Basi di Dati e Conoscenza

Progetto A.A. 2019/2020

SISTEMA INFORMATIVO DI UNA BIBLIOTECA

0239461

Valerio Crecco

Indice

1. Descrizione del Minimondo

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 di 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 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 quale 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 e 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 segnato come “prestato ad altra biblioteca” e viene tenuto traccia di quale 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 euro per ciascun giorno di ritardo (fino ad un massimo di 10 giorni), 0.50 euro per ciascun giorno di ritardo

superiore al decimo. Questa tariffa viene chiesta 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.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
3	Responsabile	Amministratore	Viene assunto che il responsabile di una biblioteca sia l'amministratore della biblioteca.
5	Libro	Copia	Lo stato "in prestito"/"disponibile" nella specifica fa riferimento ad una copia di un libro posseduta da una biblioteca.
8 - 9	Biblioteca scoperta	Turno scoperto	La biblioteca scoperta assumiamo che significhi che ci sono dei turni che non sono ricoperti da nessun bibliotecario.

Specifica disambiguata

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 dell'amministratore ed orario di apertura settimanale.

Il sistema gestisce libri, le cui copie sono disponibili in un sottoinsieme di biblioteche, in numero differente. Ciascuna copia del 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 hanno dei turni scoperti. 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 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 quale 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 una copia del libro è disponibile, il sistema restituisce lo scaffale e il ripiano in cui può essere prelevata la copia. 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, la copia viene segnato come “prestato ad altra biblioteca” e viene tenuto traccia di quale 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 euro per ciascun giorno di ritardo (fino ad un massimo di 10 giorni), 0.50 euro per ciascun giorno di ritardo superiore al decimo. Questa tariffa viene chiesta 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.

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Amministratore	Persona responsabile di una biblioteca.	Responsabile	Biblioteca, Libro.
Biblioteca	Entità che gestisce i libri		Utente, Copia, Bibliotecario, Amministratore,Copia trasferita, Turno.
Bibliotecario	Persona che lavora nella biblioteca		Biblioteca, Amministratore, Turno
Libro	Entità astratta gestita dalla biblioteca	Copia	Copia, Amministratore, Prestito
Copia	Entità fisica gestita dalla biblioteca	Libro	Biblioteca, Libro, Copia in prestito/disponibile/ trasferita

Recapito	Dati per reperire un utente della biblioteca		Utente
Turno	Orario di lavoro dei bibliotecari		Prestito,Biblioteca, Turno ricoperto, Bibliotecario.
Prestito	Prestito di una copia di un libro richiesta da un utente		Utente, Libro, Turno

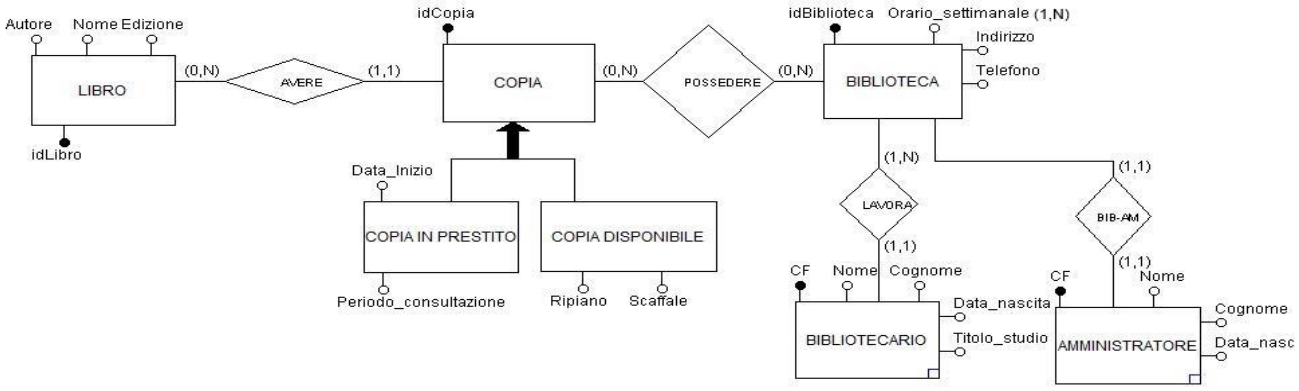
Raggruppamento dei requisiti in insiemi omogenei

Frasi di carattere generale
Si vuole realizzare una base di dati per un circuito di biblioteche, di cui vogliamo rappresentare i dati delle biblioteche del circuito, dei bibliotecari che vi lavorano, dei libri gestiti e degli utenti iscritti al circuito e dei relativi prestiti effettuati.
Frasi relative alle biblioteche del circuito
Per le biblioteche del circuito, identificate da un codice (idBiblioteca), rappresentiamo l'indirizzo, il numero di telefono, l'orario settimanale e gli amministratori di ciascuna di esse.
Frasi relative ai bibliotecari
Per i bibliotecari, identificati dal codice fiscale, rappresentiamo nome, cognome, data di nascita, titolo di studio.
Frasi relative agli amministratori
Gli amministratori che si occupano di realizzare su base mensile i turni della biblioteca e di gestirli in caso rimangano scoperti. Inoltre, si occupano di dismettere quei libri che non hanno avuto richieste di prestito per almeno 10 anni.
Frasi relative ai libri
Per i libri rappresentiamo il titolo, l'autore e l'edizione del libro, che identificano un libro all'interno del circuito.
Frasi relative alle copie dei libri
Per le copie dei libri, identificate da un codice (idCopia), rappresentiamo le copie in prestito e quelle che sono invece disponibili presso determinate biblioteche del circuito. Inoltre, si tiene traccia anche delle copie che vengono prestate da una biblioteca ad un'altra, segnandole come "copie trasferite".
Frasi relative agli utenti
Per gli utenti, identificati da un CF, rappresentiamo alcune informazioni anagrafiche (nome, cognome, genere, data di nascita) e un numero arbitrario di recapiti (da uno fino a tre recapiti di diverso tipo), mediante i quali possono essere contattati dai bibliotecari.
Frasi relative ai turni
Per i turni, identificati da un ID rappresentiamo data, orario di svolgimento, da quali bibliotecari verranno svolti. Inoltre, si tiene traccia di eventuali turni scoperti per motivi di malattia e dei bibliotecari incaricati di ricoprirli.
Frasi relative ai prestiti
Per i prestiti rappresentiamo il turno in cui vengono richiesti, l'utente richiedente e il libro richiesto. I libri possono essere dismessi dagli amministratori del circuito nel caso in cui non vi sia stata una richiesta di una copia di tale libro da almeno 10 anni.

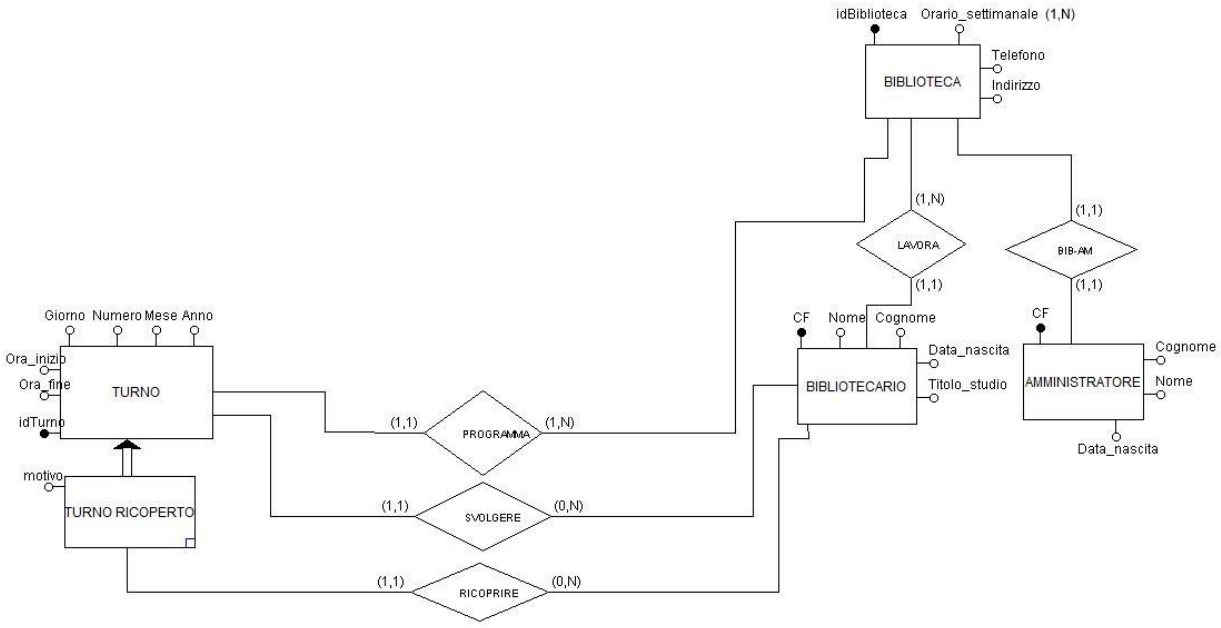
3. Progettazione concettuale

Costruzione dello schema E-R

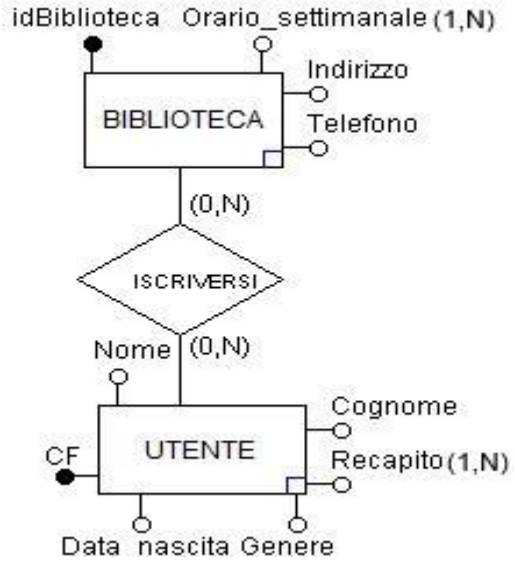
Per la realizzazione dello schema E-R, innanzitutto, è stato distinto il concetto di Libro da quello di Copia. Un libro, infatti, può avere diverse copie nel circuito. Una copia può essere in prestito presso un utente o disponibile in una o più biblioteche del circuito. Si è rappresentato il concetto di Bibliotecario assumendo che un bibliotecario possa lavorare in una sola biblioteca. Infine, si è rappresentato il concetto di Amministratore della biblioteca, anche in questo caso assumendo che un amministratore possa svolgere tale ruolo solo per una delle biblioteche del circuito.



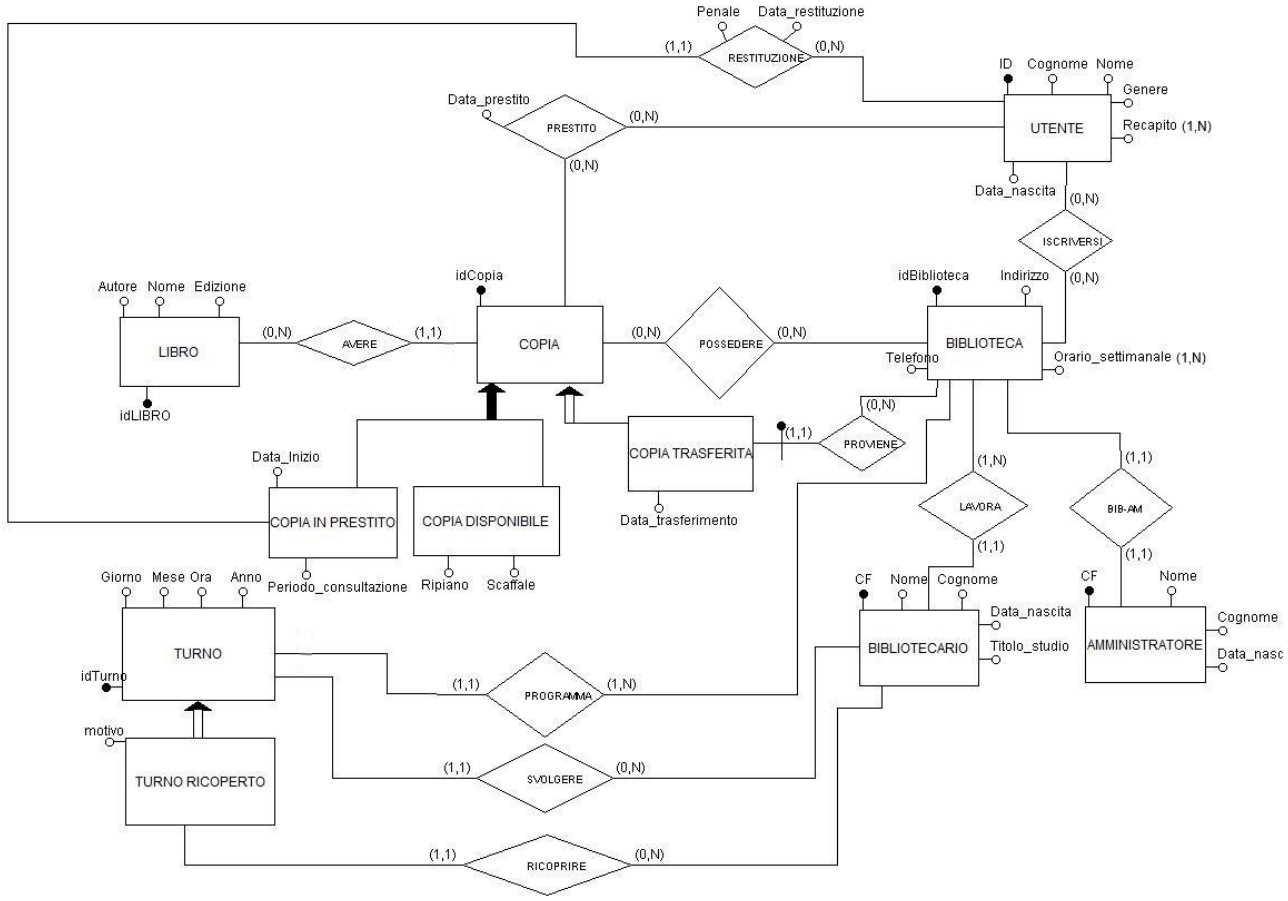
Per quanto riguarda la gestione dei turni di lavoro, sono stati gestiti i turni in relazione al mese di appartenenza. I turni vengono svolti da un bibliotecario e in caso di malattia di quest'ultimo abbiamo dei turni scoperti per i quali è specificato il motivo di assenza del bibliotecario e viene inoltre identificato un bibliotecario sostituto che risulta disponibile.



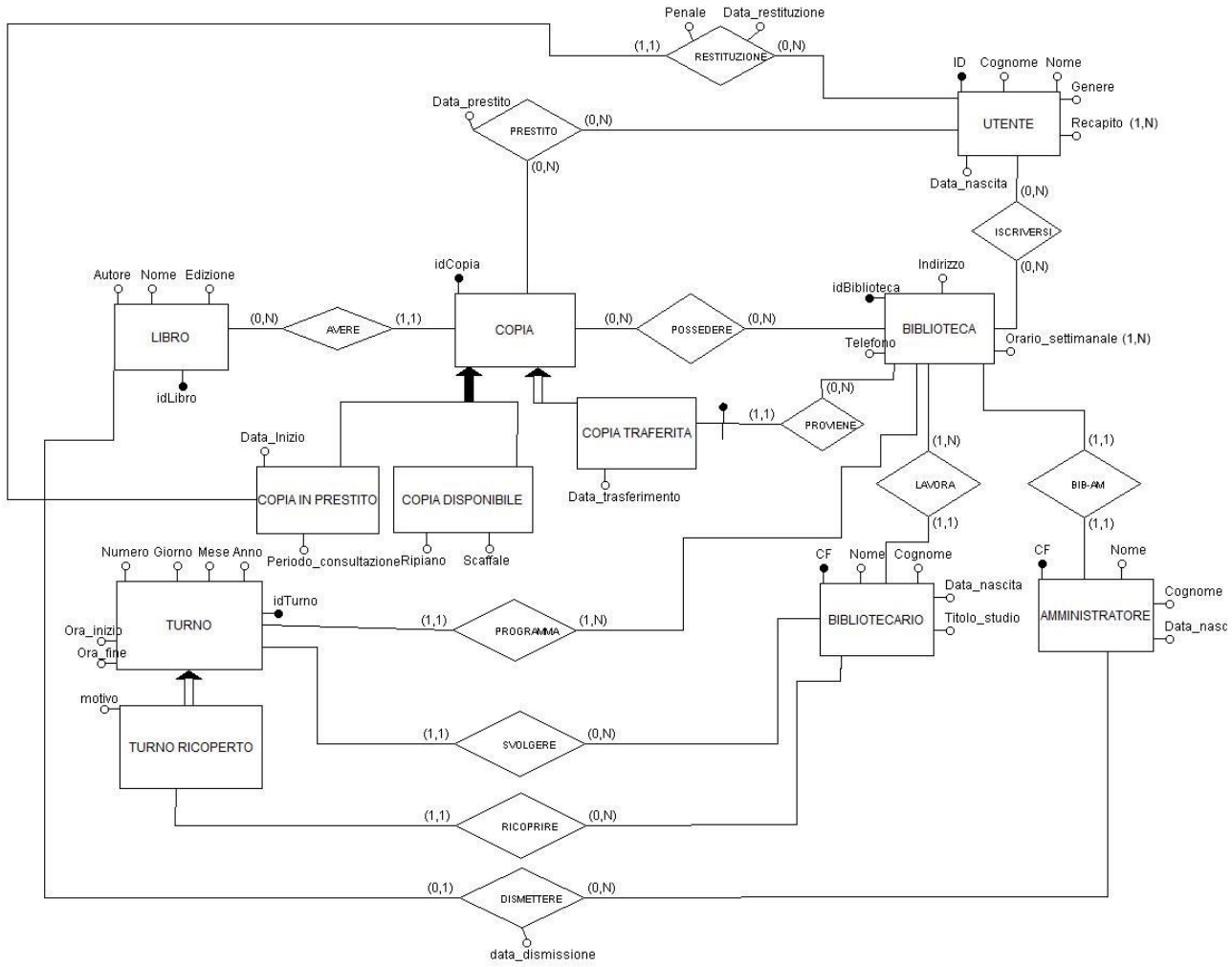
Un utente può iscriversi in più biblioteche o nessuna. Ad ogni utente sono associati un numero arbitrario di recapiti che possono essere del tipo e-mail, telefono o cellulare. Uno di questi è il preferito dell'utente.



Per la gestione dei prestiti e delle richieste di trasferimento di copie da altre biblioteche sono state create le entità Prestito (in relazione con Utente e Copia in modo da poter tenere traccia delle richieste di prestito degli utenti, imponendo che l'utente possa richiedere un solo libro per volta), e Copia Trasferita (in relazione con Turno e Biblioteca, in modo da avere traccia della biblioteca da cui viene spedito il libro e di quella di arrivo).



Integrazione finale



Nello schema E-R finale sono state aggiunte le ultime due relazioni RESTITUZIONE e DISMETTERE al fine di tener traccia rispettivamente dei libri che verranno restituiti e di quelli che verranno dismessi.

Regole aziendali

1. I turni giornalieri non devono avere una durata maggiore alle 8 ore.
2. Il bibliotecario sostituto deve afferire alla stessa biblioteca del bibliotecario sostituito.
3. Il periodo di consultazione di una copia di un libro deve essere al massimo di 3 mesi.
4. La penale da pagare di ottiene moltiplicando 0.10 euro per i primi 10 giorni di ritardo, successivamente si aggiungono 0.50 euro per ogni giorno successivo al decimo.
5. Ogni bibliotecario deve lavorare per un solo turno al giorno.

