicazione`)

REFERENCES `CircuitoDiBiblioteche`.`biblioteca` (`numero\_telefono`)

ON DELETE NO ACTION

ON UPDATE CASCADE,

CONSTRAINT `libro`

FOREIGN KEY (`libro`)

REFERENCES `CircuitoDiBiblioteche`.`libro` (`ISBN`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `biblioteca\_ubicazione\_idx` ON `CircuitoDiBiblioteche`.`copia\_di\_libro` (`biblioteca\_ubicazione` ASC) VISIBLE;

CREATE INDEX `libro\_idx` ON `CircuitoDiBiblioteche`.`copia\_di\_libro` (`libro` ASC) VISIBLE;

CREATE INDEX `stato\_idx` ON `CircuitoDiBiblioteche`.`copia\_di\_libro` (`stato` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`prestito`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`prestito` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`prestito` (

`copia\_prestata` INT UNSIGNED NOT NULL,

`utente\_destinatario` VARCHAR(16) NOT NULL,

`data\_inizio` DATE NOT NULL,

`durata\_prevista` ENUM('1 mese', '2 mesi', '3 mesi') NOT NULL,

PRIMARY KEY (`copia\_prestata`),

CONSTRAINT `copia\_prestata`

FOREIGN KEY (`copia\_prestata`)

REFERENCES `CircuitoDiBiblioteche`.`copia\_di\_libro` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `utente\_destinatario`

FOREIGN KEY (`utente\_destinatario`)

REFERENCES `CircuitoDiBiblioteche`.`utente` (`CF\_utente`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `utente\_destinatario\_idx` ON `CircuitoDiBiblioteche`.`prestito` (`utente\_destinatario` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`penale`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`penale` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`penale` (

`copia` INT UNSIGNED NOT NULL,

`utente\_pagante` VARCHAR(16) NOT NULL,

`data\_inizio\_prestito` DATE NOT NULL,

`data\_restituzione` DATE NOT NULL,

`durata\_prevista\_prestito` ENUM('1 mese', '2 mesi', '3 mesi') NOT NULL,

PRIMARY KEY (`copia`, `utente\_pagante`, `data\_inizio\_prestito`),

CONSTRAINT `copia`

FOREIGN KEY (`copia`)

REFERENCES `CircuitoDiBiblioteche`.`copia\_di\_libro` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `utente\_pagante`

FOREIGN KEY (`utente\_pagante`)

REFERENCES `CircuitoDiBiblioteche`.`utente` (`CF\_utente`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `utente\_pagante\_idx` ON `CircuitoDiBiblioteche`.`penale` (`utente\_pagante` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`tariffa`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`tariffa` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`tariffa` (

`inizio\_prestito` DATE NOT NULL,

`restituzione` DATE NOT NULL,

`durata\_pattuita\_prestito` ENUM('1 mese', '2 mesi', '3 mesi') NOT NULL,

`valore\_tariffa` FLOAT UNSIGNED NOT NULL,

PRIMARY KEY (`inizio\_prestito`, `restituzione`, `durata\_pattuita\_prestito`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`contatto`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`contatto` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`contatto` (

`recapito` VARCHAR(45) NOT NULL,

`mezzo\_comunicazione` ENUM('telefono', 'cellulare', 'email') NOT NULL,

`utente` VARCHAR(16) NOT NULL,

PRIMARY KEY (`recapito`),

CONSTRAINT `utente`

FOREIGN KEY (`utente`)

REFERENCES `CircuitoDiBiblioteche`.`utente` (`CF\_utente`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `utente\_idx` ON `CircuitoDiBiblioteche`.`contatto` (`utente` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`trasferimento`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`trasferimento` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`trasferimento` (

`copia\_trasferita` INT UNSIGNED NOT NULL,

`biblioteca\_destinazione` VARCHAR(15) NOT NULL,

PRIMARY KEY (`copia\_trasferita`),

CONSTRAINT `copia\_trasferita`

FOREIGN KEY (`copia\_trasferita`)

REFERENCES `CircuitoDiBiblioteche`.`copia\_di\_libro` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `biblioteca\_destinazione`

FOREIGN KEY (`biblioteca\_destinazione`)

REFERENCES `CircuitoDiBiblioteche`.`biblioteca` (`numero\_telefono`)

ON DELETE NO ACTION

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `biblioteca\_destinazione\_idx` ON `CircuitoDiBiblioteche`.`trasferimento` (`biblioteca\_destinazione` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `CircuitoDiBiblioteche`.`contatto\_preferito`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `CircuitoDiBiblioteche`.`contatto\_preferito` ;

CREATE TABLE IF NOT EXISTS `CircuitoDiBiblioteche`.`contatto\_preferito` (

`codice\_fiscale\_utente` VARCHAR(16) NOT NULL,

`mezzo\_comunicazione\_preferito` ENUM('telefono', 'cellulare', 'email') NOT NULL,

PRIMARY KEY (`codice\_fiscale\_utente`))

ENGINE = InnoDB;

## Codice del Front-End

**defines.h**

#pragma once

#include <stdbool.h>

#include <mysql.h>

struct configuration {

char\* host;

char\* db\_username;

char\* db\_password;

unsigned int port;

char\* database;

char username[128];

char password[128];

};

struct param\_type {

char\* varchar[7];

MYSQL\_TIME\* date[2];

MYSQL\_TIME\* time[2];

int\* integer[3];

float\* f;

short int\* b;

int numVarchar;

int numDate;

int numTime;

int numInt;

int numFloat;

int numBool;

};

extern struct configuration conf;

extern struct param\_type paramStruct;

extern int parse\_config(char\* path, struct configuration\* conf);

extern bool yesOrNo(char\* domanda, char yes, char no, bool predef, bool insensitive);

extern char multiChoice(char\* domanda, char choices[], int num);

extern void insertPassword(char password[]);

extern void print\_error(MYSQL\* conn, char\* message);

extern void print\_stmt\_error(MYSQL\_STMT\* stmt, char\* message);

extern void finish\_with\_error(MYSQL\* conn, char\* message);

extern void finish\_with\_stmt\_error(MYSQL\* conn, MYSQL\_STMT\* stmt, char\* message, bool close\_stmt);

extern bool setup\_prepared\_stmt(MYSQL\_STMT\*\* stmt, char\* statement, MYSQL\* conn);

extern void dump\_result\_set(MYSQL\* conn, MYSQL\_STMT\* stmt, char\* title);

extern void bind\_par(MYSQL\_BIND param[], struct param\_type\* paramStruct);

extern char\* fixString(char string[]);

extern void run\_as\_librarian(MYSQL\* conn);

extern void run\_as\_administrator(MYSQL\* conn);

**parse.c**

#include <stddef.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

#define BUFF\_SIZE 4096

// The final config struct will point into this

static char config[BUFF\_SIZE];

/\*\*

\* JSON type identifier. Basic types are:

\* o Object

\* o Array

\* o String

\* o Other primitive: number, boolean (true/false) or null

\*/

typedef enum {

JSMN\_UNDEFINED = 0,

JSMN\_OBJECT = 1,

JSMN\_ARRAY = 2,

JSMN\_STRING = 3,

JSMN\_PRIMITIVE = 4

} jsmntype\_t;

enum jsmnerr {

/\* Not enough tokens were provided \*/

JSMN\_ERROR\_NOMEM = -1,

/\* Invalid character inside JSON string \*/

JSMN\_ERROR\_INVAL = -2,

/\* The string is not a full JSON packet, more bytes expected \*/

JSMN\_ERROR\_PART = -3

};

/\*\*

\* JSON token description.

\* type type (object, array, string etc.)

\* start start position in JSON data string

\* end end position in JSON data string

\*/

typedef struct {

jsmntype\_t type;

int start;

int end;

int size;

#ifdef JSMN\_PARENT\_LINKS

int parent;

#endif

} jsmntok\_t;

/\*\*

\* JSON parser. Contains an array of token blocks available. Also stores

\* the string being parsed now and current position in that string

\*/

typedef struct {

unsigned int pos; /\* offset in the JSON string \*/

unsigned int toknext; /\* next token to allocate \*/

int toksuper; /\* superior token node, e.g parent object or array \*/

} jsmn\_parser;

/\*\*

\* Allocates a fresh unused token from the token pool.

\*/

static jsmntok\_t\* jsmn\_alloc\_token(jsmn\_parser\* parser, jsmntok\_t\* tokens, size\_t num\_tokens) {

jsmntok\_t\* tok;

if (parser->toknext >= num\_tokens) {

return NULL;

}

tok = &tokens[parser->toknext++];

tok->start = tok->end = -1;

tok->size = 0;

#ifdef JSMN\_PARENT\_LINKS

tok->parent = -1;

#endif

return tok;

}

/\*\*

\* Fills token type and boundaries.

\*/

static void jsmn\_fill\_token(jsmntok\_t\* token, jsmntype\_t type,

int start, int end) {

token->type = type;

token->start = start;

token->end = end;

token->size = 0;

}

/\*\*

\* Fills next available token with JSON primitive.