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Biblioteca	Contiene tutte le informazioni relative alle biblioteche del circuito	idBiblioteca, Indirizzo, Telefono, Orario_settimanale	idBiblioteca
Bibliotecario	Contiene tutti i bibliotecari del circuito	CF, Nome, Cognome, Data_nascita, Tittolo_di_studio, Biblioteca	CF
Amministratore	Contiene tutti gli amministratori del circuito	CF, Nome, Cognome, Data_nascita	CF
Libro	Contiene tutti i libri	idLIBRO, Autore, Nome, Edizione, Amministratore	idLIBRO
Copia	Contiene tutte le copie dei libri	idCopia, idLibro	idCopia
Copia in prestito	Contiene tutte le copie in prestito	idCopia, DataInizio, Periodo_di_consulazione	idCopia

Copia disponibile	Contiene tutte le copie disponibili	idCopia, Scaffale, Ripiano	idCopia
Copia trasferita	Contiene tutte le copie di libri prestate da una biblioteca ad un'altra a seguito di una richiesta di trasferimento	idCopia, Data_trasferimento, idBiblioteca	idCopia, idBiblioteca, Data_trasferimento
Turno	Contiene tutti i turni	idTurno, Numero ,Giorno, Ora_inizio, Ora_fine, Bibliotecario, Biblioteca, Mese, Anno	idTurno
Turno ricoperto	Contiene tutti i turni ricoperti e il relativo motivo	idTurno, Bibliotecario, Motivo	idTurno
Utente	Contiene tutti gli utenti della biblioteca	CF, Nome, Cognome, data_di_nascita, Genere, Recapito	CF

4. Progettazione logica

Volume dei dati

Concetto nello schema	Tipo[1]	Volume atteso
Biblioteca	E	3
Bibliotecario	E	15 (5 bibliotecari per ogni biblioteca)
Amministratore	E	3
Libro	E	500
Copia	E	2500 (almeno 5 copie per ogni libro)
Copia in prestito	E	2500
Copia disponibile	E	2500
Copia trasferita	E	2500
Turno	E	5 (a settimana)
Turno ricoperto	E	5
Utente	E	150 (50 utenti per ogni biblioteca almeno)
Prestito	R	1500 (10 richieste per ogni utente)
Lavora	R	15
BIB-AM	R	15
Possedere	R	2500
Avere	R	2500
Proviene	R	2500
Iscriversi	R	150
Programma	R	31(giorni del mese)
Svolgere	R	31
Ricoprire	R	31
Dismettere	R	500
Restituzione	R	2500

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
1	Realizzazione turni	1/m
2	Report turni	1/g
3	Richiesta malattia	2/g
4	Sostituzione bibliotecario	2/g
5	Registrazione utente	5/g
6	Aggiungi prestito	10/g
7	Recupero copie disponibili	10/g
8	Richiesta di trasferimento della copia	10/g

[1] Indicare con E le entità, con R le relazioni

9	Report libri	10/g
10	Dismissione libro	1/g

Costo delle operazioni

OPERAZIONE 1			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
BIBLIOTECARI	E	2	L
AMMINISTRATORI	E	1	L
BIBLIOTECHE	E	2	L
TURNI	E	1	S

→ ((1*2) + 1 + (1*2) + 2)*1/m = 7/m

OPERAZIONE 2			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
AMMINISTRATORI	E	1	L
BIBLIOTECARI	E	1	L
BIBLIOTECHE	E	1	L
TURNI	E	1	S

→(1+1+1+2)*1/g = 5/g

OPERAZIONE 3			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
BIBLIOTECARI	E	2	L
BIBLIOTECHE	E	1	L
TURNI	E	3	L
TURNI	E	1	S

→((2*1) + 1 + (3*1) + (1*2)) = 8*2/g = 16/g

OPERAZIONE 4			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
AMMINISTRATORI	E	4	L
BIBLIOTECHE	E	6	L
TURNI	E	5	L
BIBLIOTECARI	E	3	L
TURNI	E	2	S
TURNI RICOPERTI	E	1	S

→((4*1) + (6*1) + (5*1) + (3*1) + (2*2) + (1*2)) = 24*2/g = 48/g

OPERAZIONE 5			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
UTENTI	E	1	S
RECAPITI	E	4	S
BIBLIOTECARI	E	1	L
ISCRIVERSI	R	1	S

→((1*2) + (4*2) + 1 + (1*2)) = 13*5/g = 65/g

OPERAZIONE 6			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
PRESTITI	E	1	S
COPIE_IN_PRESTITO	E	1	S
COPIE_DISPONIBILI	E	1	S

→((1*2) + (1*2) + (1*2)) = 6*10/g = 60/g

OPERAZIONE 7			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
BIBLIOTECARI	E	2	L
COPIE_DISPONIBILI	E	2	L
COPIE	E	2	L
LIBRI	E	2	L
POSSEDERE	R	2	L

→((2*1) + (2*1) + (2*1) + (2*1)) = 8*10/g = 80/g

OPERAZIONE 8			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
BIBLIOTECARI	E	2	L
BIBLIOTECHE	E	2	L
POSSEDERE	E	3	L
COPIE	E	3	L
LIBRI	E	3	L
COPIE_DISPONIBILI	E	2	L
COPIE_TRASFERITE	E	1	S
POSSEDERE	E	2	S

→((2*1) + (2*1) + (3*1) + (3*1) + (3*1) + (2*1) + (1*2) + (2*2)) = 21*10/g = 210/g

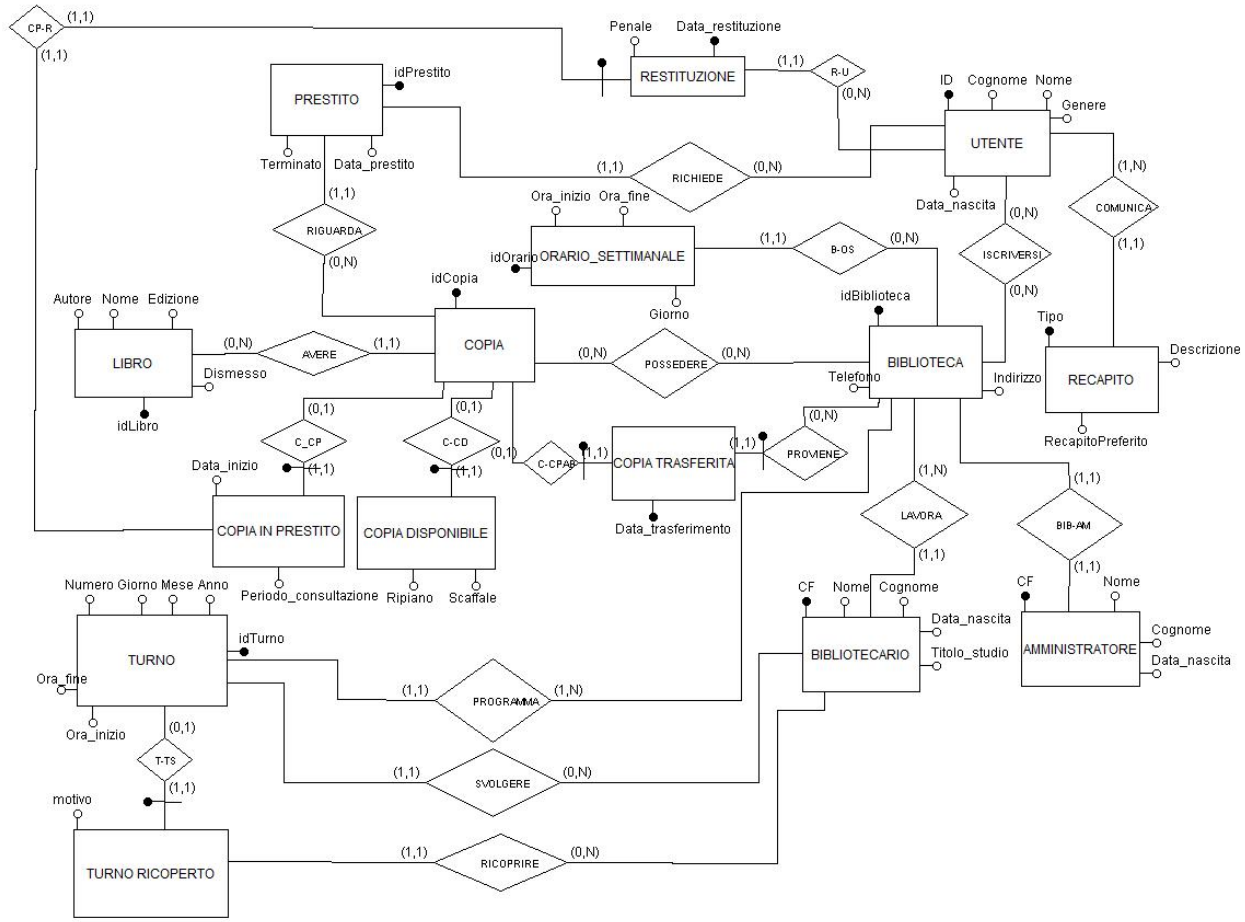
OPERAZIONE 9			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
BIBLIOTECARI	E	1	L
LIBRI	E	1	L
COPIE	E	1	L
PRESTITI	E	1	L
POSSEDERE	R	1	L
UTENTI	E	1	L

→(1+1+1+1+1+1) = 6*10/g = 60/g

OPERAZIONE 10			
CONCETTO	COSTRUTTO	ACCESSI	TIPO
LIBRI	E	1	L
COPIE	E	1	L
PRESTITI	E	1	L
LIBRI	E	1	S

→(1 + 1 + 1 +(1*2)) = 5/g

Ristrutturazione dello schema E-R



Lo schema E-R riportato rappresenta il risultato della ristrutturazione. Tutte le generalizzazioni sono state eliminate sostituendole con delle associazioni. L'associazione 'dismettere' è stata rimossa andando ad aggiungere un attributo booleano 'dismesso' all'interno dell'entità libro cosicché da evitare di avere dei campi con valori nulli. Inoltre in questo modo il costo dell'operazione di dismissione del libro si riduce dovendo andare ad accedere solamente alla tabella libro.

Trasformazione di attributi e identificatori

Traduzione di entità e associazioni

BIBLIOTECHE(**idBiblioteca,** indirizzo, telefono, Amministratore)

BIBLIOTECHE.idBiblioteca ➜ AMMINISTRATORI.**CF**

ORARI_SETTIMANALI(**idOrario**, ora_inizio, ora_fine, giorno, Biblioteca)

ORARI_SETTIMANALI.Biblioteca ➜ BIBLIOTECHE.idBiblioteca

BIBLIOTECARI(**CF**, nome, cognome, data_nascita, titolo_di_studio, BibliotecaDiAppartenenza)

BIBLIOTECARI.BibliotecaDiAppartenenza ➜ BIBLIOTECHE.idBiblioteca

AMMINISTRATORI(**CF,** nome, cognome, data_nascita)

LIBRI(**idLibro**, Autore, Nome, Edizione, Dismesso)

COPIE(**idCopia**, Libro)

COPIE.Libro ➜ LIBRI.idLibro

POSSESSO(**Copia, Biblioteca**)

POSSESSO.Copia ➜ COPIE.idCopia

POSSESSO.Biblioteca ➜ BIBLIOTECHE.idBiblioteca

COPIE_IN_PRESTITO(**idCopia**, data_inizio, periodo_consultazione)

COPIE_IN_PRESTITO.idCopia ➜ COPIE.idCopia

COPIE_DISPONIBILI(**idCopia**, scaffale, ripiano)

COPIE_DISPONIBILI.idCopia ➜ COPIE.idCopia

COPIE_TRASFERITE(**idCopia, DataTrasferimento, BibliotecaPartenza**)

COPIE_TRASFERITE.idCopia ➜ COPIE.idCopia

COPIE_TRASFERITE.Turno ➜ TURNI.idTurno

COPIE_TRASFERITE.BibliotecaPartenza ➜ BIBLIOTECHE.idBiblioteca

RESTITUZIONE(**Copia**, penale, **data_restituzione**, Utente)

RESTITUZIONE.Copia ➜ COPIE_IN_PRESTITO.idCopia

RESTITUZIONE.Utente ➜ UTENTI.CF

TURNI(**idTurno**, ora_inizio ora_fine, numero, giorno, mese, anno, turno_ricoperto, Biblioteca, Bibliotecario)

TURNI.Biblioteca ➜ BIBLIOTECHE.idBiblioteca

TURNI.Bibliotecario ➜ BIBLIOTECARI.CF

TURNI_RICOPERTI(**Turno,** motivo, BibliotecatioSostituito)

TURNI_SCOPERTI.Turno ➜ TURNI.idTurno

TURNI_SCOPERTI.BibliotecarioSostituito ➜ BIBLIOTECARI.CF

UTENTI(**CF,** nome, cognome, genere, data_nascita)

RECAPITI(**Tipo,** Descrizione, RecapitoPreferito, **Utente**)

RECAPITI.Utente ➜ UTENTI.CF

ISCRIVERSI(**Utente, Biblioteca**)

ISCRIVERSI.Utente ➜ UTENTI.CF

ISCRIVERSI.Biblioteca ➜ BIBLIOTECHE.idBiblioteca

PRESTITI(**idPrestito**, Utente**,** Copia, Data_prestito, Terminato)

PRESTITI.Utente ➜ UTENTI.CF

PRESTITI.Copia ➜ COPIE.idCopia

Normalizzazione del modello relazionale

1NF

Una relazione è in prima forma normale se è presente per ogni relazione una chiave primaria, non vi sono gruppi di attributi che si ripetono all'interno di una relazione e le colonne sono indivisibili.
Il modello relazionale precedentemente rappresentato rispetta tali regole.

2NF

Una relazione è in seconda forma normale se è già in prima forma normale e se gli attributi dipendono dall'intera chiave composta, dunque, non ci devono essere dipendenze parziali nelle relazioni.
Anche in questo caso lo schema relazionale soddisfa tale requisito. A tale scopo la relazione RESTITUZIONE è stata tradotta realizzando la tabella RESTITUZIONE perché se avessimo messo gli attributi 'penale', 'data_restituzione' e 'Utente' all'interno della tabella COPIE_IN_PRESTITO oltre ad avere molti valori nulli, non avremo avuto una dipendenza tra tutti gli attributi e la chiave della tabella.

3NF

Una relazione è in terza forma normale se è già in prima e seconda forma normale e non vi sono dipendenze transitive tra gli attributi della relazione e la sua chiave, ovvero tutti gli attributi della relazione dipendono esclusivamente dalla chiave della relazione e non da altri attributi non-chiave.
Nello schema relazionale tutte le tabelle rispettano tale proprietà.

5. Progettazione fisica

Utenti e privilegi

AMMINISTRATORI:

Turno: INSERT, UPDATE, DELETE, SELECT; (perché gli amministratori possono gestire i turni delle biblioteche andando a inserire, modificare, eliminare ed interrogare le informazioni di tale tabella)

Turno ricoperto: INSERT, SELECT; (perché gli amministratori possono decidere un sostituto per ricoprire un turno rimasto scoperto a causa dell'assenza dei bibliotecari)

Bibliotecario: SELECT; (perché gli amministratori possono scegliere un bibliotecario sostituto in caso un turno rimanga scoperto)

Libro: UPDATE, SELECT; (perché gli amministratori possono dismettere un libro che non è stato richiesto in prestito per un periodo maggiore ai 10 anni)

BIBLIOTECARI:

Libro: SELECT; (perché i bibliotecari possono verificare se vi è la disponibilità di un libro presso la biblioteca dove lavorano)

Copia : SELECT; (perché i bibliotecari possono verificare se vi sono copie, di un certo libro, disponibili presso la biblioteca dove lavorano)

Copia disponibile: INSERT, SELECT, DELETE; (perché i bibliotecari possono visualizzare le copie disponibili, inserire in tale tabella i libri che tornano disponibili a seguito di una restituzione e rimuovere da tale tabella le copie che non sono più disponibili perché prestate ad un utente)

Copia in prestito: INSERT, SELECT, DELETE; (perché i bibliotecari possono visualizzare le copie che sono attualmente in prestito, inserire le copie che vengono date in prestito e rimuovere da qui le copie che tornano disponibili a seguito della restituzione)

Copia trasferita: INSERT, SELECT, DELETE; (perché i bibliotecari possono richiedere il trasferimento di un libro)

Possedere: INSERT, SELECT, DELETE; (perché i bibliotecari possono verificare presso quali biblioteche sono disponibili determinate copie dei libri)

Turno: UPDATE; (perché i bibliotecari possono far richiesta di malattia andando a segnare il turno in cui mancheranno come "scoperto")

Prestito: SELECT;

Recapito: SELECT; (perché i bibliotecari hanno la possibilità di consultare i recapiti degli utenti iscritti per sollecitare la restituzione)

UTENTI:

Iscriversi: INSERT; (perché gli utenti possono iscriversi alle biblioteche del circuito)

Utente: INSERT; (perché gli utenti possono iscriversi come utenti del sistema)

Recapito: INSERT; (perché gli utenti possono fornire un insieme di contatti alla biblioteca a cui si iscrivono)

Libro: SELECT; (perché gli utenti possono richiedere in prestito i libri del circuito)

Copia in prestito: INSERT; (perché gli utenti possono decidere di consulatare un copia presa in prestito per un periodo di 1,2 o 3 mesi)

Prestito: INSERT, UPDATE; (perché gli utenti possono prendere in prestito i libri del circuito e nel momento della restituzione, i prestiti vengono segnati come terminati)

Restituzione: INSERT; (perché gli utenti devono restituire i libri presi in prestito)

Strutture di memorizzazione

Tabella BIBLIOTECA		
Attributo	**Tipo di dato**	**Attributi**[2]
idBiblioteca	INT	PK, NN, AI
Indirizzo	VARCHAR(45)	NN
Telefono	VARCHAR(10)	NN
Amministratore	VARCHAR(16)	NN
Tabella ORARIO_SETTIMANALE		
idOrario_settimanale	INT	PK, NN, AI
Ora_inizio	TIME	NN
Ora_fine	TIME	NN
Giorno	VARCHAR(15)	NN
Biblioteca	INT	NN
Tabella BIBLIOTECARIO		
CF	VARCHAR(16)	PK, NN
Nome	VARCHAR(30)	NN
Cognome	VARCHAR(30)	NN
Data_nascita	DATE	NN
Titolo_di_studio	VARCHAR(45)	NN
Biblioteca	INT	NN
Tabella AMMINISTRATORE		
CF	VARCHAR(45)	PK, NN
Nome	VARCHAR(30)	NN
Cognome	VARCHAR(30)	NN
Data_nascita	DATE	NN
Tabella UTENTE		
CF	VARCHAR(16)	PK, NN
Nome	VARCHAR(30)	NN
Cognome	VARCHAR(30)	NN
Data_nascita	DATE	NN
Genere	VARCHAR(5)	NN
Tabella ISCRIVERSI		
Biblioteca	INT	PK, NN

[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.