\*/

static int jsmn\_parse\_primitive(jsmn\_parser\* parser, const char\* js,

size\_t len, jsmntok\_t\* tokens, size\_t num\_tokens) {

jsmntok\_t\* token;

int start;

start = parser->pos;

for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {

switch (js[parser->pos]) {

#ifndef JSMN\_STRICT

/\* In strict mode primitive must be followed by "," or "}" or "]" \*/

case ':':

#endif

case '\t': case '\r': case '\n': case ' ':

case ',': case ']': case '}':

goto found;

}

if (js[parser->pos] < 32 || js[parser->pos] >= 127) {

parser->pos = start;

return JSMN\_ERROR\_INVAL;

}

}

#ifdef JSMN\_STRICT

/\* In strict mode primitive must be followed by a comma/object/array \*/

parser->pos = start;

return JSMN\_ERROR\_PART;

#endif

found:

if (tokens == NULL) {

parser->pos--;

return 0;

}

token = jsmn\_alloc\_token(parser, tokens, num\_tokens);

if (token == NULL) {

parser->pos = start;

return JSMN\_ERROR\_NOMEM;

}

jsmn\_fill\_token(token, JSMN\_PRIMITIVE, start, parser->pos);

#ifdef JSMN\_PARENT\_LINKS

token->parent = parser->toksuper;

#endif

parser->pos--;

return 0;

}

/\*\*

\* Fills next token with JSON string.

\*/

static int jsmn\_parse\_string(jsmn\_parser\* parser, const char\* js,

size\_t len, jsmntok\_t\* tokens, size\_t num\_tokens) {

jsmntok\_t\* token;

int start = parser->pos;

parser->pos++;

/\* Skip starting quote \*/

for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {

char c = js[parser->pos];

/\* Quote: end of string \*/

if (c == '\"') {

if (tokens == NULL) {

return 0;

}

token = jsmn\_alloc\_token(parser, tokens, num\_tokens);

if (token == NULL) {

parser->pos = start;

return JSMN\_ERROR\_NOMEM;

}

jsmn\_fill\_token(token, JSMN\_STRING, start + 1, parser->pos);

#ifdef JSMN\_PARENT\_LINKS

token->parent = parser->toksuper;

#endif

return 0;

}

/\* Backslash: Quoted symbol expected \*/

if (c == '\\' && parser->pos + 1 < len) {

int i;

parser->pos++;

switch (js[parser->pos]) {

/\* Allowed escaped symbols \*/

case '\"': case '/': case '\\': case 'b':

case 'f': case 'r': case 'n': case 't':

break;

/\* Allows escaped symbol \uXXXX \*/

case 'u':

parser->pos++;

for (i = 0; i < 4 && parser->pos < len && js[parser->pos] != '\0'; i++) {

/\* If it isn't a hex character we have an error \*/

if (!((js[parser->pos] >= 48 && js[parser->pos] <= 57) ||

(js[parser->pos] >= 65 && js[parser->pos] <= 70) ||

(js[parser->pos] >= 97 && js[parser->pos] <= 102))) {

parser->pos = start;

return JSMN\_ERROR\_INVAL;

}

parser->pos++;

}

parser->pos--;

break;

/\* Unexpected symbol \*/

default:

parser->pos = start;

return JSMN\_ERROR\_INVAL;

}

}

}

parser->pos = start;

return JSMN\_ERROR\_PART;

}

/\*\*

\* Parse JSON string and fill tokens.

\*/

static int jsmn\_parse(jsmn\_parser\* parser, const char\* js, size\_t len, jsmntok\_t\* tokens, unsigned int num\_tokens) {

int r;

int i;

jsmntok\_t\* token;

int count = parser->toknext;

for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {

char c;

jsmntype\_t type;

c = js[parser->pos];

switch (c) {

case '{': case '[':

count++;

if (tokens == NULL) {

break;

}

token = jsmn\_alloc\_token(parser, tokens, num\_tokens);

if (token == NULL)

return JSMN\_ERROR\_NOMEM;

if (parser->toksuper != -1) {

tokens[parser->toksuper].size++;

#ifdef JSMN\_PARENT\_LINKS

token->parent = parser->toksuper;

#endif

}

token->type = (c == '{' ? JSMN\_OBJECT : JSMN\_ARRAY);

token->start = parser->pos;

parser->toksuper = parser->toknext - 1;

break;

case '}': case ']':

if (tokens == NULL)

break;

type = (c == '}' ? JSMN\_OBJECT : JSMN\_ARRAY);

#ifdef JSMN\_PARENT\_LINKS

if (parser->toknext < 1) {

return JSMN\_ERROR\_INVAL;

}

token = &tokens[parser->toknext - 1];

for (;;) {

if (token->start != -1 && token->end == -1) {

if (token->type != type) {

return JSMN\_ERROR\_INVAL;

}

token->end = parser->pos + 1;

parser->toksuper = token->parent;

break;

}

if (token->parent == -1) {

if (token->type != type || parser->toksuper == -1) {

return JSMN\_ERROR\_INVAL;

}

break;

}

token = &tokens[token->parent];

}

#else

for (i = parser->toknext - 1; i >= 0; i--) {

token = &tokens[i];

if (token->start != -1 && token->end == -1) {

if (token->type != type) {

return JSMN\_ERROR\_INVAL;

}

parser->toksuper = -1;

token->end = parser->pos + 1;

break;

}

}

/\* Error if unmatched closing bracket \*/

if (i == -1) return JSMN\_ERROR\_INVAL;

for (; i >= 0; i--) {

token = &tokens[i];

if (token->start != -1 && token->end == -1) {

parser->toksuper = i;

break;

}

}

#endif

break;

case '\"':

r = jsmn\_parse\_string(parser, js, len, tokens, num\_tokens);

if (r < 0) return r;

count++;

if (parser->toksuper != -1 && tokens != NULL)

tokens[parser->toksuper].size++;

break;

case '\t': case '\r': case '\n': case ' ':

break;

case ':':

parser->toksuper = parser->toknext - 1;

break;

case ',':

if (tokens != NULL && parser->toksuper != -1 &&

tokens[parser->toksuper].type != JSMN\_ARRAY &&

tokens[parser->toksuper].type != JSMN\_OBJECT) {

#ifdef JSMN\_PARENT\_LINKS

parser->toksuper = tokens[parser->toksuper].parent;

#else

for (i = parser->toknext - 1; i >= 0; i--) {

if (tokens[i].type == JSMN\_ARRAY || tokens[i].type == JSMN\_OBJECT) {

if (tokens[i].start != -1 && tokens[i].end == -1) {

parser->toksuper = i;

break;

}

}

}

#endif

}

break;

#ifdef JSMN\_STRICT

/\* In strict mode primitives are: numbers and booleans \*/

case '-': case '0': case '1': case '2': case '3': case '4':

case '5': case '6': case '7': case '8': case '9':

case 't': case 'f': case 'n':

/\* And they must not be keys of the object \*/

if (tokens != NULL && parser->toksuper != -1) {

jsmntok\_t\* t = &tokens[parser->toksuper];

if (t->type == JSMN\_OBJECT ||

(t->type == JSMN\_STRING && t->size != 0)) {

return JSMN\_ERROR\_INVAL;

}

}

#else

/\* In non-strict mode every unquoted value is a primitive \*/

default:

#endif

r = jsmn\_parse\_primitive(parser, js, len, tokens, num\_tokens);

if (r < 0) return r;

count++;

if (parser->toksuper != -1 && tokens != NULL)

tokens[parser->toksuper].size++;

break;

#ifdef JSMN\_STRICT

/\* Unexpected char in strict mode \*/

default:

return JSMN\_ERROR\_INVAL;

#endif

}

}

if (tokens != NULL) {

for (i = parser->toknext - 1; i >= 0; i--) {

/\* Unmatched opened object or array \*/

if (tokens[i].start != -1 && tokens[i].end == -1) {

return JSMN\_ERROR\_PART;

}

}

}

return count;

}

/\*\*

\* Creates a new parser based over a given buffer with an array of tokens

\* available.

\*/

static void jsmn\_init(jsmn\_parser\* parser) {

parser->pos = 0;

parser->toknext = 0;

parser->toksuper = -1;

}

static int jsoneq(const char\* json, jsmntok\_t\* tok, const char\* s)

{

if (tok->type == JSMN\_STRING

&& (int)strlen(s) == tok->end - tok->start

&& strncmp(json + tok->start, s, tok->end - tok->start) == 0) {

return 0;

}

return -1;

}

static size\_t load\_file(char\* filename)

{

FILE\* f = fopen(filename, "rb");

if (f == NULL) {

fprintf(stderr, "Unable to open file %s\n", filename);

exit(1);

}

fseek(f, 0, SEEK\_END);

size\_t fsize = ftell(f);

fseek(f, 0, SEEK\_SET); //same as rewind(f);

if (fsize >= BUFF\_SIZE) {

fprintf(stderr, "Configuration file too large\n");

abort();

}

fread(config, fsize, 1, f);

fclose(f);

config[fsize] = 0;

return fsize;

}

char\* strndup(char\* str, int chars)

{

char\* buffer;

int n;

buffer = (char\*)malloc(chars + 1);

if (buffer)

{

for (n = 0; ((n < chars) && (str[n] != 0)); n++) buffer[n] = str[n];

buffer[n] = 0;

}

return buffer;

}

int parse\_config(char\* path, struct configuration\* conf)

{

int i;

int r;

jsmn\_parser p;

jsmntok\_t t[128]; /\* We expect no more than 128 tokens \*/

load\_file(path);

jsmn\_init(&p);

r = jsmn\_parse(&p, config, strlen(config), t, sizeof(t) / sizeof(t[0]));

if (r < 0) {

printf("Failed to parse JSON: %d\n", r);

return 0;

}

/\* Assume the top-level element is an object \*/

if (r < 1 || t[0].type != JSMN\_OBJECT) {

printf("Object expected\n");

return 0;

}

/\* Loop over all keys of the root object \*/

for (i = 1; i < r; i++) {

if (jsoneq(config, &t[i], "host") == 0) {

/\* We may use strndup() to fetch string value \*/

conf->host = strndup(config + t[i + 1].start, t[i + 1].end - t[i + 1].start);

i++;

}

else if (jsoneq(config, &t[i], "username") == 0) {

conf->db\_username = strndup(config + t[i + 1].start, t[i + 1].end -  
t[i + 1].start);

i++;

}

else if (jsoneq(config, &t[i], "password") == 0) {

conf->db\_password = strndup(config + t[i + 1].start, t[i + 1].end -  
t[i + 1].start);

i++;

}

else if (jsoneq(config, &t[i], "port") == 0) {

conf->port = strtol(config + t[i + 1].start, NULL, 10);

i++;

}

else if (jsoneq(config, &t[i], "database") == 0) {

conf->database = strndup(config + t[i + 1].start, t[i + 1].end - t[i + 1].start);

i++;

}

else {

printf("Unexpected key: %.\*s\n", t[i].end - t[i].start, config + t[i].start);

}

}

return 1;

}

**inout.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <signal.h>

#include <stdbool.h>

#include <conio.h>

#include "defines.h"

bool yesOrNo(char\* domanda, char yes, char no, bool predef, bool insensitive)

{

// I caratteri 'yes' e 'no' devono essere minuscoli

yes = tolower(yes);

no = tolower(no);

// Decide quale delle due lettere mostrare come predefinite

char s, n;

if (predef) {

s = toupper(yes);

n = no;

}

else {

s = yes;

n = toupper(no);

}

// Richiesta della risposta

while (true) {

// Mostra la domanda

printf("%s [%c/%c]: ", domanda, s, n);

char c;

fgets(&c, 2, stdin);

// Controlla quale risposta è stata data

if (c == '\0') { // getInput() non può restituire '\n'!

return predef;

}

else if (c == yes) {

return true;

}

else if (c == no) {

return false;

}

else if (c == toupper(yes)) {

if (predef || insensitive) return true;

}

else if (c == toupper(yes)) {

if (!predef || insensitive) return false;

}

}

}

char multiChoice(char\* domanda, char choices[], int num)

{

char inputStr[3];

// Genera la stringa delle possibilità

char\* possib = (char\*)malloc(2 \* num \* sizeof(char));

int i, j = 0;

for (i = 0; i < num; i++) {

possib[j++] = choices[i];

possib[j++] = '/';

}

possib[j - 1] = '\0'; // Per eliminare l'ultima '/'

// Chiede la risposta

while (true) {

// Mostra la domanda

printf("%s [%s]: ", domanda, possib);

fgets(inputStr, 3, stdin);

char c = inputStr[0];

// Controlla se è un carattere valido

for (i = 0; i < num; i++) {

if (c == choices[i])

return c;

}

}

}

void insertPassword(char password[]) {

int ch;

int i = 0;

while ((ch = \_getch()) != EOF && ch != '\n' && ch != '\r' && i < sizeof(password) - 1) {

if (ch == '\b' && i > 0) {

printf("\b \b"); // Destructive backspace

fflush(stdout);

i--;

password[i] = '\0';

}

else if (ch != '\b') {

putchar('\*');

password[i++] = (char)ch;

}

}

password[i] = '\0';

printf("\n");

fflush(stdout);

}

**utils.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

void print\_stmt\_error(MYSQL\_STMT\* stmt, char\* message)

{

fprintf(stderr, "%s\n", message);

if (stmt != NULL) {

fprintf(stderr, "Error %u (%s): %s\n",

mysql\_stmt\_errno(stmt),

mysql\_stmt\_sqlstate(stmt),

mysql\_stmt\_error(stmt));

}

}

void print\_error(MYSQL\* conn, char\* message)

{

fprintf(stderr, "%s\n", message);

if (conn != NULL) {

#if MYSQL\_VERSION\_ID >= 40101

fprintf(stderr, "Error %u (%s): %s\n",

mysql\_errno(conn), mysql\_sqlstate(conn), mysql\_error(conn));

#else

fprintf(stderr, "Error %u: %s\n",

mysql\_errno(conn), mysql\_error(conn));

#endif

}

}

bool setup\_prepared\_stmt(MYSQL\_STMT\*\* stmt, char\* statement, MYSQL\* conn)

{

bool update\_length = true;

\*stmt = mysql\_stmt\_init(conn);

if (\*stmt == NULL)

{

print\_error(conn, (char\*)"Could not initialize statement handler");

return false;

}

if (mysql\_stmt\_prepare(\*stmt, statement, strlen(statement)) != 0) {

print\_stmt\_error(\*stmt, (char\*)"Could not prepare statement");

return false;

}

mysql\_stmt\_attr\_set(\*stmt, STMT\_ATTR\_UPDATE\_MAX\_LENGTH, &update\_length);

return true;

}

void finish\_with\_error(MYSQL\* conn, char\* message)

{

print\_error(conn, message);

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

void finish\_with\_stmt\_error(MYSQL\* conn, MYSQL\_STMT\* stmt, char\* message, bool close\_stmt)

{

print\_stmt\_error(stmt, message);

if (close\_stmt) mysql\_stmt\_close(stmt);

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

static void print\_dashes(MYSQL\_RES\* res\_set)

{

MYSQL\_FIELD\* field;

unsigned int i, j;

mysql\_field\_seek(res\_set, 0);

putchar('+');

for (i = 0; i < mysql\_num\_fields(res\_set); i++) {

field = mysql\_fetch\_field(res\_set);

for (j = 0; j < field->max\_length + 2; j++)

putchar('-');

putchar('+');

}

putchar('\n');

}

static void dump\_result\_set\_header(MYSQL\_RES\* res\_set)

{

MYSQL\_FIELD\* field;

unsigned long col\_len;

unsigned int i;

/\* determine column display widths -- requires result set to be \*/

/\* generated with mysql\_store\_result(), not mysql\_use\_result() \*/

mysql\_field\_seek(res\_set, 0); /\* it's like seek() function for file descriptors \*/

for (i = 0; i < mysql\_num\_fields(res\_set); i++) {

field = mysql\_fetch\_field(res\_set); /\* type of each field (column) \*/

col\_len = strlen(field->name); /\* length of the name of each field \*/

if (col\_len < field->max\_length)

col\_len = field->max\_length;

if (col\_len < 4 && !IS\_NOT\_NULL(field->flags))

col\_len = 4; /\* 4 = length of the word "NULL" \*/

field->max\_length = col\_len; /\* reset column info \*/

}

print\_dashes(res\_set);

putchar('|');

mysql\_field\_seek(res\_set, 0);

for (i = 0; i < mysql\_num\_fields(res\_set); i++) {

field = mysql\_fetch\_field(res\_set);

printf(" %-\*s |", (int)field->max\_length, field->name);

}

putchar('\n');

print\_dashes(res\_set);

}

void dump\_result\_set(MYSQL\* conn, MYSQL\_STMT\* stmt, char\* title)

{

int i;

int status;

int num\_fields; /\* number of columns in result \*/

MYSQL\_FIELD\* fields; /\* for result set metadata \*/

MYSQL\_BIND\* rs\_bind; /\* for output buffers \*/

MYSQL\_RES\* rs\_metadata; /\* to get metadata of result set \*/

MYSQL\_TIME\* date;

size\_t attr\_size;

/\* Prefetch the whole result set. This in conjunction with

\* STMT\_ATTR\_UPDATE\_MAX\_LENGTH set in `setup\_prepared\_stmt`

\* updates the result set metadata which are fetched in this

\* function, to allow to compute the actual max length of

\* the columns.

\*/

if (mysql\_stmt\_store\_result(stmt)) {

fprintf(stderr, " mysql\_stmt\_execute(), 1 failed\n");

fprintf(stderr, " %s\n", mysql\_stmt\_error(stmt));

exit(0);

}

/\* the column count is > 0 if there is a result set \*/

/\* 0 if the result is only the final status packet \*/

num\_fields = mysql\_stmt\_field\_count(stmt);

if (num\_fields > 0) {

/\* there is a result set to fetch \*/

printf("%s\n", title);

if ((rs\_metadata = mysql\_stmt\_result\_metadata(stmt)) == NULL) {

finish\_with\_stmt\_error(conn,stmt,"Unable to retrieve result metadata\n", true);

}

dump\_result\_set\_header(rs\_metadata); /\* function that allows to print pretty tables on stdout \*/

fields = mysql\_fetch\_fields(rs\_metadata);

rs\_bind = (MYSQL\_BIND\*)malloc(sizeof(MYSQL\_BIND) \* num\_fields);

if (!rs\_bind) {

finish\_with\_stmt\_error(conn, stmt, “Cannot allocate output buffers\n", true);

}

memset(rs\_bind, 0, sizeof(MYSQL\_BIND) \* num\_fields);

/\* set up and bind result set output buffers \*/

for (i = 0; i < num\_fields; ++i) {

// Properly size the parameter buffer

switch (fields[i].type) {

case MYSQL\_TYPE\_DATE:

case MYSQL\_TYPE\_TIMESTAMP:

case MYSQL\_TYPE\_DATETIME:

case MYSQL\_TYPE\_TIME:

attr\_size = sizeof(MYSQL\_TIME);

break;

case MYSQL\_TYPE\_FLOAT:

attr\_size = sizeof(float);

break;

case MYSQL\_TYPE\_DOUBLE:

attr\_size = sizeof(double);

break;

case MYSQL\_TYPE\_TINY:

attr\_size = sizeof(signed char);

break;

case MYSQL\_TYPE\_SHORT:

case MYSQL\_TYPE\_YEAR:

attr\_size = sizeof(short int);

break;

case MYSQL\_TYPE\_LONG:

case MYSQL\_TYPE\_INT24:

attr\_size = sizeof(int);

break;

case MYSQL\_TYPE\_LONGLONG:

attr\_size = sizeof(int);

break;

default:

attr\_size = fields[i].max\_length;

break;