Utente	VARCHAR(16)	PK, NN
Tabella RECAPITO		
Tipo	INT	PK, NN
Descrizione	VARCHAR(45)	NN
Recapito_preferito	TINYINT	NN
Utente	VARCHAR(16)	PK, NN
Tabella LIBRO		
idLibro	INT	PK, NN, AI
Nome	VARCHAR(45)	NN
Autore	VARCHAR(45)	NN
Edizione	VARCHAR(10)	NN
Dismesso	TINYINT	NN
Tabella COPIA		
idCopia	INT	PK, NN, AI
Libro	INT	NN
Tabella POSSEDERE		
Copia	INT	PK, NN
Biblioteca	INT	PK, NN
Tabella COPIA IN PRESTITO		
Data_inizio	DATE	NN
Periodo_consultazione	INT	NN
Copia	INT	PK, NN
Tabella COPIA DISPONIBILE		
Ripiano	INT	NN
Scaffale	INT	NN
Copia	INT	PK, NN
Tabella COPIA TRASFERITA		
Copia	INT	PK, NN
BibliotecaPartenza	INT	PK, NN
Turno	INT	PK, NN
Tabella TURNO		
idTurno	INT	PK, NN, AI
Ora_inizio	TIME	NN
Ora_fine	TIME	NN
Numero	INT	NN
Giorno	VARCHAR(15)	NN
Mese	VARCHAR(15)	NN
Anno	INT	NN
Turno_coperto	TINYINT	NN
Biblioteca	INT	NN
BibliotecarioSostituto	VARCHAR(16)	NN
Tabella TURNO RICOPERTO		
Motivo	VARCHAR(50)	NN
Turno	INT	PK, NN
BibliotecarioSostituito	VARCHAR(16)	NN
Tabella RESTITUZIONE		
Penale	FLOAT	NN
Data_restituzione	DATE	PK, NN

Utente	INT	NN
CopiaInPrestito	INT	PK, NN
Tabella PRESTITO		
idPrestito	INT	PK, NN, AI
Utente	VARCHAR(16)	NN
Copia	INT	NN
Data_prestito	DATE	NN
Terminato	TINYINT	NN

Indici

Tabella <nome>	
Indice <nome>	**Tipo[3]:**
Colonna 1	<nome>

[3] IDX = index, UQ = unique, FT = full text, PR = primary.

Trigger

1 – trigger per realizzare la regola aziendale che afferma che la durata di un prestito (periodo di consultaziome) non può essere superiore ai 3 mesi.

```
CREATE DEFINER = CURRENT_USER TRIGGER
`BD_PROJECT`.`COPIE_IN_PRESTITO_BEFORE_INSERT` BEFORE INSERT ON
`COPIE_IN_PRESTITO` FOR EACH ROW

BEGIN
   if NEW.periodo_consultazione > 3 then
                signal sqlstate '45011' set message_text = 'consultation period too long';
   end if;

END
```

2 – trigger per realizzare la regola aziendale che dice che la durata di un turno non può essere superiore alle 8 ore. Inoltre, vengono realizzati dei controlli sui campi giorno e mese inseriti in input.

```
CREATE DEFINER = CURRENT_USER TRIGGER
`BD_PROJECT`.`TURNI_BEFORE_INSERT` BEFORE INSERT ON `TURNI` FOR
EACH ROW
BEGIN
    if(TIMEDIFF(NEW.ora_fine, NEW.ora_inizio) > '08:00:00') then
                signal sqlstate '45010' set message_text = 'duration of the slots too long';
   end if;

   if(NEW.giorno <> 'lunedi' AND NEW.giorno <> 'martedi' AND NEW.giorno <> 'mercoledi' AND
NEW.giorno <> 'giovedi' AND NEW.giorno <> 'venerdi') then
                signal sqlstate '45012' set message_text = 'Inserted day not valid';
   end if;

   if(NEW.mese <> 'gennaio' AND NEW.mese <> 'febbraio' AND NEW.mese <> 'marzo' AND
NEW.mese <> 'aprile' AND NEW.mese <> 'maggio' AND NEW.mese <> 'giugno' AND NEW.mese <>
```

```
'luglio' AND NEW.mese <> 'settembre' AND NEW.mese <> 'ottobre' AND NEW.mese <> 'novembre' AND NEW.mese <> 'dicembre') then
		signal sqlstate '45013' set message_text = 'Inserted month not valid';
   end if;
END
```

3 – trigger per realizzare dei controlli sul genere inserito dall'utente in fase di iscrizione di un nuovo utente di una biblioteca

```
CREATE DEFINER = CURRENT_USER TRIGGER `BD_PROJECT`.`UTENTI_BEFORE_INSERT` BEFORE INSERT ON `UTENTI` FOR EACH ROW
BEGIN

   if(NEW.genere <> 'M' AND NEW.genere <> 'm' AND NEW.genere <> 'F' AND NEW.genere <> 'f' AND NEW.genere <> 'other') then
		signal sqlstate '45014' set message_text = 'Gender not valid';
   end if;



END
```

4 – trigger per verificare la validità della e-mail inserita dall'utente

```
CREATE DEFINER = CURRENT_USER TRIGGER `BD_PROJECT`.`RECAPITI_BEFORE_INSERT` BEFORE INSERT ON `RECAPITI` FOR EACH ROW
BEGIN

   if(NEW.tipo = 3) then
		if (NEW.Descrizione NOT 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 '45015' set message_text = 'Email not valid';
		end if;
   end if;



END
```

Eventi

Viste

Stored Procedures e transazioni

1 – add_disease_request(…): questa procedura serve a consentire ad un bibliotecario di segnare un suo turno di lavoro come turno scoperto, cosicché l'amministratore della biblioteca potrà sostituirlo individuando un suo sostituto.

```
CREATE PROCEDURE `add_disease_request` (in var_username varchar(45), in var_ora_inizio time, in var_ora_fine time, in var_numero int, in var_giorno varchar(15), in var_mese varchar(15), in var_anno int, out var_return int)
BEGIN

	declare idBibliotecario varchar(16);
	declare idBiblioteca int;
	declare var_ret int;

	set transaction isolation level read committed;
	start transaction;

	select `BB`.`CF`, `BB`.`BIBLIOTECA_idBIBLIOTECA`
	 from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
	where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
	into idBibliotecario, idBiblioteca;

	select count(*)
	from `TURNI`
	where `Ora_inizio` = var_ora_inizio and `Ora_fine` = var_ora_fine
				and `Numero` = var_numero
				and `Giorno` = var_giorno
				and `Mese` = var_mese
				and `Anno` = var_anno
				and `BIBLIOTECA_idBIBLIOTECA` = idBiblioteca
				and `BIBLIOTECARIO_CF` = idBibliotecario
				into var_ret;

	if var_ret = 1 then
		update `TURNI` set `turno_Coperto` = 0 where `Ora_inizio` = var_ora_inizio
					and `Ora_fine` = var_ora_fine
					and `Numero` = var_numero
					and `Giorno` = var_giorno
					and `Mese` = var_mese
					and `Anno` = var_anno
```

```
                                    and `BIBLIOTECA_idBIBLIOTECA` = idBiblioteca
                                    and `BIBLIOTECARIO_CF` = idBibliotecario;
            set var_return = var_ret;
    else
            set var_return = 0;
    end if;

    commit;

END
```

2 – add_librarian_sub(…): questa procedura serve per tener traccia dei turni scoperti per un determinato motivo, e per aggiornare i turni andando andando a modificare la riga della tabella turni con il nuovo bibliotecario che svolgerà tale turno e marcando tale turno come "coperto".

```
CREATE PROCEDURE `add_librarian_sub` (in var_motivo varchar(50), in var_idTurno int, in var_cf varchar(16))
BEGIN
    set transaction isolation level read uncommitted;
    start transaction;

    insert into `TURNI_RICOPERTI` (`Motivo`, `TURNO_idTURNO`, `BIBLIOTECARIO_CF`) values (var_motivo, var_idTurno, var_cf);
    update `TURNI` set `BIBLIOTECARIO_CF` = var_cf where `idTURNO` = var_idTurno;
    update `TURNI` set `turno_Coperto` = 1 where `idTURNO` = var_idTurno;

    commit;
END
```

3 – add_loan(…): questa procedura si occupa di andare ad inserire un nuovo prestito nella tabella contenente lo storico di tutti i prestiti, e va ad aggiornare le tabelle che contengono le informazioni sulle copie attualmente in prestito ed attualmente non disponibili

```
CREATE PROCEDURE `add_loan` (in var_id_utente varchar(16), in var_id_copia int, in var_terminato boolean, in var_periodo_consultazione int, out var_id_prestito int)
BEGIN

    declare exit handler for sqlexception
     begin
            rollback;
            resignal;
     end;

    set transaction isolation level read uncommitted;
```

```
start transaction;

insert into `PRESTITI` (`UTENTE_CF`,`COPIA_idCOPIA`, `Data_prestito`, `Terminato`) values (var_id_utente, var_id_copia, current_date(), var_terminato);
set var_id_prestito = last_insert_id();

insert into `COPIE_IN_PRESTITO` (`Data_inizio`, `Periodo_consultazione`, `COPIA_idCOPIA`) values (current_date(),var_periodo_consultazione, var_id_copia);

delete from `COPIE_DISPONIBILI` where `COPIA_idCOPIA` = var_id_copia;

commit;

END
```

4 – add_slot(…): questa procedura permette agli amministratori di creare un nuovo turno di lavoro

```
CREATE PROCEDURE `add_slot` (in var_ora_inizio time, in var_ora_fine time, in var_numero int, in var_giorno varchar(15), in var_mese varchar(15), in var_anno int, in var_cf varchar(16), out var_idTurno int)

BEGIN

declare var_idBiblioteca int;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

-- ho scelto come livello di isolamento READ UNCOMMITTED

-- perché la lettura avviene su una tabella (BIBLIOTECARI)

-- prepopolata e sulla quale non vengono fatte operazioni

-- di insert, update e delete

set transaction isolation level read uncommitted;
```

```
        start transaction;


        select `B`.`BIBLIOTECA_idBIBLIOTECA`

        from `BIBLIOTECARI` as `B`

        where `B`.`CF` = var_cf into var_idBiblioteca;


         insert into `TURNI`
        (`Ora_inizio`,`Ora_fine`,`Numero`,`Giorno`,`Mese`,`Anno`,`BIBLIOTECA_idBIBLIOTECA`,
        `BIBLIOTECARIO_CF`, `turno_Coperto`) values (var_ora_inizio, var_ora_fine, var_numero,
        var_giorno, var_mese, var_anno, var_idBiblioteca, var_cf, 1);

        set var_idTurno = last_insert_id();


        commit;
END
```

5 – add_user(…): questa procedura permette la creazione e l'iscrizione di un nuovo utente ad una determinata biblioteca, prendendo in input le sue informazioni generali.

CREATE PROCEDURE `add_user` (in var_username VARCHAR(45), in var_password VARCHAR(30), in var_role VARCHAR(6), in var_CF VARCHAR(16), in var_name VARCHAR(30),

```
in var_surname varchar(30), in var_birth date, in var_gender varchar(5), in var_telefono varchar(15), in var_cellulare varchar(15), in var_email varchar(45), in var_pref_recap int, in var_bibliotecario varchar(45))
BEGIN

    declare idBibliotecario int;
    declare idBiblioteca int;

    declare exit handler for sqlexception
    begin
        rollback;
        resignal;
    end;

  -- ho scelto come livello di isolamento read uncommitted perché, nonostante ci sia un'operazione di
  -- lettura, la tabella dal quale si legge (BIBLIOTECARI) è una tabella prepopolata nella quale non è possibile
  -- fare nuovi inserimenti. Ovvero ho assunto che nel circuito i bibliotecari non possano essere ne assunti ne
  -- rimossi

    set transaction isolation level read uncommitted;
    start transaction;

    insert into `UTILIZZATORI_SISTEMA` (`username`,`password`,`ruolo`)
    values (var_username, md5(var_password), var_role);

    insert into `UTENTI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Genere`, `UTILIZZATORI_SISTEMA_username`) values (var_CF, var_name, var_surname, var_birth, var_gender, var_username);

    insert into `RECAPITI` (`Tipo`, `Descrizione`, `UTENTE_CF`, `RecapitoPreferito`) values (1, var_telefono, var_CF, 0);
    insert into `RECAPITI` (`Tipo`, `Descrizione`, `UTENTE_CF`, `RecapitoPreferito`) values (2, var_cellulare, var_CF, 0);
    insert into `RECAPITI` (`Tipo`, `Descrizione`, `UTENTE_CF`, `RecapitoPreferito`) values (3, var_email, var_CF, 0);

    update `RECAPITI` set `RecapitoPreferito` = 1 where `Tipo` = var_pref_recap and `UTENTE_CF` = var_CF;

  -- recupero l'id della biblioteca nella quale lavora il bibliotecario che sta registrando un nuovo utente
```

```
    select `BB`.`BIBLIOTECA_idBIBLIOTECA`
    from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
    where `BB`.`UTILIZZATORI_SISTEMA_username` = var_bibliotecario into idBiblioteca;

    insert into `ISCRIVERSI` (`BIBLIOTECA_idBIBLIOTECA`, `UTENTE_CF`) values (idBiblioteca, var_CF);

    commit;

END
```

6 – check_inserted_copy_id(…): procedura che verifica che l'id di una copia inserito esista nella base di dati.

```
CREATE PROCEDURE `check_inserted_copy_id` (in var_idCopia int, in var_name varchar(45), in var_username varchar(45))
BEGIN

    declare var_num_rows int;
    declare var_idBiblioteca int;

    declare exit handler for sqlexception
    begin
        rollback;
        resignal;
    end;

    set transaction read only;
    set transaction isolation level read committed;
    start transaction;

  -- recupero l'id della biblioteca in cui lavora il bibliotecario
    select `BB`.`BIBLIOTECA_idBIBLIOTECA`
    from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `US` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `US`.`username`)
    where `US`.`username` = var_username into var_idBiblioteca;


    select count(*)
    from `COPIE` as `C` join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
                join `POSSEDERE` as `P` on (`C`.`idCopia` = `P`.`COPIA_idCOPIA`)
```

```
        where `C`.`idCopia` = var_idCopia and `L`.`Nome` = var_name
                and `P`.`BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca
                into var_num_rows;

        if(var_num_rows < 1) then
                signal sqlstate '45018' set message_text = "Copy ID not found";
        end if;

        commit;
END
```

7 – check_inserted_librarian_cf(…): procedura che verifica che il cf inserito come input dell'utente esista nella base di dati.

```
CREATE PROCEDURE `check_inserted_librarian_cf` (in var_cf varchar(16), in var_username varchar(45))

BEGIN

        declare var_biblioteca int;
        declare var_num_rows int;

        declare exit handler for sqlexception
        begin
                rollback;
                resignal;
        end;

        set transaction read only;
        set transaction isolation level read committed;
        start transaction;

        select `B`.`idBIBLIOTECA`
        from `AMMINISTRATORI` as `A` join `UTILIZZATORI_SISTEMA` as `U` on (`A`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
                join `BIBLIOTECHE` as `B` on (`A`.`CF` = `B`.`AMMINISTRATORE_CF`)
        where `A`.`UTILIZZATORI_SISTEMA_username` = var_username into var_biblioteca;

        select count(*)
        from `BIBLIOTECARI` as `BB` join `BIBLIOTECHE` as `B` on (`BB`.`BIBLIOTECA_idBIBLIOTECA` = `B`.`idBIBLIOTECA`)
        where `B`.`idBIBLIOTECA` = var_biblioteca and `BB`.`CF` = var_cf into var_num_rows;
```

```
	if(var_num_rows < 1) then
		signal sqlstate '45008' set message_text = "Librarian CF not found";
	else
		select `BB`.`CF`
		from `BIBLIOTECARI` as `BB` join `BIBLIOTECHE` as `B` on (`BB`.`BIBLIOTECA_idBIBLIOTECA` = `B`.`idBIBLIOTECA`)
		where `B`.`idBIBLIOTECA` = var_biblioteca and `BB`.`CF` = var_cf;
	end if;

	commit;
END
```

8 – check_inserted_slot(…): procedura che verifica se le informazioni inserite in input in fase di individuazione del turno da ricoprire sono valide o meno, ovvero se esiste un turno nella base di dati che corrisponde alle informazioni passate in input.

```
CREATE PROCEDURE `check_inserted_slot` (in var_oraInizio time, in var_oraFine time, in var_numero int, in var_giorno varchar(15), in var_mese varchar(15), in var_anno int, in var_bibliotecario varchar(16), in var_username varchar(45), out var_idTurno int)

BEGIN
	declare var_biblioteca int;
	declare var_num_rows int;

	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction read only;
	set transaction isolation level read committed;
	start transaction;

	select `B`.`idBIBLIOTECA`
	from `AMMINISTRATORI` as `A` join `UTILIZZATORI_SISTEMA` as `U` on (`A`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
		join `BIBLIOTECHE` as `B` on (`A`.`CF` = `B`.`AMMINISTRATORE_CF`)
	where `A`.`UTILIZZATORI_SISTEMA_username` = var_username into var_biblioteca;


	select count(*)
```

```
        from `TURNI`
        where `Ora_inizio` = var_oraInizio and `Ora_fine` = var_oraFine
                        and `Numero` = var_numero
                        and `Giorno` = var_giorno
                        and `Mese` = var_mese
                        and `Anno` = var_anno
                        and `BIBLIOTECA_idBIBLIOTECA` = var_biblioteca
                        and `BIBLIOTECARIO_CF` = var_bibliotecario
                        and `turno_Coperto` = 0
                        into var_num_rows;

        if(var_num_rows < 1) then
                signal sqlstate '45007' set message_text = "Slot not found";
        else
                select `idTURNO`
                from `TURNI`
                where `Ora_inizio` = var_oraInizio and `Ora_fine` = var_oraFine
                        and `Numero` = var_numero
                        and `Giorno` = var_giorno
                        and `Mese` = var_mese
                        and `Anno` = var_anno
                        and `BIBLIOTECA_idBIBLIOTECA` = var_biblioteca
                        and `BIBLIOTECARIO_CF` = var_bibliotecario
                        into var_idTurno;
        end if;
        commit;
END
```

9 – dismiss_book(…): procedura che permette ad un amministratore di andare a dismettere un libro che non viene richiesto in prestito per almeno 10 anni.

```
CREATE PROCEDURE `dismiss_book` (out var_count int)
BEGIN

        declare var_diff int;
        declare var_idLibro int;
        declare done int default false;
        declare cur cursor for select DATEDIFF(current_date(), `P`.`Data_prestito`), `L`.`idLIBRO` from `LIBRI` as `L` join `COPIE` as `C` on (`L`.`idLIBRO` = `C`.`LIBRO_idLIBRO`) join `PRESTITI` as `P` on (`C`.`idCOPIA` = `P`.`COPIA_idCOPIA`) where `Dismesso` = 0;
        declare continue handler for not found set done = true;
```

```
declare exit handler for sqlexception
begin
        rollback;
        resignal;
end;

set transaction isolation level read committed;
start transaction;

set var_count = 0;
open cur;

read_loop: loop

fetch cur into var_diff, var_idLibro;
        if done then
                leave read_loop;
        end if;

if((var_diff/365) >= 10) then
                update `LIBRI` set `Dismesso` = 1 where `idLIBRO` = var_idLibro
                and `Dismesso` = 0;
                set var_count = var_count + 1;
end if;
end loop;
close cur;
commit;

END
```

10 – find_slot_to_sub(…): procedura che permette di mostrare a schermo i turni scoperti

```
CREATE PROCEDURE `find_slots_to_sub` (in var_username varchar(45))
BEGIN

        declare var_idBiblioteca int;
        declare var_num_rows int;

        declare exit handler for sqlexception
        begin
                rollback;
                resignal;
```

```
        end;

        set transaction read only;
        set transaction isolation level repeatable read;
        start transaction;

        select `B`.`idBIBLIOTECA`
        from `AMMINISTRATORI` as `A` join `UTILIZZATORI_SISTEMA` as `U`
                on (`A`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
                join `BIBLIOTECHE` as `B` on (`A`.`CF` = `B`.`AMMINISTRATORE_CF`)
        where `A`.`UTILIZZATORI_SISTEMA_username` = var_username into var_idBiblioteca;

        select count(*)
        from `TURNI`
        where `turno_Coperto` = 0 and `BIBLIOTECA_idBIBLIOTECA` =  var_idBiblioteca
        into var_num_rows;

        if(var_num_rows < 1) then
                signal sqlstate '45001' set message_text = "All slots are occupeted";
        else
                select `idTurno`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`,
                `BIBLIOTECARIO_CF`
                from `TURNI`
                where `turno_Coperto` = 0 and `BIBLIOTECA_idBIBLIOTECA` =  var_idBiblioteca;
        end if;

        commit;
END
```