}

// Setup the binding for the current parameter

rs\_bind[i].buffer\_type = fields[i].type;

rs\_bind[i].buffer = malloc(attr\_size + 1);

rs\_bind[i].buffer\_length = attr\_size + 1;

if (rs\_bind[i].buffer == NULL) {

finish\_with\_stmt\_error(conn, stmt, "Cannot allocate buffers\n", true);

}

}

if (mysql\_stmt\_bind\_result(stmt, rs\_bind)) {

finish\_with\_stmt\_error(conn, stmt, "Unable to bind parameters\n", true);

}

/\* fetch and display result set rows \*/

while (true) {

status = mysql\_stmt\_fetch(stmt);

if (status == 1 || status == MYSQL\_NO\_DATA)

break;

putchar('|');

for (i = 0; i < num\_fields; i++) {

if (rs\_bind[i].is\_null\_value) {

printf(" %-\*s |", (int)fields[i].max\_length, "NULL");

continue;

}

switch (rs\_bind[i].buffer\_type) {

case MYSQL\_TYPE\_VAR\_STRING:

case MYSQL\_TYPE\_DATETIME:

printf(" %-\*s |", (int)fields[i].max\_length, (char\*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_DATE:

case MYSQL\_TYPE\_TIMESTAMP:

date = (MYSQL\_TIME\*)rs\_bind[i].buffer;

printf("%d-%02d-%02d|",date->year, date->month, date->day);

break;

case MYSQL\_TYPE\_TIME:

date = (MYSQL\_TIME\*)rs\_bind[i].buffer;

printf(" %02d : %02d |", date->hour, date->minute);

break;

case MYSQL\_TYPE\_STRING:

printf(" %-\*s |", (int)fields[i].max\_length, (char\*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_FLOAT:

case MYSQL\_TYPE\_DOUBLE:

printf(" %.02f |", \*(float\*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_LONG:

case MYSQL\_TYPE\_SHORT:

case MYSQL\_TYPE\_TINY:

printf(" %-\*d |", (int)fields[i].max\_length, \*(int\*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_NEWDECIMAL:

printf(" %-\*.02lf |", (int)fields[i].max\_length, \*(float\*)rs\_bind[i].buffer);

break;

default:

printf("Error: Unhandled type (%d)\n", rs\_bind[i].buffer\_type);

abort();

}

}

putchar('\n');

print\_dashes(rs\_metadata);

}

mysql\_free\_result(rs\_metadata); /\* free metadata \*/

/\* free output buffers \*/

for (i = 0; i < num\_fields; i++) {

free(rs\_bind[i].buffer);

}

free(rs\_bind);

}

}

void bind\_par(MYSQL\_BIND param[], struct param\_type\* paramStructPtr) {

struct param\_type paramStruct = \*paramStructPtr;

// Prepare parameters

memset(param, 0, sizeof(param));

int count; // Number of entries of param[] that have been initalized

for (int i = 0; i < paramStruct.numVarchar; i++) {

param[i].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[i].buffer = paramStruct.varchar[i];

param[i].buffer\_length = strlen(paramStruct.varchar[i]);

}

count = paramStruct.numVarchar;

for (int i = 0; i < paramStruct.numDate; i++) {

param[i + count].buffer\_type = MYSQL\_TYPE\_DATE;

param[i + count].buffer = (char \*)paramStruct.date[i]; // Pointer to a MYSQL\_TIME data structure

param[i + count].buffer\_length = sizeof(MYSQL\_TIME);

}

count = count + paramStruct.numDate;

for (int i = 0; i < paramStruct.numTime; i++) {

param[i + count].buffer\_type = MYSQL\_TYPE\_TIME;

param[i + count].buffer = (char \*)paramStruct.time[i];

param[i + count].buffer\_length = sizeof(MYSQL\_TIME);

}

count = count + paramStruct.numTime;

for (int i = 0; i < paramStruct.numInt; i++) {

param[i + count].buffer\_type = MYSQL\_TYPE\_LONG;

param[i + count].buffer = paramStruct.integer[i];

param[i + count].buffer\_length = sizeof(int);

}

count = count + paramStruct.numInt;

for (int i = 0; i < paramStruct.numFloat; i++) {

param[i + count].buffer\_type = MYSQL\_TYPE\_FLOAT;

param[i + count].buffer = paramStruct.f;

param[i + count].buffer\_length = sizeof(float);

}

count = count + paramStruct.numFloat;

for (int i = 0; i < paramStruct.numBool; i++) {

param[i + count].buffer\_type = MYSQL\_TYPE\_SHORT;

param[i + count].buffer = paramStruct.b;

param[i + count].buffer\_length = sizeof(short int);

}

}

char\* fixString(char string[]) {

int count = 0;

char c;

while (true) {

c = string[count];

if (!isdigit(c)) {

break;

}

count++;

}

char\* newString = (char\*)malloc(sizeof(char) \* (count + 1));

if (newString == NULL) {

printf("Cannot allocate memory in the heap\n");

fflush(stdout);

exit(EXIT\_FAILURE);

}

for (int i = 0; i < count; i++) {

newString[i] = string[i];

}

newString[count] = '\0';

return newString;

}

**administrator.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

static void add\_sick\_leave(MYSQL\* conn);

static void add\_book(MYSQL\* conn) {

MYSQL\_STMT\* aggiungi\_copia\_procedure;

MYSQL\_BIND param[7];

struct param\_type paramStruct;

char isbn[16];

char titolo[48];

char autore[48];

char biblioteca[24];

char scaffaleStr[16];

char ripianoStr[16];

// Get the required information

printf("\nISBN: ");

fflush(stdout);

fgets(isbn, 16, stdin);

isbn[strlen(isbn) - 1] = '\0';

printf("Title: ");

fflush(stdout);

fgets(titolo, 48, stdin);

titolo[strlen(titolo) - 1] = '\0';

printf("Author: ");

fflush(stdout);

fgets(autore, 48, stdin);

autore[strlen(autore) - 1] = '\0';

printf("Library phone number: ");

fflush(stdout);

fgets(biblioteca, 24, stdin);

biblioteca[strlen(biblioteca) - 1] = '\0';

printf("Shelving unit number: ");

fflush(stdout);

fgets(scaffaleStr, 16, stdin);

printf("Shelf number: ");

fflush(stdout);

fgets(ripianoStr, 16, stdin);

int scaffale = atoi(scaffaleStr);

int ripiano = atoi(ripianoStr);

int id = 0;

if (!setup\_prepared\_stmt(&aggiungi\_copia\_procedure,"call aggiungi\_copia(?, ?, ?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_copia\_procedure, "Unable to initialize book insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = isbn;

paramStruct.varchar[1] = titolo;

paramStruct.varchar[2] = autore;

paramStruct.varchar[3] = biblioteca;

paramStruct.integer[0] = &scaffale;

paramStruct.integer[1] = &ripiano;

paramStruct.integer[2] = &id;

paramStruct.numVarchar = 4;

paramStruct.numInt = 3;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_copia\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, aggiungi\_copia\_procedure, "Could not bind parameters for book insertion\n", true);

}

if (mysql\_stmt\_execute(aggiungi\_copia\_procedure) != 0) {

print\_stmt\_error(aggiungi\_copia\_procedure, "An error occurred while adding the book\n");

goto out;

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.integer[0] = &id;

paramStruct.numInt = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_result(aggiungi\_copia\_procedure, param)) {

finish\_with\_stmt\_error(conn, aggiungi\_copia\_procedure, "Could not retrieve output parameter\n", true);

}

if (mysql\_stmt\_fetch(aggiungi\_copia\_procedure)) {

finish\_with\_stmt\_error(conn, aggiungi\_copia\_procedure, "Could not buffer results\n", true);

}

printf("Book correctly added with ID %d\n", id);

fflush(stdout);

out:

mysql\_stmt\_close(aggiungi\_copia\_procedure);

}

static void add\_work\_shift(MYSQL\* conn) {

MYSQL\_STMT\* aggiungi\_turno\_procedure;

MYSQL\_BIND param[4];

struct param\_type paramStruct;

char bibliotecario[24];

char giornoStr[16];

char meseStr[16];

char annoStr[16];

char inizioOreStr[16];

char inizioMinStr[16];

char fineOreStr[16];

char fineMinStr[16];

printf("\nLibrarian tax code: ");

fflush(stdout);

fgets(bibliotecario, 24, stdin);

bibliotecario[strlen(bibliotecario) - 1] = '\0';

printf("Day [1-31]: ");

fflush(stdout);

fgets(giornoStr, 16, stdin);

printf("Month [1-12]: ");

fflush(stdout);

fgets(meseStr, 16, stdin);

printf("Year: ");

fflush(stdout);

fgets(annoStr, 16, stdin);

printf("Hour of the start of work shift [0-23]: ");

fflush(stdout);

fgets(inizioOreStr, 16, stdin);

printf("Minute of the start of work shift [0-59]: ");

fflush(stdout);

fgets(inizioMinStr, 16, stdin);

printf("Hour of the end of work shift [0-23]: ");

fflush(stdout);

fgets(fineOreStr, 16, stdin);

printf("Minute of the end of work shift [0-59]: ");

fflush(stdout);

fgets(fineMinStr, 16, stdin);

int giorno = atoi(giornoStr);

int mese = atoi(meseStr);

int anno = atoi(annoStr);

int inizioOre = atoi(inizioOreStr);

int inizioMin = atoi(inizioMinStr);

int fineOre = atoi(fineOreStr);

int fineMin = atoi(fineMinStr);

struct MYSQL\_TIME date;

struct MYSQL\_TIME time1;

struct MYSQL\_TIME time2;

memset(&date, 0, sizeof(date));

memset(&time1, 0, sizeof(time1));

memset(&time2, 0, sizeof(time2));

date.day = giorno;

date.month = mese;

date.year = anno;

time1.hour = inizioOre;

time1.minute = inizioMin;

time2.hour = fineOre;

time2.minute = fineMin;

if (!setup\_prepared\_stmt(&aggiungi\_turno\_procedure, "call aggiungi\_turno(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_turno\_procedure, "Unable to initialize work shift insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = bibliotecario;

paramStruct.date[0] = &date;

paramStruct.time[0] = &time1;

paramStruct.time[1] = &time2;

paramStruct.numVarchar = 1;

paramStruct.numDate = 1;

paramStruct.numTime = 2;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_turno\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, aggiungi\_turno\_procedure, "Could not bind parameters for work shift insertion\n", true);

}

if (mysql\_stmt\_execute(aggiungi\_turno\_procedure) != 0) {

print\_stmt\_error(aggiungi\_turno\_procedure, "An error occurred while adding the work shift\n");

}

else {

printf("Work shift correctly added\n");

fflush(stdout);

}

mysql\_stmt\_close(aggiungi\_turno\_procedure);

}

static void add\_librarian(MYSQL\* conn, char username[]) {

MYSQL\_STMT\* aggiungi\_bibliotecario\_procedure;

MYSQL\_BIND param[8];

struct param\_type paramStruct;

char cf[24];

char nome[48];

char cognome[48];

char giornoNascitaStr[16];

char meseNascitaStr[16];

char annoNascitaStr[16];

char luogoNascita[48];

char titoloStudio[48];

char biblioteca[24];

printf("\nTax code: ");

fflush(stdout);

fgets(cf, 24, stdin);

cf[strlen(cf) - 1] = '\0';

printf("Name: ");

fflush(stdout);

fgets(nome, 48, stdin);

nome[strlen(nome) - 1] = '\0';

printf("Surname: ");

fflush(stdout);

fgets(cognome, 48, stdin);

cognome[strlen(cognome) - 1] = '\0';

printf("Day of birth [1-31]: ");

fflush(stdout);

fgets(giornoNascitaStr, 16, stdin);

printf("Month of birth [1-12]: ");

fflush(stdout);

fgets(meseNascitaStr, 16, stdin);

printf("Year of birth: ");

fflush(stdout);

fgets(annoNascitaStr, 16, stdin);

printf("City of birth: ");

fflush(stdout);

fgets(luogoNascita, 48, stdin);

luogoNascita[strlen(luogoNascita) - 1] = '\0';

printf("Educational qualification: ");

fflush(stdout);

fgets(titoloStudio, 48, stdin);

titoloStudio[strlen(titoloStudio) - 1] = '\0';

printf("Library phone number: ");

fflush(stdout);

fgets(biblioteca, 24, stdin);

biblioteca[strlen(biblioteca) - 1] = '\0';

int giornoNascita = atoi(giornoNascitaStr);

int meseNascita = atoi(meseNascitaStr);

int annoNascita = atoi(annoNascitaStr);

struct MYSQL\_TIME date;

memset(&date, 0, sizeof(date));

date.day = giornoNascita;

date.month = meseNascita;

date.year = annoNascita;

if (!setup\_prepared\_stmt(&aggiungi\_bibliotecario\_procedure, "call aggiungi\_bibliotecario  
(?, ?, ?, ?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_bibliotecario\_procedure, "Unable to initialize librarian insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = cf;

paramStruct.varchar[1] = nome;

paramStruct.varchar[2] = cognome;

paramStruct.varchar[3] = luogoNascita;

paramStruct.varchar[4] = titoloStudio;

paramStruct.varchar[5] = biblioteca;

paramStruct.varchar[6] = username;

paramStruct.date[0] = &date;

paramStruct.numVarchar = 7;

paramStruct.numDate = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_bibliotecario\_procedure, param) != 0) {

print\_stmt\_error(aggiungi\_bibliotecario\_procedure, "Could not bind parameters for librarian insertion\n");

goto again;

}

if (mysql\_stmt\_execute(aggiungi\_bibliotecario\_procedure) != 0) {

print\_stmt\_error(aggiungi\_bibliotecario\_procedure, "An error occurred while adding the librarian\n");

goto again;

}

else {

goto end;

}

again:

mysql\_stmt\_close(aggiungi\_bibliotecario\_procedure);

printf("\nLet's try again!\n");

fflush(stdout);

add\_librarian(conn, username);

return;

end:

printf("Librarian correctly added\n");

fflush(stdout);

mysql\_stmt\_close(aggiungi\_bibliotecario\_procedure);

}

static void create\_user(MYSQL\* conn) {

MYSQL\_STMT\* crea\_user\_procedure;

MYSQL\_BIND param[3];

struct param\_type paramStruct;

char options[2] = { '1', '2' };

char r;

char username[48];

char password[48];

char ruolo[48];

printf("\nUsername: ");

fflush(stdout);

fgets(username, 48, stdin);

username[strlen(username) - 1] = '\0';

printf("Password: ");

fflush(stdout);

insertPassword(password);

printf("Assign a possible role:\n");

printf("\t1) Amministratore\n");

printf("\t2) Bibliotecario\n");

fflush(stdout);

r = multiChoice("Select role", options, 2);

//Convert role into enum value

switch (r) {

case '1':

strcpy(ruolo, "amministratore");

break;

case '2':

strcpy(ruolo, "bibliotecario");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if (!setup\_prepared\_stmt(&crea\_user\_procedure, "call crea\_user(?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, crea\_user\_procedure, "Unable to initialize user insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = username;

paramStruct.varchar[1] = password;

paramStruct.varchar[2] = ruolo;

paramStruct.numVarchar = 3;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(crea\_user\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, crea\_user\_procedure, "Could not bind parameters for user insertion\n", true);

}

if (mysql\_stmt\_execute(crea\_user\_procedure) != 0) {

print\_stmt\_error(crea\_user\_procedure, "An error occurred while adding the user\n");

mysql\_stmt\_close(crea\_user\_procedure);

printf("Let's try again!\n");

fflush(stdout);

create\_user(conn);

}

else {

printf("User correctly added\n");

fflush(stdout);

mysql\_stmt\_close(crea\_user\_procedure);

if (r == '2') {

add\_librarian(conn, username);

}

}

}

static void write\_off\_books(MYSQL\* conn) {

MYSQL\_STMT\* dismetti\_copie\_procedure;

if (!setup\_prepared\_stmt(&dismetti\_copie\_procedure, "call dismetti\_copie()", conn)) {

finish\_with\_stmt\_error(conn, dismetti\_copie\_procedure, "Unable to initialize books dissolution statement\n", false);

}

if (mysql\_stmt\_execute(dismetti\_copie\_procedure) != 0) {

print\_stmt\_error(dismetti\_copie\_procedure, "An error occurred while writing off the books\n");

}

else {

printf("Books correctly wrote off\n");

fflush(stdout);

}

mysql\_stmt\_close(dismetti\_copie\_procedure);

}

static void report\_uncovered\_libraries(MYSQL\* conn) {

MYSQL\_STMT\* report\_biblioteche\_scoperte\_procedure;

MYSQL\_BIND param[2];

struct param\_type paramStruct;

char strGiorno1[16];

char strMese1[16];

char strAnno1[16];

char strGiorno2[16];

char strMese2[16];

char strAnno2[16];

printf("\nFirst day [1-31]: ");

fflush(stdout);

fgets(strGiorno1, 16, stdin);

printf("First month [1-12]: ");

fflush(stdout);

fgets(strMese1, 16, stdin);

printf("First year: ");

fflush(stdout);

fgets(strAnno1, 16, stdin);

printf("LastDay [1-31]: ");

fflush(stdout);

fgets(strGiorno2, 16, stdin);

printf("LastMonth [1-12]: ");

fflush(stdout);

fgets(strMese2, 16, stdin);

printf("LastYear: ");

fflush(stdout);

fgets(strAnno2, 16, stdin);

int giorno1 = atoi(strGiorno1);

int mese1 = atoi(strMese1);

int anno1 = atoi(strAnno1);

int giorno2 = atoi(strGiorno2);

int mese2 = atoi(strMese2);

int anno2 = atoi(strAnno2);

struct MYSQL\_TIME date1;

struct MYSQL\_TIME date2;

memset(&date1, 0, sizeof(date1));

memset(&date2, 0, sizeof(date2));

date1.day = giorno1;

date1.month = mese1;

date1.year = anno1;

date2.day = giorno2;

date2.month = mese2;

date2.year = anno2;

if(!setup\_prepared\_stmt(&report\_biblioteche\_scoperte\_procedure,"call report\_biblioteche\_scoperte(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, report\_biblioteche\_scoperte\_procedure, "Unable to initialize uncovered libraries report statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.date[0] = &date1;

paramStruct.date[1] = &date2;

paramStruct.numDate = 2;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(report\_biblioteche\_scoperte\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, report\_biblioteche\_scoperte\_procedure, "Could not bind parameters for uncovered libraries report\n", true);

}

if (mysql\_stmt\_execute(report\_biblioteche\_scoperte\_procedure) != 0) {

print\_stmt\_error(report\_biblioteche\_scoperte\_procedure, "An error occurred while reporting uncovered libraries\n");

goto out;