11 – free_slot_report(…): procedura che permette di generare il report riguardante i turni scoperti

```
CREATE PROCEDURE `free_slot_report` (in var_username varchar(45))
BEGIN

        declare var_idBiblioteca int;
        set transaction isolation level read committed;
        start transaction;

   -- recupero l'id della biblioteca della quale è responsabile l'amminastratore
        select `B`.`idBIBLIOTECA`
        from `AMMINISTRATORI` as `A` join `UTILIZZATORI_SISTEMA` as `U` on
        (`A`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
```

```
        join `BIBLIOTECHE` as `B` on (`A`.`CF` = `B`.`AMMINISTRATORE_CF`)
    where `A`.`UTILIZZATORI_SISTEMA_username` = var_username
    into var_idBiblioteca;


    select `T`.`idTURNO`,`B`.`Nome`,`B`.`Cognome`,`T`.`Numero`, `T`.`Giorno`, `T`.`Mese`,
`T`.`Anno`
    from `TURNI` as `T` join `BIBLIOTECARI` as `B` on (`T`.`BIBLIOTECARIO_CF` = `B`.`CF`)
    where `T`.`BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca and `T`.`turno_Coperto` = 0;

    commit;

END
```

12 – loan_restitution(…): procedura che permette ad un bibliotecario di reinserire una copia restituita nel circuito.

```
CREATE PROCEDURE `loan_restitution` (in var_idCopia int,in var_cf varchar(16), in var_data date,
in var_periodo_consultazione int, in var_idPrestito int, in var_ripiano int, in var_scaffale int, out
var_penale float)
BEGIN
    declare var_diff int;
    declare var_num_rows int;

    declare exit handler for sqlexception
    begin
        rollback;
        resignal;
    end;

    set transaction isolation level read committed;
    start transaction;

    select count(*)
    from `UTENTI` as `U` join `PRESTITI` as `P` on (`U`.`CF` = `P`.`UTENTE_CF`)
        join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
         join `COPIE_IN_PRESTITO` as `CP` on (`C`.`idCOPIA` = `CP`.`COPIA_idCOPIA`)
         join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
    where `C`.`idCOPIA` = var_idCopia and `P`.`Data_prestito` = var_data
                and `CP`.`Periodo_consultazione` = var_periodo_consultazione
                and `P`.`idPrestito` = var_idPrestito
                into var_num_rows;
```

```
        if(var_num_rows < 1) then
                signal sqlstate '45017' set message_text = 'Inserted information not valid';
        else
                select DATEDIFF(current_date(), var_data) into var_diff;

                if(var_diff - (var_periodo_consultazione*30) <= 0) then
                        insert into `RESTITUZIONI` (`Penale`, `Data_restituzione`, `UTENTE_CF`, `COPIA_idCOPIA`) values (0.0, current_date(), var_cf, var_idCopia);
                        delete from `COPIE_IN_PRESTITO` where `COPIA_idCOPIA` = var_idCopia;
                        insert into `COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) values (var_ripiano, var_scaffale, var_idCopia);
                        update `PRESTITI` set `Terminato` = 1 where `idPrestito` = var_idPrestito;
                else
                        if((var_diff-(var_periodo_consultazione*30)) <= 10) then
                                set var_penale = 0.1*(var_diff-(var_periodo_consultazione*30));
                        else
                                set var_penale = ((0.1*10) + (((var_diff-(var_periodo_consultazione*30))-10)*0.5));
                        end if;

                        insert into `RESTITUZIONI` (`Penale`, `Data_restituzione`, `UTENTE_CF`, `COPIA_idCOPIA`) values (var_penale, current_date(), var_cf, var_idCopia);
                        delete from `COPIE_IN_PRESTITO` where `COPIA_idCOPIA` = var_idCopia;
                        insert into `COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`)
                        values (var_ripiano, var_scaffale, var_idCopia);
                        update `PRESTITI` set `Terminato` = 1 where `idPrestito` = var_idPrestito;

                end if;
         end if;

        commit;
END
```

13 – loan_restitution_user(…): procedura che permette ad un utente di reiserire nel circuito la copia che si sta restituendo.

```
CREATE PROCEDURE `loan_restitution_user` (in var_idCopia int, in var_username varchar(45), in var_data date, in var_periodo_consultazione int, in var_idPrestito int, in var_ripiano int, in var_scaffale int, out var_penale float)
BEGIN
```

```
declare var_diff int;
declare var_cf varchar(16);
declare var_num_rows int;

declare exit handler for sqlexception
begin
	rollback;
	resignal;
end;

set transaction isolation level read committed;
start transaction;

select `U`.`CF`
from `UTENTI` as `U` join `UTILIZZATORI_SISTEMA` as `US` on (`U`.`UTILIZZATORI_SISTEMA_username` = `US`.`username`)
where `US`.`username` = var_username
into var_cf;

select count(*)
from `UTENTI` as `U` join `PRESTITI` as `P` on (`U`.`CF` = `P`.`UTENTE_CF`)
	join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
	 join `COPIE_IN_PRESTITO` as `CP` on (`C`.`idCOPIA` = `CP`.`COPIA_idCOPIA`)
	 join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
where `C`.`idCOPIA` = var_idCopia and `P`.`Data_prestito` = var_data
		 and `CP`.`Periodo_consultazione` = var_periodo_consultazione
		and `P`.`idPrestito` = var_idPrestito
		into var_num_rows;

if(var_num_rows < 1) then
	signal sqlstate '45017' set message_text = 'Inserted information not valid';
 else

	select DATEDIFF(current_date(), var_data) into var_diff;

	if(var_diff - (var_periodo_consultazione*30) <= 0) then
		insert into `RESTITUZIONI` (`Penale`, `Data_restituzione`, `UTENTE_CF`, `COPIA_idCOPIA`) values (0.0, current_date(), var_cf, var_idCopia);
		delete from `COPIE_IN_PRESTITO` where `COPIA_idCOPIA` = var_idCopia;
		insert into `COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) values (var_ripiano, var_scaffale, var_idCopia);
		update `PRESTITI` set `Terminato` = 1 where `idPrestito` = var_idPrestito;
```

```
            else
                if((var_diff-(var_periodo_consultazione*30)) <= 10) then
                    set var_penale = 0.1*(var_diff-(var_periodo_consultazione*30));
                else
                    set var_penale = ((0.1*10) + (((var_diff-(var_periodo_consultazione*30))-
                    10)*0.5));
                end if;

                insert into `RESTITUZIONI` (`Penale`, `Data_restituzione`, `UTENTE_CF`,
                `COPIA_idCOPIA`) values (var_penale, current_date(), var_cf, var_idCopia);
                delete from `COPIE_IN_PRESTITO` where `COPIA_idCOPIA` = var_idCopia;
                insert into `COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`)
                values (var_ripiano, var_scaffale, var_idCopia);
                update `PRESTITI` set `Terminato` = 1 where `idPrestito` = var_idPrestito;

            end if;
        end if;

        commit;
END
```

14 – loans_report(…): procedura che permette ad un bibliotecario di generare il report riguardante i prestiti della biblioteca in cui lavora.

```
CREATE PROCEDURE `loans_report` (in var_username varchar(45))
BEGIN
        declare idBiblioteca int;
        set transaction read only;
        set transaction isolation level read committed;
        start transaction;

    -- recupero l'id della biblioteca in cui lavora il bibliotecario che sta richiedendo il report
        select `BB`.`BIBLIOTECA_idBIBLIOTECA`
        from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U`
        on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
        where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username into idBiblioteca;

        select `L`.`idLIBRO`, `L`.`Nome`, `C`.`idCOPIA`, `U`.`CF`, `U`.`Nome`, `U`.`Cognome`
        from `LIBRI` as `L` join `COPIE` as `C` on (`L`.`idLIBRO` = `C`.`LIBRO_idLIBRO`)
                join `PRESTITI` as `P` on (`C`.`idCOPIA` = `P`.`COPIA_idCOPIA`)
                join `POSSEDERE` as `PS` on (`C`.`idCOPIA` = `PS`.`COPIA_idCOPIA`)
                join `UTENTI` as `U` on (`P`.`UTENTE_CF` = `U`.`CF`)
```

```
        where `PS`.`BIBLIOTECA_idBIBLIOTECA` = idBiblioteca and `P`.`Terminato` = 0;

        commit;
END
```

15 – login(…): procedura per effettuare il login nel circuito e recuperare il proprio ruolo

```
CREATE PROCEDURE `login`(inout var_username varchar(45), in var_pass varchar(45), out var_role
INT)
BEGIN
        declare var_user_role ENUM('amministratore', 'bibliotecario', 'utente');

        select `ruolo`, `username` from `UTILIZZATORI_SISTEMA`
                where `username` = var_username
    and `password` = md5(var_pass)
    into var_user_role, var_username;

    -- See the corresponding enum in the client
                if var_user_role = 'amministratore' then
                        set var_role = 1;
                elseif var_user_role = 'bibliotecario' then
                        set var_role = 2;
                elseif var_user_role = 'utente' then
                        set var_role = 3;
                else
                        set var_role = 4;
                end if;
END
```

16 – recover_available_book_copies(…): procedura che permette di verificare se il libro richiesto è presente nel circuito

```
CREATE PROCEDURE `recover_available_book_copies` (in var_username varchar(45), in var_titolo
varchar(45), out var_return int)
BEGIN
        declare idBiblioteca int;
        declare var_ret int;

        set transaction read only;
        set transaction isolation level read committed;
        start transaction;
```

```
-- recupero l'id della biblioteca in cui lavora il bibliotecario
        select `BB`.`BIBLIOTECA_idBIBLIOTECA`
        from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
        where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
        into idBiblioteca;

        select count(*)
        from `COPIE_DISPONIBILI` as `CD` join `COPIE` as `C`
                on (`CD`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
                join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
                join `POSSEDERE` as `P` on (`C`.`idCOPIA` = `P`.`COPIA_idCOPIA`)
        where `P`.`BIBLIOTECA_idBIBLIOTECA` = idBiblioteca and `L`.`Nome` = var_titolo
                                and `L`.`Dismesso` = 0 into var_ret;

        if (var_ret = 0) then
                set var_return = 0;
        else
                set var_return = var_ret;
        end if;

        commit;
END
```

17 – show_all_librarians(…): procedura che permette di mostrare tutti i bibliotecari che lavorano in una biblioteca del circuito.

```
CREATE PROCEDURE `show_all_librarians` (in var_username varchar(45))
BEGIN
        declare var_idBiblioteca int;

        set transaction read only;
        set transaction isolation level read committed;
        start transaction;

        select `B`.`idBIBLIOTECA`
        from `AMMINISTRATORI` as `A` join `UTILIZZATORI_SISTEMA` as `U` on (`A`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
                join `BIBLIOTECHE` as `B` on (`A`.`CF` = `B`.`AMMINISTRATORE_CF`)
        where `A`.`UTILIZZATORI_SISTEMA_username` = var_username into var_idBiblioteca;
```

```
	select `B`.`idBIBLIOTECA`,`BB`.`CF`, `BB`.`Nome`, `BB`.`Cognome`
	from `BIBLIOTECARI` as `BB`join `BIBLIOTECHE` as `B` on (`BB`.`BIBLIOTECA_idBIBLIOTECA` = `B`.`idBIBLIOTECA`)
	where `BB`.`BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca;

	commit;
END
```

18 – show_available_copies(…): procedura che permette di mostrare le copie disponibili di un determinato libro

```
CREATE PROCEDURE `show_available_copies` (in var_username varchar(45), in var_titolo varchar(45))
BEGIN
	declare idBiblioteca int;

	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction read only;
	set transaction isolation level read committed;
	start transaction;

	select `BB`.`BIBLIOTECA_idBIBLIOTECA`
	from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
	where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username into idBiblioteca;

	select `CD`.`COPIA_idCOPIA`, `CD`.`Ripiano`, `CD`.`Scaffale`
	from `COPIE_DISPONIBILI` as `CD` join `COPIE` as `C`
	on (`CD`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
			join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
			join `POSSEDERE` as `P` on (`C`.`idCOPIA` = `P`.`COPIA_idCOPIA`)
	where `P`.`BIBLIOTECA_idBIBLIOTECA` = idBiblioteca and `L`.`Nome` = var_titolo
		and `L`.`Dismesso` = 0;

	commit;
END
```

19 – show_dismissed_books(…): procedura che permette di mostrare i libri dismessi dal circuito.

```
CREATE PROCEDURE `show_dismissed_books` ()
BEGIN
	declare var_num_rows int;
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction read only;
	set transaction isolation level repeatable read;
	start transaction;

	select count(*)
	from `LIBRI`
	where `Dismesso` = 1
	into var_num_rows;

	if(var_num_rows < 1) then
		signal sqlstate '45002' set message_text = "There are no dismissed books";
	else
		select `idLIBRO`, `Nome`
		from `LIBRI`
		where `Dismesso` = 1;
	end if;

	commit;
END
```

20 – show_free_librarians(…): procedura che permette di mostrarei bibliotecari disponibili a ricoprireun turno rimasto scoperto.

```
CREATE PROCEDURE `show_free_librarians` (in var_username varchar(45), in var_bibliotecario varchar(16), in var_idTurno int)
BEGIN
	declare var_idBiblioteca int;
	set transaction read only;
	set transaction isolation level read committed;
	start transaction;
```

```
	select `B`.`idBIBLIOTECA`
	from `AMMINISTRATORI` as `A` join `UTILIZZATORI_SISTEMA` as `U` on (`A`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
			join `BIBLIOTECHE` as `B` on (`A`.`CF` = `B`.`AMMINISTRATORE_CF`)
	where `A`.`UTILIZZATORI_SISTEMA_username` = var_username
	into var_idBiblioteca;

	select distinct `BB`.`CF`, `BB`.`Nome`, `BB`.`Cognome`
	from `BIBLIOTECARI` as `BB` join `TURNI` as `T` on (`BB`.`CF` = `T`.`BIBLIOTECARIO_CF`)
	where `BB`.`BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca
		and `T`.`BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca
		and `T`.`BIBLIOTECARIO_CF` != var_bibliotecario
		and `T`.`idTurno` != var_idTurno;

	commit;
END
```

21 – show_librarian_slot(…): procedura che permette di mostrare i turni di lavoro di un bibliotecario.

```
CREATE PROCEDURE `show_librarian_slots` (in var_username varchar(45))
BEGIN
	declare var_idBiblioteca int;
	declare var_cf varchar(16);
	declare var_num_rows int;
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;
	set transaction read only;
	set transaction isolation level repeatable read;
	start transaction;

	select `B`.`idBIBLIOTECA`, `BB`.`CF`
	from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
	join `BIBLIOTECHE` as `B` on (`BB`.`BIBLIOTECA_idBIBLIOTECA` = `B`.`idBiblioteca`)
	where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
	into var_idBiblioteca, var_cf;
```

```
        select count(*)
        from `TURNI`
        where `turno_Coperto` = 1 and `BIBLIOTECA_idBIBLIOTECA` =  var_idBiblioteca
                        and `BIBLIOTECARIO_CF` = var_cf
        into var_num_rows;

         if(var_num_rows < 1) then
                signal sqlstate '45004' set message_text = "The are no slot for this librarian";
        else
                select `idTurno`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`,
                `BIBLIOTECARIO_CF`
                from `TURNI`
                 where `turno_Coperto` = 1 and `BIBLIOTECA_idBIBLIOTECA` =  var_idBiblioteca
                                                        and `BIBLIOTECARIO_CF` = var_cf;
         end if;

        commit;
END
```

22 – show_library_users(…): procedura che permette di mostrare gli utenti iscritti ad una determinata biblioteca del circuito.

```
CREATE PROCEDURE `show_library_users` (in var_username varchar(45))
BEGIN
        declare idBiblioteca int;
        declare var_num_rows int;
        declare exit handler for sqlexception
        begin
                rollback;
                resignal;
        end;

        set transaction read only;
        set transaction isolation level read committed;
        start transaction;

        select `BB`.`BIBLIOTECA_idBIBLIOTECA`
        from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on
        (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
        where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
        into idBiblioteca;
```

```
	select count(*)
	from `UTENTI` as `U` join `ISCRIVERSI` as `I` on (`U`.`CF` = `I`.`UTENTE_CF`)
	where `I`.`BIBLIOTECA_idBIBLIOTECA` = idBiblioteca
	into var_num_rows;

	if(var_num_rows < 1) then
		signal sqlstate '45006' set message_text = "There are no users in this library";
	else
		select `U`.`CF`, `U`.`Nome`,`U`.`Cognome`
		from `UTENTI` as `U` join `ISCRIVERSI` as `I` on (`U`.`CF` = `I`.`UTENTE_CF`)
		where `I`.`BIBLIOTECA_idBIBLIOTECA` = idBiblioteca;
	end if;

	commit;
END
```

23 – show_pending_loans(…): procedura che permette di mostrare ad un utente i prestiti che ha in corso, e dunque di scegliere quale restituire.

```
CREATE PROCEDURE `show_pending_loans` (in var_username varchar(45))
BEGIN
	declare var_num_rows int;
	declare var_utenteCF varchar(16);
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction read only;
	set transaction isolation level repeatable read;
	start transaction;

	select `U`.`CF`
	  from `UTENTI` as `U` join `UTILIZZATORI_SISTEMA` as `US` on (`U`.`UTILIZZATORI_SISTEMA_username` = `US`.`username`)
	 where `US`.`username` = var_username into var_utenteCF;

	select count(*)
	from `UTENTI` as `U` join `PRESTITI` as `P` on (`U`.`CF` = `P`.`UTENTE_CF`)
		join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
		join `COPIE_IN_PRESTITO` as `CP` on (`C`.`idCOPIA` = `CP`.`COPIA_idCOPIA`)
```

```
        join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
    where `U`.`CF` = var_utenteCF and `P`.`Terminato` = 0
    into var_num_rows;

    if(var_num_rows < 1) then
        signal sqlstate '45003' set message_text = "There are no book on loan for this user";
    else
        select `C`.`idCopia`, `L`.`Nome`, `CP`.`Data_inizio` as `Data inizio`,
        `CP`.`Periodo_consultazione` as `Durata`, `P`.`idPRESTITO`
        from `UTENTI` as `U` join `PRESTITI` as `P` on (`U`.`CF` = `P`.`UTENTE_CF`)
            join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
        join `COPIE_IN_PRESTITO` as `CP` on (`C`.`idCOPIA` = `CP`.`COPIA_idCOPIA`)
        join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
        where `U`.`CF` = var_utenteCF and `P`.`Terminato` = 0;

    end if;

    commit;

END
```