}

dump\_result\_set(conn, report\_biblioteche\_scoperte\_procedure, "");

if (mysql\_stmt\_next\_result(report\_biblioteche\_scoperte\_procedure) > 0) {

finish\_with\_stmt\_error(conn, report\_biblioteche\_scoperte\_procedure, "Unexpected contidion\n", true);

}

out:

mysql\_stmt\_close(report\_biblioteche\_scoperte\_procedure);

}

static void add\_sick\_leave\_continue(MYSQL\* conn, char malato[], struct MYSQL\_TIME\* date) {

MYSQL\_STMT\* aggiungi\_malattia\_procedure;

MYSQL\_BIND param[4];

struct param\_type paramStruct;

char sostituto[24];

char motivo[48];

printf("\nSubstitute librarian tax code: ");

fflush(stdout);

fgets(sostituto, 24, stdin);

sostituto[strlen(sostituto) - 1] = '\0';

printf("Sick leave reason: ");

fflush(stdout);

fgets(motivo, 48, stdin);

motivo[strlen(motivo) - 1] = '\0';

if (!setup\_prepared\_stmt(&aggiungi\_malattia\_procedure, "call aggiungi\_malattia(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_malattia\_procedure, "Unable to initialize sick leave insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = malato;

paramStruct.varchar[1] = motivo;

paramStruct.varchar[2] = sostituto;

paramStruct.date[0] = date;

paramStruct.numVarchar = 3;

paramStruct.numDate = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_malattia\_procedure, param) != 0) {

print\_stmt\_error(aggiungi\_malattia\_procedure, "Could not bind parameters for sick leave insertion\n");

goto again;

}

if (mysql\_stmt\_execute(aggiungi\_malattia\_procedure) != 0) {

print\_stmt\_error(aggiungi\_malattia\_procedure, "An error occurred while adding the sick leave\n");

goto again;

}

else {

goto end;

}

again:

mysql\_stmt\_close(aggiungi\_malattia\_procedure);

printf("\nLet's try again!\n");

fflush(stdout);

add\_sick\_leave(conn);

return;

end:

printf("Sick leave correctly added\n");

fflush(stdout);

mysql\_stmt\_close(aggiungi\_malattia\_procedure);

}

static void add\_sick\_leave(MYSQL\* conn) {

MYSQL\_STMT\* trova\_sostituti\_procedure;

MYSQL\_BIND param[2];

struct param\_type paramStruct;

char malato[24];

char giornoStr[16];

char meseStr[16];

char annoStr[16];

printf("\nSick librarian tax code: ");

fflush(stdout);

fgets(malato, 24, stdin);

malato[strlen(malato) - 1] = '\0';

printf("Sick leave day [1-31]: ");

fflush(stdout);

fgets(giornoStr, 16, stdin);

printf("Sick leave month [1-12]: ");

fflush(stdout);

fgets(meseStr, 16, stdin);

printf("Sick leave year: ");

fflush(stdout);

fgets(annoStr, 16, stdin);

int giorno = atoi(giornoStr);

int mese = atoi(meseStr);

int anno = atoi(annoStr);

struct MYSQL\_TIME date;

memset(&date, 0, sizeof(date));

date.day = giorno;

date.month = mese;

date.year = anno;

if (!setup\_prepared\_stmt(&trova\_sostituti\_procedure, "call trova\_sostituti(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, trova\_sostituti\_procedure, "Unable to initialize substitute librarians search statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = malato;

paramStruct.date[0] = &date;

paramStruct.numVarchar = 1;

paramStruct.numDate = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(trova\_sostituti\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, trova\_sostituti\_procedure, "Could not bind parameters for substitute librarians search\n", true);

}

if (mysql\_stmt\_execute(trova\_sostituti\_procedure) != 0) {

finish\_with\_stmt\_error(conn, trova\_sostituti\_procedure, "An error occurred while looking for substitute librarians\n", true);

}

dump\_result\_set(conn, trova\_sostituti\_procedure, "");

if (mysql\_stmt\_next\_result(trova\_sostituti\_procedure) > 0) {

finish\_with\_stmt\_error(conn, trova\_sostituti\_procedure, "Unexpected contidion\n", true);

}

mysql\_stmt\_close(trova\_sostituti\_procedure);

add\_sick\_leave\_continue(conn, malato, &date);

}

void run\_as\_administrator(MYSQL\* conn) {

char options[7] = { '1', '2', '3', '4', '5', '6', '7'};

char op;

printf("Switching to administrator role...\n");

fflush(stdout);

if (!parse\_config("users/amministratore.json", &conf)) {

fprintf(stderr, "Unable to load administrator configuration\n");

exit(EXIT\_FAILURE);

}

if (mysql\_change\_user(conn, conf.db\_username, conf.db\_password, conf.database)) {

fprintf(stderr, "mysql\_change\_user() failed\n");

exit(EXIT\_FAILURE);

}

while (true) {

printf("\033[2J\033[H"); // Clean the shell

printf("\*\*\* What should I do for you? \*\*\*\n\n");

printf("1) Add a book into a library\n");

printf("2) Manage librarians' work shifts\n");

printf("3) Create a user\n");

printf("4) Write off some books\n");

printf("5) Show the uncovered libraries\n");

printf("6) Set a librarian on sick leave\n");

printf("7) Quit\n");

fflush(stdout);

op = multiChoice("Select an option", options, 7);

switch (op) {

case '1':

add\_book(conn);

break;

case '2':

add\_work\_shift(conn);

break;

case '3':

create\_user(conn);

break;

case '4':

write\_off\_books(conn);

break;

case '5':

report\_uncovered\_libraries(conn);

break;

case '6':

add\_sick\_leave(conn);

break;

case '7':

return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

getchar();

}

}

**librarian.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

static void start\_loan(MYSQL\* conn, char bibliotecaImp[16]);

static char\* find\_library(MYSQL\* conn) {

MYSQL\_STMT\* ottieni\_biblioteca\_impiego\_procedure;

MYSQL\_BIND param[2];

struct param\_type paramStruct;

char biblioteca[16] = "zzzzzzzzzzzzzzz";

if(!setup\_prepared\_stmt(&ottieni\_biblioteca\_impiego\_procedure,"call ottieni\_biblioteca\_impiego(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, ottieni\_biblioteca\_impiego\_procedure, "Unable to initialize library search statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = conf.username;

paramStruct.varchar[1] = biblioteca;

paramStruct.numVarchar = 2;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(ottieni\_biblioteca\_impiego\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, ottieni\_biblioteca\_impiego\_procedure, "Could not bind parameters for library search\n", true);

}

if (mysql\_stmt\_execute(ottieni\_biblioteca\_impiego\_procedure) != 0) {

finish\_with\_stmt\_error(conn, ottieni\_biblioteca\_impiego\_procedure, "An error occurred while looking for the library\n", true);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = biblioteca;

paramStruct.numVarchar = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_result(ottieni\_biblioteca\_impiego\_procedure, param)) {

finish\_with\_stmt\_error(conn, ottieni\_biblioteca\_impiego\_procedure, "Could not retrieve output parameter\n", true);

}

if (mysql\_stmt\_fetch(ottieni\_biblioteca\_impiego\_procedure)) {

finish\_with\_stmt\_error(conn, ottieni\_biblioteca\_impiego\_procedure, "Could not buffer results\n", true);

}

mysql\_stmt\_close(ottieni\_biblioteca\_impiego\_procedure);

return fixString(biblioteca);

}

static void add\_fav\_contact(MYSQL\* conn, char cf[17]) {

MYSQL\_STMT\* aggiungi\_contatto\_preferito\_procedure;

MYSQL\_BIND param[2];

struct param\_type paramStruct;

char options[3] = { '1', '2', '3' };

char op;

char mezzoComPref[45];

printf("\nChoose the favorite type of contact:\n");

printf("\t1) Telephone\n");

printf("\t2) Mobile phone\n");

printf("\t3) E-mail\n");

fflush(stdout);

op = multiChoice("Select type of contact", options, 3);

switch (op) {

case '1':

strcpy(mezzoComPref, "telefono");

break;

case '2':

strcpy(mezzoComPref, "cellulare");

break;

case '3':

strcpy(mezzoComPref, "email");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if(!setup\_prepared\_stmt(&aggiungi\_contatto\_preferito\_procedure,"call aggiungi\_contatto\_preferito(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_contatto\_preferito\_procedure, "Unable to initialize favorite contact insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = mezzoComPref;

paramStruct.varchar[1] = cf;

paramStruct.numVarchar = 2;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_contatto\_preferito\_procedure, param) != 0) {

print\_stmt\_error(aggiungi\_contatto\_preferito\_procedure, "Could not bind parameters for favorite contact insertion\n");

goto again;

}

if (mysql\_stmt\_execute(aggiungi\_contatto\_preferito\_procedure) != 0) {

print\_stmt\_error(aggiungi\_contatto\_preferito\_procedure, "An error occurred while adding the favorite contact\n");

goto again;

}

else {

goto end;

}

again:

mysql\_stmt\_close(aggiungi\_contatto\_preferito\_procedure);

printf("\nLet's try again!\n");

fflush(stdout);

add\_fav\_contact(conn, cf);

return;

end:

printf("Client's information correctly added\n");

fflush(stdout);

mysql\_stmt\_close(aggiungi\_contatto\_preferito\_procedure);

}

static void add\_contacts(MYSQL\* conn, char cf[17], int numContatti) {

MYSQL\_STMT\* aggiungi\_contatto\_procedure;

MYSQL\_BIND param[3];

struct param\_type paramStruct;

char options[3] = { '1', '2', '3' };

char op;

char recapito[48];

char mezzoCom[45];

for (int i = 0; i < numContatti; i++) {

again:

printf("\nChoose the type of contact:\n");

printf("\t1) Telephone\n");

printf("\t2) Mobile phone\n");

printf("\t3) E-mail\n");

fflush(stdout);

op = multiChoice("Select type of contact", options, 3);

printf("Contact number / e-mail: ");

fflush(stdout);

fgets(recapito, 48, stdin);

recapito[strlen(recapito) - 1] = '\0';

switch (op) {

case '1':

strcpy(mezzoCom, "telefono");

break;

case '2':

strcpy(mezzoCom, "cellulare");

break;

case '3':

strcpy(mezzoCom, "email");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if(!setup\_prepared\_stmt(&aggiungi\_contatto\_procedure, "call aggiungi\_contatto  
(?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_contatto\_procedure, "Unable to initialize contact insertion statement", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = recapito;

paramStruct.varchar[1] = mezzoCom;

paramStruct.varchar[2] = cf;

paramStruct.numVarchar = 3;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_contatto\_procedure, param) != 0) {

print\_stmt\_error(aggiungi\_contatto\_procedure, "Could not bind parameters for contact insertion\n");

mysql\_stmt\_close(aggiungi\_contatto\_procedure);

printf("\nLet's try again!\n");

fflush(stdout);

goto again;

}

if (mysql\_stmt\_execute(aggiungi\_contatto\_procedure) != 0) {

print\_stmt\_error(aggiungi\_contatto\_procedure, "Could not bind parameters for contact insertion\n");

mysql\_stmt\_close(aggiungi\_contatto\_procedure);

printf("\nLet's try again!\n");

fflush(stdout);

goto again;

}

mysql\_stmt\_close(aggiungi\_contatto\_procedure);

}

add\_fav\_contact(conn, cf);

}

static void add\_client(MYSQL\* conn) {

MYSQL\_STMT\* aggiungi\_utente\_procedure;

MYSQL\_BIND param[5];

struct param\_type paramStruct;

char cf[24];

char nome[48];

char cognome[48];

char indirizzo[48];

char giornoStr[16];

char meseStr[16];

char annoStr[16];

char numContattiStr[16];

printf("\nTax code: ");

fflush(stdout);

fgets(cf, 24, stdin);

cf[strlen(cf) - 1] = '\0';

printf("Name: ");

fflush(stdout);

fgets(nome, 48, stdin);

nome[strlen(nome) - 1] = '\0';

printf("Surname: ");

fflush(stdout);

fgets(cognome, 48, stdin);

cognome[strlen(cognome) - 1] = '\0';

printf("Address: ");

fflush(stdout);

fgets(indirizzo, 48, stdin);

indirizzo[strlen(indirizzo) - 1] = '\0';

printf("Day of birth [1-31]: ");

fflush(stdout);

fgets(giornoStr, 16, stdin);

printf("Month of birth [1-12]: ");

fflush(stdout);

fgets(meseStr, 16, stdin);

printf("Year of birth: ");

fflush(stdout);

fgets(annoStr, 16, stdin);

printf("Number of contacts: ");

fflush(stdout);

fgets(numContattiStr, 16, stdin);

int giorno = atoi(giornoStr);

int mese = atoi(meseStr);

int anno = atoi(annoStr);

int numContatti = atoi(numContattiStr);

struct MYSQL\_TIME date;

memset(&date, 0, sizeof(date));

date.day = giorno;

date.month = mese;

date.year = anno;

if (!setup\_prepared\_stmt(&aggiungi\_utente\_procedure, "call aggiungi\_utente(?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, aggiungi\_utente\_procedure, "Unable to initialize client insertion statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = cf;

paramStruct.varchar[1] = nome;

paramStruct.varchar[2] = cognome;

paramStruct.varchar[3] = indirizzo;

paramStruct.date[0] = &date;

paramStruct.numVarchar = 4;

paramStruct.numDate = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(aggiungi\_utente\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, aggiungi\_utente\_procedure, "Could not bind parameters for client insertion\n", true);

}

if (mysql\_stmt\_execute(aggiungi\_utente\_procedure) != 0) {

print\_stmt\_error(aggiungi\_utente\_procedure, "An error occurred while adding the client\n");

mysql\_stmt\_close(aggiungi\_utente\_procedure);

}

else {

mysql\_stmt\_close(aggiungi\_utente\_procedure);

add\_contacts(conn, cf, numContatti);

}

}

static void start\_transfer(MYSQL\* conn, char isbn[14], char bibliotecaImp[16], char cf[17], char durata[46]) {

MYSQL\_STMT\* inizia\_trasferimento\_procedure;

MYSQL\_BIND param[5];

struct param\_type paramStruct;

char bibliotecaPart[24];

printf("\nChoose a library: ");

fflush(stdout);

fgets(bibliotecaPart, 24, stdin);

bibliotecaPart[strlen(bibliotecaPart) - 1] = '\0';

if (!setup\_prepared\_stmt(&inizia\_trasferimento\_procedure, "call inizia\_trasferimento  
(?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, inizia\_trasferimento\_procedure, "Unable to initialize transfer start statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = isbn;

paramStruct.varchar[1] = bibliotecaImp;

paramStruct.varchar[2] = bibliotecaPart;

paramStruct.varchar[3] = cf;

paramStruct.varchar[4] = durata;

paramStruct.numVarchar = 5;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(inizia\_trasferimento\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, inizia\_trasferimento\_procedure, "Could not bind parameters for transfer start\n", true);

}

if (mysql\_stmt\_execute(inizia\_trasferimento\_procedure) != 0) {

print\_stmt\_error(inizia\_trasferimento\_procedure, "An error occurred while starting the transfer\n");

mysql\_stmt\_close(inizia\_trasferimento\_procedure);

printf("Let's try again!\n");

fflush(stdout);

start\_loan(conn, bibliotecaImp);

}

dump\_result\_set(conn, inizia\_trasferimento\_procedure, "You are going to lend this copy of selected library:\n");

if (mysql\_stmt\_next\_result(inizia\_trasferimento\_procedure) > 0) {

finish\_with\_stmt\_error(conn, inizia\_trasferimento\_procedure, "Unexpected contidion\n", true);

}

mysql\_stmt\_close(inizia\_trasferimento\_procedure);

}

static void start\_loan(MYSQL\* conn, char bibliotecaImp[16]) {

MYSQL\_STMT\* inizia\_prestito\_procedure;

MYSQL\_BIND param[4];

struct param\_type paramStruct;

char options[3] = { '1', '2', '3' };

char dur;

char isbn[16];

char cf[24];

char durata[48];

printf("\nBook ISBN: ");

fflush(stdout);

fgets(isbn, 16, stdin);

isbn[strlen(isbn) - 1] = '\0';

printf("Client tax code: ");

fflush(stdout);

fgets(cf, 24, stdin);

cf[strlen(cf) - 1] = '\0';

printf("\nChoose the expected duration of the loan:\n");

printf("\t1) One month\n");

printf("\t2) Two months\n");

printf("\t3) Three months\n");

fflush(stdout);

dur = multiChoice("Select expected duration", options, 3);

switch (dur) {

case '1':

strcpy(durata, "1 mese");

break;

case '2':

strcpy(durata, "2 mesi");

break;

case '3':

strcpy(durata, "3 mesi");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if (!setup\_prepared\_stmt(&inizia\_prestito\_procedure, "call inizia\_prestito(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Unable to initialize loan start statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = isbn;

paramStruct.varchar[1] = bibliotecaImp;

paramStruct.varchar[2] = cf;

paramStruct.varchar[3] = durata;

paramStruct.numVarchar = 4;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(inizia\_prestito\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Could not bind parameters for loan start\n", true);

}

if (mysql\_stmt\_execute(inizia\_prestito\_procedure) != 0) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "An error occurred while starting the loan\n", true);

}

short int trasf = 0;

if (mysql\_stmt\_store\_result(inizia\_prestito\_procedure)) { // Funzione di libreria che estrae il result set corrente per poterlo poi processare

fprintf(stderr, " mysql\_stmt\_execute(), 1 failed\n");

fprintf(stderr, " %s\n", mysql\_stmt\_error(inizia\_prestito\_procedure));

exit(0);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.b = &trasf;

paramStruct.numBool = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_result(inizia\_prestito\_procedure, param)) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Unable to bind column parameters\n", true);

}

if (mysql\_stmt\_fetch(inizia\_prestito\_procedure)) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Could not buffer results\n", true);

}

if (mysql\_stmt\_next\_result(inizia\_prestito\_procedure) != 0) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Unexpected condition\n", true);

}

if (!trasf) {

dump\_result\_set(conn, inizia\_prestito\_procedure, "Required book is available in your library. You are going to lend this copy:\n");

if (mysql\_stmt\_next\_result(inizia\_prestito\_procedure) > 0) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Unexpected contidion\n", true);

}

mysql\_stmt\_close(inizia\_prestito\_procedure);

}

else {

dump\_result\_set(conn, inizia\_prestito\_procedure, "Required book is not available in your library. Here there are the libraries you can ask a transfer to:\n");

if (mysql\_stmt\_next\_result(inizia\_prestito\_procedure) > 0) {

finish\_with\_stmt\_error(conn, inizia\_prestito\_procedure, "Unexpected contidion\n", true);

}

mysql\_stmt\_close(inizia\_prestito\_procedure);

start\_transfer(conn, isbn, bibliotecaImp, cf, durata);

}

}

static void get\_isbn(MYSQL\* conn) {

MYSQL\_STMT\* ottieni\_ISBN\_procedure;