24 – show_user_loans(…): procedura che permette ad un bibliotecario di vedere i prestiti in corso di un determinato utente.

```
CREATE PROCEDURE `show_user_loans` (in var_utenteCF varchar(16))
BEGIN
    declare var_num_rows int;
    declare exit handler for sqlexception
    begin
        rollback;
        resignal;
    end;

    set transaction read only;
    set transaction isolation level repeatable read;
    start transaction;

    select count(*)
    from `UTENTI` as `U` join `PRESTITI` as `P` on (`U`.`CF` = `P`.`UTENTE_CF`)
        join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
        join `COPIE_IN_PRESTITO` as `CP` on (`C`.`idCOPIA` = `CP`.`COPIA_idCOPIA`)
        join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
```

```
        where `U`.`CF` = var_utenteCF and `P`.`Terminato` = 0
        into var_num_rows;

        if(var_num_rows < 1) then
                signal sqlstate '45003' set message_text = "There are no book on loan for this user";
        else
                select `C`.`idCopia`, `L`.`Nome`, `CP`.`Data_inizio` as `Data inizio`,
                `CP`.`Periodo_consultazione` as `Durata`, `P`.`idPRESTITO`
                from `UTENTI` as `U` join `PRESTITI` as `P` on (`U`.`CF` = `P`.`UTENTE_CF`)
                        join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
                 join `COPIE_IN_PRESTITO` as `CP` on (`C`.`idCOPIA` = `CP`.`COPIA_idCOPIA`)
                 join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
                where `U`.`CF` = var_utenteCF and `P`.`Terminato` = 0;

        end if;

        commit;
END
```

25 – trasferable_copies(…): procedura che permette di mostrare le copie di un libro che possono essere trasferite

```
CREATE PROCEDURE `transferable_copies` (in var_titolo varchar(45), in var_username varchar(45))
BEGIN
        declare var_num_rows int;
        declare var_idBiblioteca int;
        declare exit handler for sqlexception
        begin
                rollback;
                resignal;
        end;

        set transaction read only;
        set transaction isolation level repeatable read;
        start transaction;

        select `BB`.`BIBLIOTECA_idBIBLIOTECA`
        from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on
        (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
        where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
        into var_idBiblioteca;
```

```
        select count(*)
        from `BIBLIOTECHE` as `B` join `POSSEDERE` as `P` on (`B`.`idBIBLIOTECA` =
`P`.`BIBLIOTECA_idBIBLIOTECA`)
                    join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
                    join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
                  join `COPIE_DISPONIBILI` as `CD` on (`C`.`idCOPIA` = `CD`.`COPIA_idCOPIA`)
        where `L`.`Nome` = var_titolo and B.idBiblioteca != var_idBiblioteca into var_num_rows;

        if(var_num_rows < 1) then
                signal sqlstate '45005' set message_text = "There is no copy of the book in the circuit";
        else
                select `B`.`idBIBLIOTECA`, `CD`.`COPIA_idCOPIA`
                from `BIBLIOTECHE` as `B` join `POSSEDERE` as `P` on (`B`.`idBIBLIOTECA` =
                `P`.`BIBLIOTECA_idBIBLIOTECA`)
                        join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
                        join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
                  join `COPIE_DISPONIBILI` as `CD` on (`C`.`idCOPIA` = `CD`.`COPIA_idCOPIA`)
                where `L`.`Nome` = var_titolo and B.idBiblioteca != var_idBiblioteca;
        end if;

        commit;
END
```

26 – update_copies(…): procedura che permette di aggiornare la disponibilità di una copia trasferita, che dunque non sarà più disponibile nella vecchia biblioteca ma sarà disponibile nella biblioteca nella quale è stata trasferita.

```
CREATE PROCEDURE `update_copies` (in var_idCopia int, in var_idBiblioteca int, in var_username
varchar(45), in var_name varchar(45))
BEGIN
        declare idBiblioteca int;
        declare var_num_rows int;
        declare exit handler for sqlexception
        begin
                rollback;
                resignal;
        end;

        set transaction isolation level read committed;
        start transaction;

        select count(*)
```

```
        from `POSSEDERE` as `P` join `COPIE` as `C` on (`P`.`COPIA_idCOPIA` = `C`.`idCOPIA`)
                join `LIBRI` as `L` on (`C`.`LIBRO_idLIBRO` = `L`.`idLIBRO`)
        where `L`.`Nome` = var_name and `P`.`BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca
                and `C`.`idCopia` = var_idCopia
    into var_num_rows;

    if(var_num_rows < 1) then
        signal sqlstate '45019' set message_text = 'Invalid selected copy';
    else

        insert into `COPIE_TRASFERITE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`, `Data_trasferimento`) values (var_idCopia, var_idBiblioteca, current_date());
        delete from `POSSEDERE` where `COPIA_idCOPIA` = var_idCopia and `BIBLIOTECA_idBIBLIOTECA` = var_idBiblioteca;

        select distinct `BB`.`BIBLIOTECA_idBIBLIOTECA`
        from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
        where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
        into idBiblioteca;

        insert into `POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) value (var_idCopia, idBiblioteca);

    end if;

    commit;
END
```

27 – user_addresses(…): procedura che permette di mostrare il recapito preferito degli utenti iscritti ad una biblioteca al fine di poter permettere ad un bibliotecario di contattare uno di questi iscritti per sollecitare la restituzione di una copia in prestito.

```
CREATE PROCEDURE `user_addresses` (in var_username varchar(45))
BEGIN
    declare idBiblioteca int;
    set transaction read only;
    set transaction isolation level read committed;
    start transaction;

    select `BB`.`BIBLIOTECA_idBIBLIOTECA`
```

```
        from `BIBLIOTECARI` as `BB` join `UTILIZZATORI_SISTEMA` as `U` on
        (`BB`.`UTILIZZATORI_SISTEMA_username` = `U`.`username`)
         where `BB`.`UTILIZZATORI_SISTEMA_username` = var_username
        into idBiblioteca;

        select `RP`.`UTENTE_CF`, `U`.`Nome`, `U`.`Cognome`, `RP`.`Descrizione` as `Recapito`
        from `RECAPITI` as `RP` join `UTENTI` as `U` on (`RP`.`UTENTE_CF` = `U`.`CF`)
                        join `ISCRIVERSI` as `I` on (`U`.`CF` = `I`.`UTENTE_CF`)
        where `I`.`BIBLIOTECA_idBIBLIOTECA` = idBiblioteca and `RP`.`RecapitoPreferito` = 1;

         commit;
END
```

6. Appendice: Implementazione

Codice SQL per instanziare il database

```
CREATE SCHEMA IF NOT EXISTS `BD_PROJECT` DEFAULT CHARACTER SET utf8 ;
USE `BD_PROJECT` ;


-- -----------------------------------------------------
-- Table `BD_PROJECT`.`UTILIZZATORI_SISTEMA`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`UTILIZZATORI_SISTEMA` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(32) NOT NULL,
  `ruolo` ENUM('amministratore', 'bibliotecario', 'utente') NOT NULL,
  PRIMARY KEY (`username`))
ENGINE = InnoDB
AUTO_INCREMENT = 31
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`AMMINISTRATORI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`AMMINISTRATORI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`AMMINISTRATORI` (
  `CF` VARCHAR(16) NOT NULL,
  `nome` VARCHAR(30) NOT NULL,
  `cognome` VARCHAR(30) NOT NULL,
  `data_nascita` DATE NOT NULL,
  `UTILIZZATORI_SISTEMA_username` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CF`),
  INDEX `fk_AMMINISTRATORI_UTILIZZATORI_SISTEMA1_idx` (`UTILIZZATORI_SISTEMA_username` ASC),
```

```
  CONSTRAINT `fk_AMMINISTRATORI_UTILIZZATORI_SISTEMA1`
    FOREIGN KEY (`UTILIZZATORI_SISTEMA_username`)
    REFERENCES `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`BIBLIOTECHE`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`BIBLIOTECHE` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`BIBLIOTECHE` (
  `idBIBLIOTECA` INT(11) NOT NULL AUTO_INCREMENT,
  `Telefono` VARCHAR(10) NOT NULL,
  `Indirizzo` VARCHAR(45) NOT NULL,
  `AMMINISTRATORE_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`idBIBLIOTECA`),
  INDEX `fk_BIBLIOTECA_AMMINISTRATORE1_idx` (`AMMINISTRATORE_CF` ASC),
  CONSTRAINT `fk_BIBLIOTECA_AMMINISTRATORE1`
    FOREIGN KEY (`AMMINISTRATORE_CF`)
    REFERENCES `BD_PROJECT`.`AMMINISTRATORI` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`BIBLIOTECARI`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `BD_PROJECT`.`BIBLIOTECARI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`BIBLIOTECARI` (
  `CF` VARCHAR(16) NOT NULL,
  `Nome` VARCHAR(30) NOT NULL,
  `Cognome` VARCHAR(30) NOT NULL,
  `Data_nascita` DATE NOT NULL,
  `Titolo_di_studio` VARCHAR(45) NOT NULL,
  `BIBLIOTECA_idBIBLIOTECA` INT(11) NOT NULL,
  `UTILIZZATORI_SISTEMA_username` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CF`),
  INDEX `fk_BIBLIOTECARIO_BIBLIOTECA1_idx` (`BIBLIOTECA_idBIBLIOTECA` ASC),
  INDEX `fk_BIBLIOTECARI_UTILIZZATORI_SISTEMA1_idx` (`UTILIZZATORI_SISTEMA_username` ASC),
  CONSTRAINT `fk_BIBLIOTECARIO_BIBLIOTECA1`
    FOREIGN KEY (`BIBLIOTECA_idBIBLIOTECA`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_BIBLIOTECARI_UTILIZZATORI_SISTEMA1`
    FOREIGN KEY (`UTILIZZATORI_SISTEMA_username`)
    REFERENCES `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`LIBRI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`LIBRI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`LIBRI` (
```

```
  `idLIBRO` INT(11) NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Autore` VARCHAR(45) NOT NULL,
  `Edizione` VARCHAR(10) NOT NULL,
  `Dismesso` TINYINT(1) NOT NULL,
  PRIMARY KEY (`idLIBRO`))
ENGINE = InnoDB
AUTO_INCREMENT = 51
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`COPIE`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`COPIE` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`COPIE` (
  `idCOPIA` INT(11) NOT NULL AUTO_INCREMENT,
  `LIBRO_idLIBRO` INT(11) NOT NULL,
  PRIMARY KEY (`idCOPIA`),
  INDEX `fk_COPIA_LIBRO1_idx` (`LIBRO_idLIBRO` ASC),
  CONSTRAINT `fk_COPIA_LIBRO1`
    FOREIGN KEY (`LIBRO_idLIBRO`)
    REFERENCES `BD_PROJECT`.`LIBRI` (`idLIBRO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 101
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`COPIE_DISPONIBILI`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `BD_PROJECT`.`COPIE_DISPONIBILI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`COPIE_DISPONIBILI` (
  `Ripiano` INT(11) NOT NULL,
  `Scaffale` INT(11) NOT NULL,
  `COPIA_idCOPIA` INT(11) NOT NULL,
  PRIMARY KEY (`COPIA_idCOPIA`),
  CONSTRAINT `fk_COPIA DISPONIBILE_COPIA1`
    FOREIGN KEY (`COPIA_idCOPIA`)
    REFERENCES `BD_PROJECT`.`COPIE` (`idCOPIA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `BD_PROJECT`.`COPIE_IN_PRESTITO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`COPIE_IN_PRESTITO` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`COPIE_IN_PRESTITO` (
  `Data_inizio` DATE NOT NULL,
  `Periodo_consultazione` INT(11) NOT NULL,
  `COPIA_idCOPIA` INT(11) NOT NULL,
  PRIMARY KEY (`COPIA_idCOPIA`),
  CONSTRAINT `fk_COPIA IN PRESTITO_COPIA1`
    FOREIGN KEY (`COPIA_idCOPIA`)
    REFERENCES `BD_PROJECT`.`COPIE` (`idCOPIA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `BD_PROJECT`.`COPIE_TRASFERITE`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`COPIE_TRASFERITE` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`COPIE_TRASFERITE` (
  `COPIA_idCOPIA` INT(11) NOT NULL,
  `BIBLIOTECA_idBIBLIOTECA` INT(11) NOT NULL,
  `Data_trasferimento` DATE NOT NULL,
  PRIMARY KEY (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`, `Data_trasferimento`),
  INDEX `fk_COPIA PRESTATA AD ALTRA BIBLIOTECA_BIBLIOTECA1_idx` (`BIBLIOTECA_idBIBLIOTECA`
ASC),
  CONSTRAINT `fk_COPIA PRESTATA AD ALTRA BIBLIOTECA_BIBLIOTECA1`
    FOREIGN KEY (`BIBLIOTECA_idBIBLIOTECA`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_COPIA PRESTATA AD ALTRA BIBLIOTECA_COPIA1`
    FOREIGN KEY (`COPIA_idCOPIA`)
    REFERENCES `BD_PROJECT`.`COPIE` (`idCOPIA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `BD_PROJECT`.`UTENTI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`UTENTI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`UTENTI` (
  `CF` VARCHAR(16) NOT NULL,
  `Nome` VARCHAR(30) NOT NULL,
```

```
  `Cognome` VARCHAR(30) NOT NULL,
  `Data_nascita` DATE NOT NULL,
  `Genere` VARCHAR(5) NOT NULL,
  `UTILIZZATORI_SISTEMA_username` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CF`),
  INDEX `fk_UTENTI_UTILIZZATORI_SISTEMA1_idx` (`UTILIZZATORI_SISTEMA_username` ASC),
  CONSTRAINT `fk_UTENTI_UTILIZZATORI_SISTEMA1`
    FOREIGN KEY (`UTILIZZATORI_SISTEMA_username`)
    REFERENCES `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`ISCRIVERSI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`ISCRIVERSI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`ISCRIVERSI` (
  `BIBLIOTECA_idBIBLIOTECA` INT(11) NOT NULL,
  `UTENTE_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`BIBLIOTECA_idBIBLIOTECA`, `UTENTE_CF`),
  INDEX `fk_BIBLIOTECA_has_UTENTE_BIBLIOTECA1_idx` (`BIBLIOTECA_idBIBLIOTECA` ASC),
  INDEX `fk_ISCRIVERSI_UTENTE1_idx` (`UTENTE_CF` ASC),
  CONSTRAINT `fk_BIBLIOTECA_has_UTENTE_BIBLIOTECA1`
    FOREIGN KEY (`BIBLIOTECA_idBIBLIOTECA`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ISCRIVERSI_UTENTE1`
    FOREIGN KEY (`UTENTE_CF`)
    REFERENCES `BD_PROJECT`.`UTENTI` (`CF`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`ORARI_SETTIMANALI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`ORARI_SETTIMANALI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`ORARI_SETTIMANALI` (
  `ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Ora_inizio` TIME NOT NULL,
  `Ora_fine` TIME NOT NULL,
  `Giorno` VARCHAR(15) NOT NULL,
  `BIBLIOTECA_idBIBLIOTECA` INT(11) NOT NULL,
  INDEX `fk_ORARIO_SETTIMANALE_BIBLIOTECA1_idx` (`BIBLIOTECA_idBIBLIOTECA` ASC),
  PRIMARY KEY (`ID`),
  CONSTRAINT `fk_ORARIO_SETTIMANALE_BIBLIOTECA1`
    FOREIGN KEY (`BIBLIOTECA_idBIBLIOTECA`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`POSSEDERE`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`POSSEDERE` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`POSSEDERE` (
```

```
  `COPIA_idCOPIA` INT(11) NOT NULL AUTO_INCREMENT,
  `BIBLIOTECA_idBIBLIOTECA` INT(11) NOT NULL,
  PRIMARY KEY (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`),
  INDEX `fk_COPIA_has_BIBLIOTECA_BIBLIOTECA1_idx` (`BIBLIOTECA_idBIBLIOTECA` ASC),
  INDEX `fk_COPIA_has_BIBLIOTECA_COPIA1_idx` (`COPIA_idCOPIA` ASC),
  CONSTRAINT `fk_COPIA_has_BIBLIOTECA_BIBLIOTECA1`
    FOREIGN KEY (`BIBLIOTECA_idBIBLIOTECA`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_COPIA_has_BIBLIOTECA_COPIA1`
    FOREIGN KEY (`COPIA_idCOPIA`)
    REFERENCES `BD_PROJECT`.`COPIE` (`idCOPIA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 101
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `BD_PROJECT`.`PRESTITI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`PRESTITI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`PRESTITI` (
  `idPRESTITO` INT(11) NOT NULL AUTO_INCREMENT,
  `COPIA_idCOPIA` INT(11) NOT NULL,
  `Data_prestito` DATE NOT NULL,
  `Terminato` INT NOT NULL,
  `UTENTE_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`idPRESTITO`),
  INDEX `fk_PRESTITO_COPIA1_idx` (`COPIA_idCOPIA` ASC),
  INDEX `fk_PRESTITO_UTENTE1_idx` (`UTENTE_CF` ASC),
```

```
  CONSTRAINT `fk_PRESTITO_COPIA1`
    FOREIGN KEY (`COPIA_idCOPIA`)
    REFERENCES `BD_PROJECT`.`COPIE` (`idCOPIA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PRESTITO_UTENTE1`
    FOREIGN KEY (`UTENTE_CF`)
    REFERENCES `BD_PROJECT`.`UTENTI` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`RESTITUZIONI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`RESTITUZIONI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`RESTITUZIONI` (
  `Data_restituzione` DATE NOT NULL,
  `COPIA_idCOPIA` INT(11) NOT NULL,
  `Penale` FLOAT NOT NULL,
  `UTENTE_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`Data_restituzione`, `COPIA_idCOPIA`),
  INDEX `fk_RESTITUIRE_COPIA1_idx` (`COPIA_idCOPIA` ASC),
  INDEX `fk_RESTITUZIONE_UTENTE1_idx` (`UTENTE_CF` ASC),
  CONSTRAINT `fk_RESTITUIRE_COPIA1`
    FOREIGN KEY (`COPIA_idCOPIA`)
    REFERENCES `BD_PROJECT`.`COPIE` (`idCOPIA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_RESTITUZIONE_UTENTE1`
    FOREIGN KEY (`UTENTE_CF`)
```

```
  REFERENCES `BD_PROJECT`.`UTENTI` (`CF`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `BD_PROJECT`.`RECAPITI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`RECAPITI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`RECAPITI` (
  `Tipo` INT(11) NOT NULL,
  `Descrizione` VARCHAR(45) NOT NULL,
  `RecapitoPreferito` TINYINT(4) NOT NULL,
  `UTENTE_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`Tipo`, `UTENTE_CF`),
  INDEX `fk_Recapito_UTENTE1_idx` (`UTENTE_CF` ASC),
  CONSTRAINT `fk_Recapito_UTENTE1`
    FOREIGN KEY (`UTENTE_CF`)
    REFERENCES `BD_PROJECT`.`UTENTI` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `BD_PROJECT`.`TURNI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`TURNI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`TURNI` (
```

```
  `idTURNO` INT(11) NOT NULL AUTO_INCREMENT,
  `Ora_inizio` TIME NOT NULL,
  `Ora_fine` TIME NOT NULL,
  `Numero` INT(11) NOT NULL,
  `Giorno` VARCHAR(15) NOT NULL,
  `Mese` VARCHAR(15) NOT NULL,
  `Anno` INT(11) NOT NULL,
  `BIBLIOTECA_idBIBLIOTECA` INT(11) NOT NULL,
  `turno_Coperto` TINYINT(1) NOT NULL,
  `BIBLIOTECARIO_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`idTURNO`),
  INDEX `fk_TURNO_BIBLIOTECA1_idx` (`BIBLIOTECA_idBIBLIOTECA` ASC),
  INDEX `fk_TURNO_BIBLIOTECARIO1_idx` (`BIBLIOTECARIO_CF` ASC),
  CONSTRAINT `fk_TURNO_BIBLIOTECA1`
    FOREIGN KEY (`BIBLIOTECA_idBIBLIOTECA`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TURNO_BIBLIOTECARIO1`
    FOREIGN KEY (`BIBLIOTECARIO_CF`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECARI` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `BD_PROJECT`.`TURNI_RICOPERTI`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BD_PROJECT`.`TURNI_RICOPERTI` ;

CREATE TABLE IF NOT EXISTS `BD_PROJECT`.`TURNI_RICOPERTI` (
  `Motivo` VARCHAR(50) NOT NULL,
```

```
  `TURNO_idTURNO` INT(11) NOT NULL,
  `BIBLIOTECARIO_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`TURNO_idTURNO`),
  INDEX `fk_TURNO_RICOPERTO_BIBLIOTECARIO1_idx` (`BIBLIOTECARIO_CF` ASC),
  CONSTRAINT `fk_TURNO SCOPERTO_TURNO1`
    FOREIGN KEY (`TURNO_idTURNO`)
    REFERENCES `BD_PROJECT`.`TURNI` (`idTURNO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TURNO_RICOPERTO_BIBLIOTECARIO1`
    FOREIGN KEY (`BIBLIOTECARIO_CF`)
    REFERENCES `BD_PROJECT`.`BIBLIOTECARI` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



SET SQL_MODE = '';
GRANT USAGE ON *.* TO amministratore;
 DROP USER amministratore;
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
CREATE USER 'amministratore' IDENTIFIED BY 'amministratore';

GRANT INSERT, SELECT, UPDATE, DELETE, TRIGGER ON TABLE `BD_PROJECT`.`TURNI` TO 'amministratore';
GRANT SELECT, INSERT ON TABLE `BD_PROJECT`.`TURNI_RICOPERTI` TO 'amministratore';
GRANT SELECT, UPDATE ON TABLE `BD_PROJECT`.`BIBLIOTECARI` TO 'amministratore';
GRANT SELECT, UPDATE ON TABLE `BD_PROJECT`.`LIBRI` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`add_slot` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`dismiss_book` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`free_slot_report` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`add_user` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`find_slots_to_sub` TO 'amministratore';
```

```
GRANT EXECUTE ON procedure `BD_PROJECT`.`check_inserted_slot` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_free_librarians` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`check_inserted_librarian_cf` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`add_librarian_sub` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_all_librarians` TO 'amministratore';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_dismissed_books` TO 'amministratore';
SET SQL_MODE = '';
GRANT USAGE ON *.* TO bibliotecario;
 DROP USER bibliotecario;
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
CREATE USER 'bibliotecario' IDENTIFIED BY 'bibliotecario';

GRANT SELECT, INSERT, DELETE ON TABLE `BD_PROJECT`.`COPIE` TO 'bibliotecario';
GRANT DELETE, INSERT, SELECT ON TABLE `BD_PROJECT`.`COPIE_DISPONIBILI` TO 'bibliotecario';
GRANT DELETE, INSERT, SELECT ON TABLE `BD_PROJECT`.`COPIE_IN_PRESTITO` TO 'bibliotecario';
GRANT SELECT, DELETE, INSERT ON TABLE `BD_PROJECT`.`COPIE_TRASFERITE` TO 'bibliotecario';
GRANT SELECT ON TABLE `BD_PROJECT`.`LIBRI` TO 'bibliotecario';
GRANT INSERT, SELECT, DELETE ON TABLE `BD_PROJECT`.`POSSEDERE` TO 'bibliotecario';
GRANT UPDATE ON TABLE `BD_PROJECT`.`TURNI` TO 'bibliotecario';
GRANT SELECT ON TABLE `BD_PROJECT`.`RECAPITI` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`add_disease_request` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`add_loan` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_available_copies` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`recover_available_book_copies` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`loans_report` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`update_copies` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`add_user` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`transferable_copies` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_library_users` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`user_addresses` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_user_loans` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`loan_restitution` TO 'bibliotecario';
GRANT EXECUTE ON procedure `BD_PROJECT`.`show_librarian_slots` TO 'bibliotecario';
```

```
GRANT EXECUTE ON procedure `BD_PROJECT`.`check_inserted_copy_id` TO 'bibliotecario';

SET SQL_MODE = '';

GRANT USAGE ON *.* TO utente;

 DROP USER utente;

SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'utente' IDENTIFIED BY 'utente';


GRANT INSERT ON TABLE `BD_PROJECT`.`COPIE_IN_PRESTITO` TO 'utente';

GRANT INSERT ON TABLE `BD_PROJECT`.`ISCRIVERSI` TO 'utente';

GRANT SELECT ON TABLE `BD_PROJECT`.`LIBRI` TO 'utente';

GRANT INSERT, UPDATE ON TABLE `BD_PROJECT`.`PRESTITI` TO 'utente';

GRANT INSERT ON TABLE `BD_PROJECT`.`RECAPITI` TO 'utente';

GRANT INSERT ON TABLE `BD_PROJECT`.`RESTITUZIONI` TO 'utente';

GRANT SELECT ON TABLE `BD_PROJECT`.`UTENTI` TO 'utente';

GRANT EXECUTE ON procedure `BD_PROJECT`.`show_pending_loans` TO 'utente';

GRANT EXECUTE ON procedure `BD_PROJECT`.`loan_restitution_user` TO 'utente';

SET SQL_MODE = '';

GRANT USAGE ON *.* TO login;

 DROP USER login;

SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'login' IDENTIFIED BY 'login';


GRANT EXECUTE ON procedure `BD_PROJECT`.`login` TO 'login';


SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