MYSQL\_BIND param[2];

struct param\_type paramStruct;

char titolo[48];

char autore[48];

printf("\nTitle: ");

fflush(stdout);

fgets(titolo, 48, stdin);

titolo[strlen(titolo) - 1] = '\0';

printf("Author: ");

fflush(stdout);

fgets(autore, 48, stdin);

autore[strlen(autore) - 1] = '\0';

if (!setup\_prepared\_stmt(&ottieni\_ISBN\_procedure, "call ottieni\_ISBN(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, ottieni\_ISBN\_procedure, "Unable to initialize ISBN query statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = titolo;

paramStruct.varchar[1] = autore;

paramStruct.numVarchar = 2;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(ottieni\_ISBN\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, ottieni\_ISBN\_procedure, "Could not bind parameters for ISBN query\n", true);

}

if (mysql\_stmt\_execute(ottieni\_ISBN\_procedure) != 0) {

finish\_with\_stmt\_error(conn, ottieni\_ISBN\_procedure, "An error occurred while looking for ISBN\n", true);

}

dump\_result\_set(conn, ottieni\_ISBN\_procedure, "");

if (mysql\_stmt\_next\_result(ottieni\_ISBN\_procedure) > 0) {

finish\_with\_stmt\_error(conn, ottieni\_ISBN\_procedure, "Unexpected contidion\n", true);

}

mysql\_stmt\_close(ottieni\_ISBN\_procedure);

}

static void show\_loans(MYSQL\* conn, char bibliotecaImp[16]) {

MYSQL\_STMT\* report\_prestiti\_procedure;

MYSQL\_BIND param[1];

struct param\_type paramStruct;

if (!setup\_prepared\_stmt(&report\_prestiti\_procedure, "call report\_prestiti(?)", conn)) {

finish\_with\_stmt\_error(conn, report\_prestiti\_procedure, "Unable to initialize loans report statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = bibliotecaImp;

paramStruct.numVarchar = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(report\_prestiti\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, report\_prestiti\_procedure, "Could not bind parameters for loans report\n", true);

}

if (mysql\_stmt\_execute(report\_prestiti\_procedure) != 0) {

finish\_with\_stmt\_error(conn, report\_prestiti\_procedure, "An error occurred while retrieving the loans report\n", true);

}

dump\_result\_set(conn, report\_prestiti\_procedure, "\n\nBooks in loan:\n");

if (mysql\_stmt\_next\_result(report\_prestiti\_procedure)) {

finish\_with\_stmt\_error(conn, report\_prestiti\_procedure, "Unexpected contidion\n", true);

}

dump\_result\_set(conn, report\_prestiti\_procedure, "\n\nInformation about clients that have the books:\n");

if (mysql\_stmt\_next\_result(report\_prestiti\_procedure)) {

finish\_with\_stmt\_error(conn, report\_prestiti\_procedure, "Unexpected contidion\n", true);

}

dump\_result\_set(conn, report\_prestiti\_procedure, "\n\nClients' contacts:\n");

if (mysql\_stmt\_next\_result(report\_prestiti\_procedure) > 0) {

finish\_with\_stmt\_error(conn, report\_prestiti\_procedure, "Unexpected contidion\n", true);

}

mysql\_stmt\_close(report\_prestiti\_procedure);

}

static void end\_loan(MYSQL\* conn) {

MYSQL\_STMT\* termina\_prestito\_procedure;

MYSQL\_BIND param[2];

struct param\_type paramStruct;

char idStr[16];

printf("\nBook ID: ");

fflush(stdout);

fgets(idStr, 16, stdin);

int id = atoi(idStr);

float tar = 0;

if (!setup\_prepared\_stmt(&termina\_prestito\_procedure, "call termina\_prestito(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, termina\_prestito\_procedure, "Unable to initialize loan end statement\n", false);

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.integer[0] = &id;

paramStruct.f = &tar;

paramStruct.numInt = 1;

paramStruct.numFloat = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(termina\_prestito\_procedure, param) != 0) {

finish\_with\_stmt\_error(conn, termina\_prestito\_procedure, "Could not bind parameters for loan end\n", true);

}

if (mysql\_stmt\_execute(termina\_prestito\_procedure) != 0) {

print\_stmt\_error(termina\_prestito\_procedure, "An error occurred while ending the loan\n");

goto out;

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.f = &tar;

paramStruct.numFloat = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_result(termina\_prestito\_procedure, param)) {

finish\_with\_stmt\_error(conn, termina\_prestito\_procedure, "Could not retrieve output parameter\n", true);

}

if (mysql\_stmt\_fetch(termina\_prestito\_procedure)) {

finish\_with\_stmt\_error(conn, termina\_prestito\_procedure, "Could not buffer results\n", true);

}

printf("Loan correcty terminated. Penalty is: %.02f euros\n", tar);

fflush(stdout);

out:

mysql\_stmt\_close(termina\_prestito\_procedure);

}

void run\_as\_librarian(MYSQL\* conn) {

char options[6] = { '1', '2', '3', '4', '5', '6' };

char op;

printf("Switching to librarian role...\n");

fflush(stdout);

if (!parse\_config("users/bibliotecario.json", &conf)) {

fprintf(stderr, "Unable to load librarian configuration\n");

exit(EXIT\_FAILURE);

}

if (mysql\_change\_user(conn, conf.db\_username, conf.db\_password, conf.database)) {

fprintf(stderr, "mysql\_change\_user() failed\n");

exit(EXIT\_FAILURE);

}

char\* bibliotecaImp = find\_library(conn);

while (true) {

printf("\033[2J\033[H"); // Clean the shell

printf("\*\*\* Telephone number of your library: %s \*\*\*\n\n", bibliotecaImp);

printf("\*\*\* What should I do for you? \*\*\*\n\n");

printf("1) Add a client\n");

printf("2) Start a book loan\n");

printf("3) Get the ISBN of a book\n");

printf("4) Show current loans\n");

printf("5) End a book loan\n");

printf("6) Quit\n");

fflush(stdout);

op = multiChoice("Select an option", options, 6);

switch (op) {

case '1':

add\_client(conn);

break;

case '2':

start\_loan(conn, bibliotecaImp);

break;

case '3':

get\_isbn(conn);

break;

case '4':

show\_loans(conn, bibliotecaImp);

break;

case '5':

end\_loan(conn);

break;

case '6':

return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

getchar();

}

}

**main.c**

// Questo file contiene la funzione 'main', in cui inizia e termina l'esecuzione del programma.

#include <stdio.h>

#include <stdlib.h>

#include <mysql.h>

#include <string.h>

#include "defines.h"

typedef enum {

ADMINISTRATOR = 1,

LIBRARIAN,

FAILED\_LOGIN

} role\_t;

struct configuration conf;

struct param\_type paramStruct;

static MYSQL\* conn;

static role\_t attempt\_login(MYSQL\* conn, char\* username, char\* password) {

MYSQL\_STMT\* login\_procedure;

MYSQL\_BIND param[3]; // Used both for input and output

int role = 0;

if (!setup\_prepared\_stmt(&login\_procedure, "call login(?, ?, ?)", conn)) {

print\_stmt\_error(login\_procedure, "Unable to initialize login statement\n");

goto err2;

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.varchar[0] = username;

paramStruct.varchar[1] = password;

paramStruct.integer[0] = &role;

paramStruct.numVarchar = 2;

paramStruct.numInt = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_param(login\_procedure, param) != 0) {

print\_stmt\_error(login\_procedure, "Could not bind parameters for login\n");

goto err;

}

if (mysql\_stmt\_execute(login\_procedure) != 0) {

print\_stmt\_error(login\_procedure, "Could not execute login procedure\n");

goto err;

}

memset(param, 0, sizeof(param));

memset(&paramStruct, 0, sizeof(paramStruct));

paramStruct.integer[0] = &role;

paramStruct.numInt = 1;

bind\_par(param, &paramStruct);

if (mysql\_stmt\_bind\_result(login\_procedure, param)) {

print\_stmt\_error(login\_procedure, "Could not retrieve output parameter\n");

goto err;

}

if (mysql\_stmt\_fetch(login\_procedure)) {

print\_stmt\_error(login\_procedure, "Could not buffer results\n");

goto err;

}

mysql\_stmt\_close(login\_procedure);

return role;

err:

mysql\_stmt\_close(login\_procedure);

err2:

return FAILED\_LOGIN;

}

int main() {

role\_t role;

if (!parse\_config((char\*)"users/login.json", &conf)) {

fprintf(stderr, "Unable to load login configuration\n");

exit(EXIT\_FAILURE);

}

conn = mysql\_init(NULL);

if (conn == NULL) {

fprintf(stderr, "mysql\_init() failed\n");

exit(EXIT\_FAILURE);

}

if (mysql\_real\_connect(conn, conf.host, conf.db\_username, conf.db\_password, conf.database, conf.port, NULL, CLIENT\_MULTI\_STATEMENTS | CLIENT\_MULTI\_RESULTS) == NULL) {

fprintf(stderr, "mysql\_real\_connect() failed\n");

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

printf("Username: ");

fflush(stdout);

fgets(conf.username, 128, stdin);

conf.username[strlen(conf.username) - 1] = '\0';

printf("Password: ");

fflush(stdout);

insertPassword(conf.password);

role = attempt\_login(conn, conf.username, conf.password);

switch (role) {

case ADMINISTRATOR:

run\_as\_administrator(conn);

break;

case LIBRARIAN:

run\_as\_librarian(conn);

break;

case FAILED\_LOGIN:

fprintf(stderr, "Invalid credentials\n");

exit(EXIT\_FAILURE);

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

printf("Bye!\n");

mysql\_close(conn);

return 0;

}

1. Indicare con E le entità, con R le relazioni [↑](#footnote-ref-1)
2. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-2)
3. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-3)