```
-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`UTILIZZATORI_SISTEMA`
-- -----------------------------------------------------
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('valerio23', '0c88028bf3aa6a6a143ed846f2be1ea4', 'amministratore');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('pippo123', '0c88028bf3aa6a6a143ed846f2be1ea4', 'amministratore');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('forrest123', '0c88028bf3aa6a6a143ed846f2be1ea4', 'amministratore');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('ale1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('matt1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('giada1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('ludo1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('daniel1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('john1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('jim1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('walter1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');
INSERT INTO `BD_PROJECT`.`UTILIZZATORI_SISTEMA` (`username`, `password`, `ruolo`) VALUES ('shelly1', '0c88028bf3aa6a6a143ed846f2be1ea4', 'bibliotecario');

COMMIT;


-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`AMMINISTRATORI`
-- -----------------------------------------------------
```

```
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`AMMINISTRATORI` (`CF`, `nome`, `cognome`, `data_nascita`, `UTILIZZATORI_SISTEMA_username`) VALUES ('ABC23DE057AHKJ90', 'Valerio', 'Crecco', '1997-05-23', 'valerio23');
INSERT INTO `BD_PROJECT`.`AMMINISTRATORI` (`CF`, `nome`, `cognome`, `data_nascita`, `UTILIZZATORI_SISTEMA_username`) VALUES ('DGC12DD050BGKJ80', 'Pippo', 'Baudo', '1950-04-12', 'pippo123');
INSERT INTO `BD_PROJECT`.`AMMINISTRATORI` (`CF`, `nome`, `cognome`, `data_nascita`, `UTILIZZATORI_SISTEMA_username`) VALUES ('APV11SD020RGLF70', 'Forrest', 'Gump', '1954-06-24', 'forrest123');

COMMIT;


-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`BIBLIOTECHE`
-- -----------------------------------------------------
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`, `Telefono`, `Indirizzo`, `AMMINISTRATORE_CF`) VALUES (1, '1234567891', 'Via Rossi', 'ABC23DE057AHKJ90');
INSERT INTO `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`, `Telefono`, `Indirizzo`, `AMMINISTRATORE_CF`) VALUES (2, '2345678912', 'Via Verdi', 'DGC12DD050BGKJ80');
INSERT INTO `BD_PROJECT`.`BIBLIOTECHE` (`idBIBLIOTECA`, `Telefono`, `Indirizzo`, `AMMINISTRATORE_CF`) VALUES (3, '3456789123', 'Via Neri', 'APV11SD020RGLF70');

COMMIT;


-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`BIBLIOTECARI`
-- -----------------------------------------------------
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('DEF23DE057AHKJ90', 'Alessio', 'Rossi', '1997-02-27', 'ingegnere meccanico', 1, 'ale1');
```

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('GHJC23DE057AHKJ9', 'Matteo', 'Verdi', '1997-04-28', 'ingegnere gestionale', 2, 'matt1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('KLP23DE057AHKJ90', 'Giada ', 'Neri', '1997-06-12', 'ingegnere medico', 3, 'giada1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('OIU23DE057AHKJ90', 'Ludovico', 'Gialli', '1997-09-13', 'ingegnere informatico', 1, 'ludo1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('YTR23DE057AHKJ90', 'Daniele', 'Arancioni', '1997-06-24', 'ingegnere gestionale', 2, 'daniel1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('EWQ23DE057AHKJ90', 'John', 'Lennon', '1997-07-23', 'ingegnere elettronico', 3, 'john1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('ZXC23DE057AHKJ90', 'Jim', 'Morrison', '1997-03-12', 'ingegnere civile', 1, 'jim1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('VBN23DE057AHKJ90', 'Walter', 'White', '1997-05-14', 'chimico', 2, 'walter1');

INSERT INTO `BD_PROJECT`.`BIBLIOTECARI` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Titolo_di_studio`, `BIBLIOTECA_idBIBLIOTECA`, `UTILIZZATORI_SISTEMA_username`) VALUES ('MLP23DE057AHKJ90', 'Sheldon', 'Cooper', '1997-08-10', 'fisico', 3, 'shelly1');

COMMIT;

-- ---

-- Data for table `BD_PROJECT`.`LIBRI`

-- ---

START TRANSACTION;

USE `BD_PROJECT`;

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (1, 'Il piccolo principe', 'Antoine De Saint-Exupery', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (2, 'Orgoglio e pregiudizio', 'Jane Austen', 'prima ', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (3, 'Il signore degli anelli', 'J.R.R Tolkien', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (4, 'Se questo è un uomo ', 'Primo Levi', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (5, '1984', 'George Orwell', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (6, 'I promessi sposi', 'Alessandro Manzoni', 'prima ', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (7, 'La Divina Commedia', 'Dante Alighieri', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (8, 'Il nome della rosa ', 'Umberto Eco', 'prima ', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (9, 'Il sentiero dei nidi di ragno ', 'Italo Calvino', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (10, 'Walden ', 'Henry David Thoreaur', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (11, 'Anna Karenina', 'Leo Tolstoy', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (12, 'La coscienza di Zeno ', 'Italo Svevo', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (13, 'Il cacciatore di aquiloni', 'Khaled Hosseini', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (14, 'Amore ai tempi del colera', 'Gabriel Garcia Marquez', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (15, 'Oliver Twist', 'Charles Dickens', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (16, 'Un uomo ', 'Oriana Fallaci', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (17, 'I miserabili', 'Victor Hugo', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (18, 'I fratelli Karamazov', 'Fyodor Dostoyevsky', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (19, 'Macbeth', 'William Shakespeare', 'prima ', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (20 , 'Venti mila leghe sotto i mari', 'Jules Verne ', 'prima ', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (21, 'Io non ho paura', 'Niccolo Ammaniti', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (22, 'Il trono di spade', 'George R.R Martin', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (23, 'Madame Bouvary', 'Gustave Flaubert', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (24, 'Le affinità elettive ', 'Goethe', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (25, 'Decamerone', 'Giovanni Boccaccio', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (26, 'Il vecchio e il mare', 'Hernest Hemingway', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (27, 'Il barone rampante', 'Italo Calvino', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (28, 'Il codive Da Vinci', 'Dan Brown', 'prima ', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (29, 'Amleto', 'William Shakespeare', 'prima', 0);

INSERT INTO `BD_PROJECT`.`LIBRI` (`idLIBRO`, `Nome`, `Autore`, `Edizione`, `Dismesso`) VALUES (30, 'La metamorfosi', 'Franz Kafka', 'prima ', 0);

COMMIT;

-- ---

-- Data for table `BD_PROJECT`.`COPIE`

-- ---

START TRANSACTION;

USE `BD_PROJECT`;

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (1, 1);

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (2, 1);

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (3, 2);

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (4, 2);

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (5, 3);

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (6, 3);

INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (7, 4);

```
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (8, 4);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (9, 5);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (10, 5);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (11, 6);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (12, 6);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (13, 7);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (14, 7);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (15, 8);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (16, 8);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (17, 9);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (18, 9);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (19, 10);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (20, 10);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (21, 11);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (22, 11);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (23, 12);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (24, 12);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (25, 13);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (26, 13);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (27, 14);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (28, 14);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (29, 15);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (30, 15);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (31, 16);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (32, 16);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (33, 17);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (34, 17);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (35, 18);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (36, 18);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (37, 19);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (38, 19);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (39, 20);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (40, 20);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (41, 21);
```

```
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (42, 21);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (43, 22);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (44, 22);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (45, 23);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (46, 23);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (47, 24);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (48, 24);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (49, 25);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (50, 25);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (51, 26);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (52, 26);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (53, 27);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (54, 27);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (55, 28);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (56, 28);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (57, 29);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (58, 29);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (59, 30);
INSERT INTO `BD_PROJECT`.`COPIE` (`idCOPIA`, `LIBRO_idLIBRO`) VALUES (60, 30);

COMMIT;


-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`COPIE_DISPONIBILI`
-- -----------------------------------------------------
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 1);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 2);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 3);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 4);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 5);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 6);
```

```
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 7);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 8);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 9);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 10);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 11);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 12);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 13);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 14);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 15);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 16);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 17);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 18);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 19);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 20);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 21);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 22);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 23);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 24);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 25);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 26);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 27);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 28);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 29);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 30);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 31);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 32);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 33);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 34);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 35);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 36);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 37);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 38);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 39);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 40);
```

```
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 41);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 42);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 43);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 44);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 45);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 46);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 47);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 48);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 49);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 50);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 51);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 52);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 53);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 54);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 55);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 56);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (3, 1, 57);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (1, 1, 58);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 59);
INSERT INTO `BD_PROJECT`.`COPIE_DISPONIBILI` (`Ripiano`, `Scaffale`, `COPIA_idCOPIA`) VALUES (2, 1, 60);

COMMIT;


-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`ORARI_SETTIMANALI`
-- -----------------------------------------------------
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`,
`BIBLIOTECA_idBIBLIOTECA`) VALUES (1, '09:00:00', '17:00:00', 'lunedi', 1);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`,
`BIBLIOTECA_idBIBLIOTECA`) VALUES (2, '09:00:00', '17:00:00', 'martedi', 1);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`,
`BIBLIOTECA_idBIBLIOTECA`) VALUES (3, '09:00:00', '17:00:00', 'mercoledi', 1);
```

```
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (4, '09:00:00', '17:00:00', 'giovedi', 1);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (5, '09:00:00', '17:00:00', 'venerdi', 1);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (6, '09:00:00', '17:00:00', 'lunedi', 2);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (7, '09:00:00', '17:00:00', 'martedi', 2);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (8, '09:00:00', '17:00:00', 'mercoledi', 2);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (9, '09:00:00', '17:00:00', 'giovedi', 2);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (10, '09:00:00', '17:00:00', 'venerdi', 2);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (11, '09:00:00', '17:00:00', 'lunedi', 3);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (12, '09:00:00', '17:00:00', 'martedi', 3);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (13, '09:00:00', '17:00:00', 'mercoledi', 3);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (14, '09:00:00', '17:00:00', 'giovedi', 3);
INSERT INTO `BD_PROJECT`.`ORARI_SETTIMANALI` (`ID`, `Ora_inizio`, `Ora_fine`, `Giorno`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (15, '09:00:00', '17:00:00', 'venerdi', 3);

COMMIT;


-- -----------------------------------------------------
-- Data for table `BD_PROJECT`.`POSSEDERE`
-- -----------------------------------------------------
START TRANSACTION;
USE `BD_PROJECT`;
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (1, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (2, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (3, 1);
```

```
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (4, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (5, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (6, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (7, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (8, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (9, 1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (10,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (11,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (12,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (13,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (14,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (15,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (16,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (17,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (18,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (19,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (20,
1);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (21,
2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (22,
2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (23,
2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (24,
2);
```

```
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (25, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (26, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (27, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (28, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (29, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (30, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (31, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (32, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (33, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (34, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (35, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (36, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (37, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (38, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (39, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (40, 2);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (41, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (42, 3);
```

```
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (43, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (44, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (45, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (46, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (47, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (48, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (49, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (50, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (51, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (52, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (53, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (54, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (55, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (56, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (57, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (58, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (59, 3);
INSERT INTO `BD_PROJECT`.`POSSEDERE` (`COPIA_idCOPIA`, `BIBLIOTECA_idBIBLIOTECA`) VALUES (60, 3);
```

COMMIT;

-- ---

-- Data for table `BD_PROJECT`.`TURNI`

-- ---

START TRANSACTION;

USE `BD_PROJECT`;

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (1, '09:00:00', '17:00:00', 1, 'mercoledi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (2, '09:00:00', '17:00:00', 2, 'giovedi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (3, '09:00:00', '17:00:00', 3, 'venerdi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (4, '09:00:00', '17:00:00', 6, 'lunedi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (5, '09:00:00', '17:00:00', 7, 'martedi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (6, '09:00:00', '17:00:00', 8, 'mercoledi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (7, '09:00:00', '17:00:00', 9, 'giovedi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (8, '09:00:00', '17:00:00', 10, 'venerdi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (9, '09:00:00', '17:00:00', 13, 'lunedi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (10, '09:00:00', '17:00:00', 14, 'martedi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (11, '09:00:00', '17:00:00', 15, 'mercoledi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (12, '09:00:00', '17:00:00', 16, 'giovedi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (13, '09:00:00', '17:00:00', 17, 'venerdi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (14, '09:00:00', '17:00:00', 20, 'lunedi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (15, '09:00:00', '17:00:00', 21, 'martedi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (16, '09:00:00', '17:00:00', 22, 'mercoledi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (17, '09:00:00', '17:00:00', 23, 'giovedi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (18, '09:00:00', '17:00:00', 24, 'venerdi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (19, '09:00:00', '17:00:00', 27, 'lunedi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (20, '09:00:00', '17:00:00', 28, 'martedi', 'settembre', 2021, 1, 1, 'ZXC23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (21, '09:00:00', '17:00:00', 29, 'mercoledi', 'settembre', 2021, 1, 1, 'OIU23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (22, '09:00:00', '17:00:00', 30, 'giovedi', 'settembre', 2021, 1, 1, 'DEF23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (23, '09:00:00', '17:00:00', 1, 'mercoledi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (24, '09:00:00', '17:00:00', 2, 'giovedi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (25, '09:00:00', '17:00:00', 3, 'venerdi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (26, '09:00:00', '17:00:00', 6, 'lunedi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (27, '09:00:00', '17:00:00', 7, 'martedi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (28, '09:00:00', '17:00:00', 8, 'mercoledi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (29, '09:00:00', '17:00:00', 9, 'giovedi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (30, '09:00:00', '17:00:00', 10, 'venerdi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (31, '09:00:00', '17:00:00', 13, 'lunedi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (32, '09:00:00', '17:00:00', 14, 'martedi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (33, '09:00:00', '17:00:00', 15, 'mercoledi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (34, '09:00:00', '17:00:00', 16, 'giovedi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (35, '09:00:00', '17:00:00', 17, 'venerdi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (36, '09:00:00', '17:00:00', 20, 'lunedi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (37, '09:00:00', '17:00:00', 21, 'martedi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (38, '09:00:00', '17:00:00', 22, 'mercoledi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (39, '09:00:00', '17:00:00', 23, 'giovedi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (40, '09:00:00', '17:00:00', 24, 'venerdi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (41, '09:00:00', '17:00:00', 27, 'lunedi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (42, '09:00:00', '17:00:00', 28, 'martedi', 'settembre', 2021, 2, 1, 'VBN23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (43, '09:00:00', '17:00:00', 29, 'mercoledi', 'settembre', 2021, 2, 1, 'YTR23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (44, '09:00:00', '17:00:00', 30, 'giovedi', 'settembre', 2021, 2, 1, 'GHJC23DE057AHKJ9');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (45, '09:00:00', '17:00:00', 1, 'mercoledi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (46, '09:00:00', '17:00:00', 2, 'giovedi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (47, '09:00:00', '17:00:00', 3, 'venerdi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (48, '09:00:00', '17:00:00', 6, 'lunedi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (49, '09:00:00', '17:00:00', 7, 'martedi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (50, '09:00:00', '17:00:00', 8, 'mercoledi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (51, '09:00:00', '17:00:00', 9, 'giovedi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (52, '09:00:00', '17:00:00', 10, 'venerdi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (53, '09:00:00', '17:00:00', 13, 'lunedi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (54, '09:00:00', '17:00:00', 14, 'martedi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (55, '09:00:00', '17:00:00', 15, 'mercoledi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (56, '09:00:00', '17:00:00', 16, 'giovedi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (57, '09:00:00', '17:00:00', 17, 'venerdi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (58, '09:00:00', '17:00:00', 20, 'lunedi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (59, '09:00:00', '17:00:00', 21, 'martedi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (60, '09:00:00', '17:00:00', 22, 'mercoledi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (61, '09:00:00', '17:00:00', 23, 'giovedi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (62, '09:00:00', '17:00:00', 24, 'venerdi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (63, '09:00:00', '17:00:00', 27, 'lunedi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (64, '09:00:00', '17:00:00', 28, 'martedi', 'settembre', 2021, 3, 1, 'MLP23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (65, '09:00:00', '17:00:00', 29, 'mercoledi', 'settembre', 2021, 3, 1, 'EWQ23DE057AHKJ90');

INSERT INTO `BD_PROJECT`.`TURNI` (`idTURNO`, `Ora_inizio`, `Ora_fine`, `Numero`, `Giorno`, `Mese`, `Anno`, `BIBLIOTECA_idBIBLIOTECA`, `turno_Coperto`, `BIBLIOTECARIO_CF`) VALUES (66, '09:00:00', '17:00:00', 30, 'giovedi', 'settembre', 2021, 3, 1, 'KLP23DE057AHKJ90');

COMMIT;

Codice del Front-End

//

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql.h>
#include "defines.h"


typedef enum{
        AMMINISTRATORE = 1,
        BIBLIOTECARIO,
        UTENTE,
        FAILED_LOGIN
} role_t;

struct configuration conf;

static MYSQL *conn;
int role_cho;
char username_g[46];

static role_t attempt_login(MYSQL *conn, char *username, char *password){

        MYSQL_STMT *login_procedure;
        MYSQL_BIND param[4];
        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;
}

//prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING; // INOUT
param[0].buffer = username;
param[0].buffer_length = strlen(username);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param[1].buffer = password;
param[1].buffer_length = strlen(password);

param[2].buffer_type = MYSQL_TYPE_LONG; // OUT
param[2].buffer = &role;
param[2].buffer_length = sizeof(role);


if (mysql_stmt_bind_param(login_procedure, param) != 0) { // Note _param
        print_stmt_error(login_procedure, "Could not bind parameters for login");
        goto err;
}

// Run procedure
if(mysql_stmt_execute(login_procedure) != 0){
        print_stmt_error(login_procedure, "Could not execute login_procedure");
        goto err;
}

// Prepare output parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING; // OUT
param[0].buffer = username;
param[0].buffer_length = strlen(username);

param[1].buffer_type = MYSQL_TYPE_LONG; // OUT
param[1].buffer = &role;
param[1].buffer_length = sizeof(role);


if(mysql_stmt_bind_result(login_procedure, param)){
        print_stmt_error(login_procedure, "Could not performe login_procedure");
        goto err;
}
```

```
	// Retrieve ouput paramenters
	if(mysql_stmt_fetch(login_procedure)){
		print_stmt_error(login_procedure, "Could not performe login_procedure");
		goto err;

	}


	strcpy(username_g, username);
	mysql_stmt_close(login_procedure);
	return role;

	err:
	mysql_stmt_close(login_procedure);
	err2:
	return FAILED_LOGIN;
}



int main(void)
{
	role_t role;

	printf("\n***** SISTEMA INFORMATIVO DI UN CIRCUITO DI BIBLIOTECHE *****\n");

	if(!parse_config("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 (probably out of memory)\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, "Failed to change user.  Error: %s\n", mysql_error(conn));
		mysql_close(conn);
		exit(EXIT_FAILURE);
	}

	printf("\nUsername: ");
	getInput(128, conf.username, false);
	printf("\nPassword: ");
```

```
	getInput(128, conf.password, true);

	role = attempt_login(conn, conf.username, conf.password);

	switch(role) {
		case AMMINISTRATORE:
			run_as_amministratore(conn, username_g);
			break;

		case BIBLIOTECARIO:
			run_as_bibliotecario(conn, username_g);
			break;

		case UTENTE:
			run_as_utente(conn, username_g);
			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("\nBye!\n");

	mysql_close (conn);

	return 0;
}
```

//

amministratore.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#include "defines.h"

#define flush(stdin) while(getchar() != '\n')
```

```
int idBiblioteca, idBiblioteca_sb;
char username_g[46];


static void find_slots_sub(MYSQL *conn){

        MYSQL_STMT *prepared_stmt;
        MYSQL_STMT *prepared_stmt2;
        MYSQL_STMT *prepared_stmt3;
        MYSQL_STMT *prepared_stmt4;
        MYSQL_STMT *prepared_stmt5;

        MYSQL_BIND param1[1];
        MYSQL_BIND param2[9];
        MYSQL_BIND param3[3];
        MYSQL_BIND param4[2];
        MYSQL_BIND param5[3];

        int ret, rows_count;

        int numero, anno, idTurno, idTurno_ck, status;
        char giorno[16];
        char mese[16];
        int ora_i, minuto_i, secondo_i, ora_f, minuto_f, secondo_f;
        MYSQL_TIME ora_inizio;
        MYSQL_TIME ora_fine;
        char bibliotecario_cf[17];
        char motivo[51];

        // chiamo find_slots_to sub() per mostrare i turni attualmente scoperti
        if(!setup_prepared_stmt(&prepared_stmt, "call find_slots_to_sub(?)", conn)){
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize find slots statement\n", false);
        }

        // Prepare parameters
        memset(param1, 0, sizeof(param1));

        param1[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param1[0].buffer = username_g;
        param1[0].buffer_length = strlen(username_g);

        if(mysql_stmt_bind_param(prepared_stmt, param1) != 0){
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for request inseriton\n", true);
        }

        // Run procedure
        if(mysql_stmt_execute(prepared_stmt) != 0){
                print_stmt_error(prepared_stmt, "An error occurred while adding the request");
```

```
                if(strcmp("45001", mysql_stmt_sqlstate(prepared_stmt)) == 0){
                        printf("\n---> There are no slots to cover");
                        mysql_stmt_close(prepared_stmt);
                        return;
                }
        }

        do{
                if(conn->server_status & SERVER_PS_OUT_PARAMS){
                        goto next;
                }

                dump_result_set(conn, prepared_stmt, "**** Uncovered slots ****");

next:
                status = mysql_stmt_next_result(prepared_stmt);
                if(status > 0){
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
                }
        } while (status == 0);

        /* Get total rows in the query */


        mysql_stmt_close(prepared_stmt);

        // inserisco le informazioni del turno che vogliamo andare a coprire
rescan_all:

        memset(&ora_inizio, 0, sizeof(MYSQL_TIME));
        memset(&ora_fine, 0, sizeof(MYSQL_TIME));

        printf("\nEnter the information of the slot to recover:\n");

rescan_hi:
        printf("Enter the start hour: \n");
        ret = scanf("%d", &ora_i);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_hi;
        }

rescan_mi:
        printf("Enter the minutes: \n");
        ret = scanf("%d", &minuto_i);
        flush(stdin);
        if(ret == 0){
```

```
                printf("not valid input\n");
                goto rescan_mi;
        }

rescan_si:
        printf("Enter the seconds: \n");
        ret = scanf("%d", &secondo_i);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_si;
        }

        ora_inizio.hour = ora_i;
        ora_inizio.minute = minuto_i;
        ora_inizio.second = secondo_i;

rescan_hf:
        printf("Enter the end hour: \n");
        ret = scanf("%d", &ora_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_hf;
        }

rescan_mf:
        printf("Enter the minutes: \n");
        ret = scanf("%d", &minuto_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_mf;
        }

rescan_sf:
        printf("Enter the seconds: \n");
        ret = scanf("%d", &secondo_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_sf;
        }

        ora_fine.hour = ora_f;
        ora_fine.minute = minuto_f;
        ora_fine.second = secondo_f;
```

```
rescan2:
        printf("Enter the number of the day (1-29,30,31): \n");
        ret = scanf("%d", &numero);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan2;
        }

        printf("Enter the day (es: lunedi, martedi..)\n");
        memset(giorno, 0, 16);
        getInput(16, giorno, false);


   printf("Enter the month (es: gennaio, febbraio..)\n");
   memset(mese, 0, 16);
        getInput(16, mese, false);

rescan3:
        printf("Enter the year: \n");
        ret = scanf("%d", &anno);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan3;
        }

        printf("Enter the librarian CF to replace: \n");
        getInput(17, bibliotecario_cf, false);


        // verifico che tale turno effettivamente esista nella base di dati
        // chiamando check_inserted_slot()
        if(!setup_prepared_stmt(&prepared_stmt2, "call check_inserted_slot(?,?,?,?,?,?,?,?,?)", conn)){
                finish_with_stmt_error(conn, prepared_stmt2, "Unable to initialize check_inserted_slot statement\n",
false);
        }

        // Prepare parameters
        memset(param2, 0, sizeof(param2));

        param2[0].buffer_type = MYSQL_TYPE_TIME;
        param2[0].buffer = &ora_inizio;
        param2[0].buffer_length = sizeof(ora_inizio);

        param2[1].buffer_type = MYSQL_TYPE_TIME;
        param2[1].buffer = &ora_fine;
        param2[1].buffer_length = sizeof(ora_fine);
```

```
param2[2].buffer_type = MYSQL_TYPE_LONG;
param2[2].buffer = &numero;
param2[2].buffer_length = sizeof(numero);

param2[3].buffer_type = MYSQL_TYPE_VAR_STRING;
param2[3].buffer = giorno;
param2[3].buffer_length = strlen(giorno);

param2[4].buffer_type = MYSQL_TYPE_VAR_STRING;
param2[4].buffer = mese;
param2[4].buffer_length = strlen(mese);

param2[5].buffer_type = MYSQL_TYPE_LONG;
param2[5].buffer = &anno;
param2[5].buffer_length = sizeof(anno);

param2[6].buffer_type = MYSQL_TYPE_VAR_STRING;
param2[6].buffer = bibliotecario_cf;
param2[6].buffer_length = strlen(bibliotecario_cf);

param2[7].buffer_type = MYSQL_TYPE_VAR_STRING;
param2[7].buffer = username_g;
param2[7].buffer_length = strlen(username_g);

param2[8].buffer_type = MYSQL_TYPE_LONG;
param2[8].buffer = &idTurno_ck;
param2[8].buffer_length = sizeof(idTurno_ck);

if(mysql_stmt_bind_param(prepared_stmt2, param2) != 0){
        finish_with_stmt_error(conn, prepared_stmt2, "Could not bind parameters for request inseriton\n", true);
}

// Run procedure
if(mysql_stmt_execute(prepared_stmt2) != 0){
        print_stmt_error(prepared_stmt2, "An error occurred while adding the request");
        if(strcmp("45007", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
                printf("\n Inserted slot not found.. please reinser the information");
                mysql_stmt_close(prepared_stmt2);
                goto rescan_all;
        }
}


if (mysql_stmt_store_result(prepared_stmt2)) {
        fprintf(stderr, " mysql_stmt_execute(), 12 failed\n");
        fprintf(stderr, " %s\n", mysql_stmt_error(prepared_stmt2));
}
```

```
	/* Get total rows in the query */
	rows_count = mysql_stmt_num_rows(prepared_stmt2);
	fprintf(stdout, " total rows in SELECT statement: %d\n", rows_count);

	if (rows_count < 1) /* validate column count */
	{
		fprintf(stderr, " invalid slot information inserted.. please retry...\n");
		goto rescan_all;
	}
	else{

		// inserisco le informazioni per ricoprire il turno scelto
		printf("\n Inserted slot OK.. ");

		memset(param2, 0, sizeof(param2));

		param2[0].buffer_type = MYSQL_TYPE_LONG;
		param2[0].buffer = &idTurno;
		param2[0].buffer_length = sizeof(idTurno);

		if(mysql_stmt_bind_result(prepared_stmt2, param2)){
			finish_with_stmt_error(conn, prepared_stmt2, "Could not retrieve output parameter", true);
		}

		// Retrieve output parameter
		if(mysql_stmt_fetch(prepared_stmt2)){
			finish_with_stmt_error(conn, prepared_stmt2, "Could not buffer the results", true);
		}

		printf("\n selected slot ID: %d", idTurno);

		do{
			if(conn->server_status & SERVER_PS_OUT_PARAMS){
				goto next1;
			}

next1:
			status = mysql_stmt_next_result(prepared_stmt2);
			if(status > 0){
				finish_with_stmt_error(conn, prepared_stmt2, "Unexpected condition", true);
			}
		} while (status == 0);

		mysql_stmt_close(prepared_stmt2);

		printf("\n Enter the reason of the absence: ");
		memset(motivo, 0, 51);
```

```
		getInput(51, motivo, false);

		// mostriamo i bibliotecari disponibili a ricoprire tale turno

		if(!setup_prepared_stmt(&prepared_stmt3, "call show_free_librarians(?,?,?)", conn)){
			finish_with_stmt_error(conn, prepared_stmt3, "Unable to initialize show_free_librarians
statement\n", false);
		}

		// Prepare parameters
		memset(param3, 0, sizeof(param3));

		param3[0].buffer_type = MYSQL_TYPE_VAR_STRING;
		param3[0].buffer = username_g;
		param3[0].buffer_length = strlen(username_g);

		param3[1].buffer_type = MYSQL_TYPE_VAR_STRING;
		param3[1].buffer = bibliotecario_cf;
		param3[1].buffer_length = strlen(bibliotecario_cf);

		param3[2].buffer_type = MYSQL_TYPE_LONG;
		param3[2].buffer = &idTurno;
		param3[2].buffer_length = sizeof(idTurno);


		if(mysql_stmt_bind_param(prepared_stmt3, param3) != 0){
			finish_with_stmt_error(conn, prepared_stmt3, "Could not bind parameters for request
inseriton\n", true);
		}

		// Run procedure
		if(mysql_stmt_execute(prepared_stmt3) != 0){
			print_stmt_error(prepared_stmt3, "An error occurred while adding the request");
		}

		do{
			if(conn->server_status & SERVER_PS_OUT_PARAMS){
				goto next2;
			}

			dump_result_set(conn, prepared_stmt3, "**** Free librarians ****");

next2:
			status = mysql_stmt_next_result(prepared_stmt3);
			if(status > 0){
				finish_with_stmt_error(conn, prepared_stmt3, "Unexpected condition", true);
			}
		} while (status == 0);
```

```
                mysql_stmt_close(prepared_stmt3);


rechoice:

                printf("\nEnter the librarian CF to select as substitute: \n");
                getInput(17, bibliotecario_cf, false);

                if(!setup_prepared_stmt(&prepared_stmt4, "call check_inserted_librarian_cf(?,?)", conn)){
                        finish_with_stmt_error(conn, prepared_stmt4, "Unable to initialize check_inserted_librarian_cf
statement\n", false);
                }

                // Prepare parameters
                memset(param4, 0, sizeof(param4));

                param4[0].buffer_type = MYSQL_TYPE_VAR_STRING;
                param4[0].buffer = bibliotecario_cf;
                param4[0].buffer_length = strlen(bibliotecario_cf);

                param4[1].buffer_type = MYSQL_TYPE_VAR_STRING;
                param4[1].buffer = username_g;
                param4[1].buffer_length = strlen(username_g);


                if(mysql_stmt_bind_param(prepared_stmt4, param4) != 0){
                        finish_with_stmt_error(conn, prepared_stmt4, "Could not bind parameters for request
inseriton\n", true);
                }

                // Run procedure
                if(mysql_stmt_execute(prepared_stmt4) != 0){
                        if(strcmp("45008", mysql_stmt_sqlstate(prepared_stmt4)) == 0){
                                printf("\n Inserted librarian CF not found.. please retry");
                                mysql_stmt_close(prepared_stmt4);
                                goto rechoice;
                        }
                }

                if (mysql_stmt_store_result(prepared_stmt4)) {
                        fprintf(stderr, " mysql_stmt_execute(), 11 failed\n");
                        fprintf(stderr, " %s\n", mysql_stmt_error(prepared_stmt4));
                        exit(0);
                }

                do{
```

```
                        if(conn->server_status & SERVER_PS_OUT_PARAMS){
                                goto next3;
                        }

next3:
                        status = mysql_stmt_next_result(prepared_stmt4);
                        if(status > 0){
                                finish_with_stmt_error(conn, prepared_stmt4, "Unexpected condition", true);
                        }
                } while (status == 0);


                printf("\n Inserted cf OK.. ");

                mysql_stmt_close(prepared_stmt4);

                if(!setup_prepared_stmt(&prepared_stmt5, "call add_librarian_sub(?,?,?)", conn)){
                        finish_with_stmt_error(conn, prepared_stmt5, "Unable to initialize add_librarian_sub
statement\n", false);
                }

                printf("\nINFO: %s, %d, %s", motivo, idTurno, bibliotecario_cf);
                // Prepare parameters
                memset(param5, 0, sizeof(param5));

                param5[0].buffer_type = MYSQL_TYPE_VAR_STRING;
                param5[0].buffer = motivo;
                param5[0].buffer_length = strlen(motivo);

                param5[1].buffer_type = MYSQL_TYPE_LONG;
                param5[1].buffer = &idTurno;
                param5[1].buffer_length = sizeof(idTurno);

                param5[2].buffer_type = MYSQL_TYPE_VAR_STRING;
                param5[2].buffer = bibliotecario_cf;
                param5[2].buffer_length = strlen(bibliotecario_cf);

                if(mysql_stmt_bind_param(prepared_stmt5, param5) != 0){
                        finish_with_stmt_error(conn, prepared_stmt5, "Could not bind parameters for user inseriton\n",
true);
                }

                // Run procedure
                if(mysql_stmt_execute(prepared_stmt5) != 0){
                        print_stmt_error(prepared_stmt5, "An error occurred while adding the user");
                }

                mysql_stmt_close(prepared_stmt5);
```

```
                printf("\nLibrarian correctly replaced!");

        }

}

static void show_dismissed_books(MYSQL *conn){

        MYSQL_STMT *prepared_stmt;
        int status;

        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call show_dismissed_books()", conn)){
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initializa dismiss_book statement\n", false);
        }

        // Run procedure
        if(mysql_stmt_execute(prepared_stmt) != 0){
                print_stmt_error(prepared_stmt, "An error occurred while retrieving the dismiss_book\n");
                if(strcmp("45002", mysql_stmt_sqlstate(prepared_stmt)) == 0){
                        printf("\n---> There are no dismissed books");
                        goto out;
                }
        }

        do{
                if(conn->server_status & SERVER_PS_OUT_PARAMS){
                        goto next;
                }

                dump_result_set(conn, prepared_stmt, "**** Dismissed books ****");

next:
                status = mysql_stmt_next_result(prepared_stmt);
                if(status > 0){
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
                }

        } while (status == 0);

out:
        mysql_stmt_close(prepared_stmt);

}

static void dismiss_book(MYSQL *conn){
```

```
MYSQL_STMT *prepared_stmt;
MYSQL_BIND param[1];
int counter;

// Prepare stored procedure call
if(!setup_prepared_stmt(&prepared_stmt, "call dismiss_book(?)", conn)){
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initializa dismiss_book statement\n", false);
}

// Prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &counter;
param[0].buffer_length = sizeof(counter);


if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for request inseriton\n", true);
}


// Run procedure
if(mysql_stmt_execute(prepared_stmt) != 0){
        print_stmt_error(prepared_stmt, "An error occurred while retrieving the dismiss_book\n");
        goto out;
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &counter;
param[0].buffer_length = sizeof(counter);

if(mysql_stmt_bind_result(prepared_stmt, param)){
        finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter", true);
}

// Retrieve output parameter
if(mysql_stmt_fetch(prepared_stmt)){
        finish_with_stmt_error(conn, prepared_stmt, "Could not buffer the results", true);
}

if(counter > 0){
        printf("\n One or more books have been dismissed");
}else{
        printf("\n There are no books to dismiss");
}
```

```
out:
	mysql_stmt_close(prepared_stmt);

	////////////////////////////////////////////////////////////////////////////////

}


static void report_slots(MYSQL *conn){

	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[1];

	int status;

	// Prepare the stored procedure call
	if(!setup_prepared_stmt(&prepared_stmt, "call free_slot_report(?)", conn)){
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize free_slot_report statement\n", false);
	}

	// prepare parameters
	memset(param, 0, sizeof(param));
	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = username_g;
	param[0].buffer_length = strlen(username_g);

	if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameter for turni scoperti report\n", true);
	}

	// run procedure
	if(mysql_stmt_execute(prepared_stmt) != 0){
		print_stmt_error(prepared_stmt, "An error occurred while retrieving the turni scoperti report");
		return;
	}

	do{
		if(conn->server_status & SERVER_PS_OUT_PARAMS){
			goto next;
		}

		dump_result_set(conn, prepared_stmt, "Uncovered slots");

next:
		status = mysql_stmt_next_result(prepared_stmt);
		if(status > 0){
			finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
```

```
                }
        } while (status == 0);

        mysql_stmt_close(prepared_stmt);


}



static void create_work_slot(MYSQL *conn){

        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[8];
        MYSQL_BIND param1[1];

        char giorno[16], mese[16];
        char bibliotecario_cf[17];
        int numero, anno, idTurno;
        int ret, status;
        MYSQL_TIME ora_inizio, ora_fine;
        int ora_i, ora_f, minuto_i, minuto_f, secondo_i, secondo_f;

        // Prepare the stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call show_all_librarians(?)", conn)){
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show_all_librarians statement\n",
false);
        }

        // prepare parameters
        memset(param1, 0, sizeof(param1));
        param1[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param1[0].buffer = username_g;
        param1[0].buffer_length = strlen(username_g);

        if(mysql_stmt_bind_param(prepared_stmt, param1) != 0){
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameter for turni scoperti report\n", true);
        }

        // run procedure
        if(mysql_stmt_execute(prepared_stmt) != 0){
                print_stmt_error(prepared_stmt, "An error occurred while show_all_librarians");
                return;
        }

        do{
                if(conn->server_status & SERVER_PS_OUT_PARAMS){
                        goto next;
```

```
                }

                dump_result_set(conn, prepared_stmt, "All employed librarians");

next:
                status = mysql_stmt_next_result(prepared_stmt);
                if(status > 0){
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
                }
        } while (status == 0);



        mysql_stmt_close(prepared_stmt);

retry_add_slot:

        printf("\nEnter the information of the slot to create.. please press enter to start...");
        flush(stdin);

        memset(&ora_inizio, 0, sizeof(MYSQL_TIME));
        memset(&ora_fine, 0, sizeof(MYSQL_TIME));

rescan_hi2:
        printf("Enter the start hour: \n");
        ret = scanf("%d", &ora_i);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_hi2;
        }

rescan_mi2:
        printf("Enter the minutes: \n");
        ret = scanf("%d", &minuto_i);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_mi2;
        }

rescan_si2:
        printf("Enter the seconds: \n");
        ret = scanf("%d", &secondo_i);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_si2;
```

```
        }

        ora_inizio.hour = ora_i;
        ora_inizio.minute = minuto_i;
        ora_inizio.second = secondo_i;

rescan_hf2:
        printf("Enter the end hour: \n");
        ret = scanf("%d", &ora_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_hf2;
        }

rescan_mf2:
        printf("Enter the minutes: \n");
        ret = scanf("%d", &minuto_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_mf2;
        }

rescan_sf2:
        printf("Enter the seconds: \n");
        ret = scanf("%d", &secondo_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_sf2;
        }

        ora_fine.hour = ora_f;
        ora_fine.minute = minuto_f;
        ora_fine.second = secondo_f;

rescan1:
        printf("\nEnter the number of the day: ");
        ret = scanf("%d", &numero);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan1;
        }

        printf("\nDay (lunedi, martedi..): ");
        getInput(16, giorno, false);
```

```
	printf("\nMonth (gennaio, febbraio..): ");
	getInput(16, mese, false);

rescan2:
	printf("\nYear: ");
	ret = scanf("%d", &anno);
	flush(stdin);
	if(ret == 0){
		printf("not valid input\n");
		goto rescan2;
	}

	printf("\nEnter the librarian CF to assign the slot: ");
	getInput(17, bibliotecario_cf, false);


	// Prepare prepared statement
	if(!setup_prepared_stmt(&prepared_stmt, "call add_slot(?,?,?,?,?,?,?,?)", conn)){
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add_slot statement\n", false);
	}

	// Prepare parameters
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_TIME;
	param[0].buffer = &ora_inizio;
	param[0].buffer_length = sizeof(ora_inizio);

	param[1].buffer_type = MYSQL_TYPE_TIME;
	param[1].buffer = &ora_fine;
	param[1].buffer_length = sizeof(ora_fine);

	param[2].buffer_type = MYSQL_TYPE_LONG;
	param[2].buffer = &numero;
	param[2].buffer_length = sizeof(numero);

	param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[3].buffer = giorno;
	param[3].buffer_length = strlen(giorno);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = mese;
	param[4].buffer_length = strlen(mese);

	param[5].buffer_type = MYSQL_TYPE_LONG;
	param[5].buffer = &anno;
	param[5].buffer_length = sizeof(anno);
```

```
param[6].buffer_type = MYSQL_TYPE_VAR_STRING;
param[6].buffer = bibliotecario_cf;
param[6].buffer_length = strlen(bibliotecario_cf);

param[7].buffer_type = MYSQL_TYPE_LONG; // OUT
param[7].buffer = &idTurno;
param[7].buffer_length = sizeof(idTurno);

if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for user inseriton\n", true);
}

// Run procedure
if(mysql_stmt_execute(prepared_stmt) != 0){
        //print_stmt_error(prepared_stmt, "An error occurred while adding the slot");
        if(strcmp("23000", mysql_stmt_sqlstate(prepared_stmt)) == 0){
                printf("\n Error inserting some foreign key");
                mysql_stmt_close(prepared_stmt);
                goto retry_add_slot;
        }

        if(strcmp("45010", mysql_stmt_sqlstate(prepared_stmt)) == 0){
                printf("\n Duration of the slot greater than 8 hours");
                mysql_stmt_close(prepared_stmt);
                goto retry_add_slot;
        }

        if(strcmp("45012", mysql_stmt_sqlstate(prepared_stmt)) == 0){
                printf("\n Invalid inserted day");
                mysql_stmt_close(prepared_stmt);
                goto retry_add_slot;
        }

        if(strcmp("45013", mysql_stmt_sqlstate(prepared_stmt)) == 0){
                printf("\n Invalid inserted month");
                mysql_stmt_close(prepared_stmt);
                goto retry_add_slot;
        }
}

// Get back the ID of the newly_added user
memset(param, 0, sizeof(param));
param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &idTurno;
param[0].buffer_length = sizeof(idTurno);

if(mysql_stmt_bind_result(prepared_stmt, param)){
```

```
                finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter", true);
        }

        // Retrieve output parameter
        if(mysql_stmt_fetch(prepared_stmt)){
                finish_with_stmt_error(conn, prepared_stmt, "Could not buffer the results", true);

        }

        printf("\n **** Slot correctly created with ID: %d ****\n", idTurno);


        mysql_stmt_close(prepared_stmt);


}



void run_as_amministratore(MYSQL *conn, char username[46]){

        char options[6] = {'1', '2', '3', '4', '5', '6'};
        char op;

        printf("Switching to administrative role...\n");

        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);
        }

        strcpy(username_g, username);
        printf("--> %s", username_g);

        while(true){

                printf("\033[2J\033[H");
                printf("*** What should I do for you? ***\n\n");
                printf("1) Replace not available librarian\n");
                printf("2) Dismiss book\n");
                printf("3) Show dismissed books\n");
                printf("4) Work slot report\n");
```

```
                printf("5) Create new work slot\n");
                printf("6) Quit\n");

                op = multiChoice("Select an option", options, 6);

                switch(op){

                        case '1':

                                find_slots_sub(conn);
                                break;

                        case '2':

                                dismiss_book(conn);
                                break;

                        case '3':

                                show_dismissed_books(conn);
                                break;

                        case '4':

                                report_slots(conn);
                                break;

                        case '5':

                                create_work_slot(conn);
                                break;

                        case '6':
                                return;


                        default:
                                fprintf(stderr, "Invalid condition at %s: %d\n", __FILE__, __LINE__);
                                abort();
                }

                printf("\npremere invio per continuare..\n");
                getchar();
        }
}
```

//

bibliotecario.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <mysql.h>
#include "defines.h"

#define flush(stdin) while(getchar() != '\n')



char idUtente_gl[17];
int idCopia_gl, Terminato_gl, Periodo_consultazione_gl;
char username_g[46];


static void disease_request(MYSQL *conn){

	MYSQL_STMT *prepared_stmt;
	MYSQL_STMT *prepared_stmt1;

	MYSQL_BIND param[8];
	MYSQL_BIND param1[1];
	int rows_count, status;
	int numero, anno, ret, ret_val = 0;
	int ora_i, ora_f, minuto_i, minuto_f, secondo_i, secondo_f;
	char giorno[16], mese[16];
	MYSQL_TIME ora_inizio;
	MYSQL_TIME ora_fine;

	// chiamo find_slots_to sub() per mostrare i turni attualmente scoperti
	if(!setup_prepared_stmt(&prepared_stmt1, "call show_librarian_slots(?)", conn)){
		finish_with_stmt_error(conn, prepared_stmt1, "Unable to initialize show_librarian_slots statement\n",
false);
	}

	// Prepare parameters
	memset(param1, 0, sizeof(param1));

	param1[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param1[0].buffer = username_g;
	param1[0].buffer_length = strlen(username_g);

	if(mysql_stmt_bind_param(prepared_stmt1, param1) != 0){
		finish_with_stmt_error(conn, prepared_stmt1, "Could not bind parameters for request inseriton\n", true);
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt1) != 0){
		print_stmt_error(prepared_stmt1, "An error occurred while adding the request");
		if(strcmp("45004", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n---> There are no slots for this librarian");
			return;
		}
	}
```

```
	do{
		if(conn->server_status & SERVER_PS_OUT_PARAMS){
			goto next;
		}

		dump_result_set(conn, prepared_stmt1, "**** Librarian slots ****");

next:
		status = mysql_stmt_next_result(prepared_stmt1);
		if(status > 0){
			finish_with_stmt_error(conn, prepared_stmt1, "Unexpected condition", true);
		}
	} while (status == 0);


	/* Get total rows in the query */
	rows_count = mysql_stmt_num_rows(prepared_stmt1);
	fprintf(stdout, " total rows in SELECT statement: %d\n", rows_count);

	mysql_stmt_close(prepared_stmt1);



rescan_all1:

	memset(&ora_inizio, 0, sizeof(MYSQL_TIME));
	memset(&ora_fine, 0, sizeof(MYSQL_TIME));

	printf("Enter the information of the slot to recover:\n");

rescan_hi:
	printf("Enter the start hour: \n");
	ret = scanf("%d", &ora_i);
	flush(stdin);
	if(ret == 0){
		printf("not valid input\n");
		goto rescan_hi;
	}

rescan_mi:
	printf("Enter the minutes: \n");
	ret = scanf("%d", &minuto_i);
	flush(stdin);
	if(ret == 0){
		printf("not valid input\n");
		goto rescan_mi;
	}

rescan_si:
	printf("Enter the seconds: \n");
	ret = scanf("%d", &secondo_i);
	flush(stdin);
	if(ret == 0){
		printf("not valid input\n");
		goto rescan_si;
	}

	ora_inizio.hour = ora_i;
	ora_inizio.minute = minuto_i;
```

```
        ora_inizio.second = secondo_i;

rescan_hf:
        printf("Enter the end hour: \n");
        ret = scanf("%d", &ora_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_hf;
        }

rescan_mf:
        printf("Enter the minutes: \n");
        ret = scanf("%d", &minuto_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_mf;
        }

rescan_sf:
        printf("Enter the seconds: \n");
        ret = scanf("%d", &secondo_f);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan_sf;
        }

        ora_fine.hour = ora_f;
        ora_fine.minute = minuto_f;
        ora_fine.second = secondo_f;

rescan1:
        printf("Enter the number of the day: \n");
        ret = scanf("%d", &numero);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan1;
        }

        printf("Enter the day (es: lunedi, martedi..)\n");
        memset(giorno, 0, 16);
        getInput(16, giorno, false);


   printf("Enter the month (es: gennaio, febbraio..)\n");
   memset(mese, 0, 16);
   getInput(16, mese, false);



rescan2:
        printf("Enter the year: \n");
        ret = scanf("%d", &anno);
        flush(stdin);
        if(ret == 0){
                printf("not valid input\n");
                goto rescan2;
```

```
}

// Prepare prepared statement
if(!setup_prepared_stmt(&prepared_stmt, "call add_disease_request(?,?,?,?,?,?,?,?)", conn)){
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add_disease_request statement\n", false);
}

// Prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = username_g;
param[0].buffer_length = strlen(username_g);

param[1].buffer_type = MYSQL_TYPE_TIME;
param[1].buffer = &ora_inizio;
param[1].buffer_length = sizeof(ora_inizio);

param[2].buffer_type = MYSQL_TYPE_TIME;
param[2].buffer = &ora_fine;
param[2].buffer_length = sizeof(ora_fine);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &numero;
param[3].buffer_length = sizeof(numero);

param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
param[4].buffer = giorno;
param[4].buffer_length = strlen(giorno);

param[5].buffer_type = MYSQL_TYPE_VAR_STRING;
param[5].buffer = mese;
param[5].buffer_length = strlen(mese);

param[6].buffer_type = MYSQL_TYPE_LONG;
param[6].buffer = &anno;
param[6].buffer_length = sizeof(anno);

param[7].buffer_type = MYSQL_TYPE_LONG; // OUT
param[7].buffer = &ret_val;
param[7].buffer_length = sizeof(ret_val);

if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for request inseriton\n", true);
        goto out;
}

// Run procedure
if(mysql_stmt_execute(prepared_stmt) != 0){
        print_stmt_error(prepared_stmt, "An error occurred while adding the request");
        goto out;
}

// Get back the ID of the newly_added user
memset(param, 0, sizeof(param));
param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &ret_val;
param[0].buffer_length = sizeof(ret_val);

if(mysql_stmt_bind_result(prepared_stmt, param)){
```

```
                finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter", true);
        }

        // Retrieve output parameter
        if(mysql_stmt_fetch(prepared_stmt)){
                finish_with_stmt_error(conn, prepared_stmt, "Could not buffer the results", true);
        }

        do{
                if(conn->server_status & SERVER_PS_OUT_PARAMS){
                        goto next1;
                }

next1:
                status = mysql_stmt_next_result(prepared_stmt);
                if(status > 0){
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
                }
        } while (status == 0);



        if(ret_val == 0){
                system("clear");
                printf("Slot not found.. please reinsert the slot information...\n");
                goto rescan_all1;
        }

        printf("**** Disease request correctly sent ****\n");

out:
        mysql_stmt_close(prepared_stmt);

}


static void add_new_user(MYSQL *conn){

        MYSQL_STMT *prepared_stmt1;
        MYSQL_BIND param1[13];

        // Input for the registration in the table Utilizzatori sistema
        char username[46];
        char password[33];
        char ruolo[7];

        // Input for the registration in the table Utenti
        char CF[17];
        char name[31];
        char surname[31];
        MYSQL_TIME date_birth;
        int n_anno, n_mese, n_giorno;
        char gender[6];

        // Input for the registration in the table Recapiti
        char telefono[11];
   char cellulare[11];
   char email[46];
   int pref_recap;
   int ret;
```

```
retry_add_user:

        printf("\nEnter the user information:\n");
        printf("press enter to start\n");
        flush(stdin);

        // Get the required information
        printf("\nUser username: ");
        getInput(46, username, false);

        printf("\nUser password: ");
        getInput(33, password, true);

        strcpy(ruolo, "utente");

        printf("\nUser CF: ");
        getInput(17, CF, false);
        printf("\nName: ");
        getInput(31, name, false);
        printf("\nSurname: ");
        getInput(31, surname, false);

        memset(&date_birth, 0, sizeof(MYSQL_TIME));

retry_ngiorno_u:
        printf("\nEnter the birth date:\n Insert the day (1-29,30,31): ");
        ret = scanf("%d", &n_giorno);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry_ngiorno_u;
        }

retry_nmese_u:
        printf("\n Enter the month (1-12): ");
        ret = scanf("%d", &n_mese);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry_nmese_u;
        }

retry_anno_u:
        printf("\n Enter the year: ");
        ret = scanf("%d", &n_anno);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry_anno_u;
        }

        date_birth.day = n_giorno;
        date_birth.month = n_mese;
        date_birth.year = n_anno;

        flush(stdin);

        printf("\nGender (format M or F or other): ");
        getInput(6, gender, false);
```

```
	printf("\nEnter at least one contact between phone, mobile phone and email: (insert null if you don't want to specify some of these contacts)\n");

rescan_contacts:
	printf("\nUser phone: ");
	memset(telefono, 0, 11);
	getInput(11, telefono, false);
	printf("\nUser mobile phone: ");
	memset(cellulare, 0, 11);
	getInput(11, cellulare, false);
	printf("\nUser email: ");
	memset(email, 0, 46);
	getInput(46, email, false);

	if((strcmp(telefono, "null") == 0) || (strcmp(telefono, "NULL") == 0) || (strcmp(telefono, "") == 0)){
		if((strcmp(cellulare, "null") == 0) || (strcmp(cellulare, "NULL") == 0) || (strcmp(cellulare, "") == 0)){
			if((strcmp(email, "null") == 0) || (strcmp(email, "NULL") == 0) || (strcmp(email, "") == 0)){
				printf("Please enter at least one of these contacts..\n");
				goto rescan_contacts;
			}
		}
	}

	// Selezione del recapito preferito

rechoice:
	printf("Specify one of the entered contacts as your favorite: \n");
	printf("Enter: \n 1) if you want your PHONE as your favorite contact\n 2) if you want your MOBILE PHONE as your favorite contact\n 3) if you want your EMAIL as your favorite contact\n");

	ret = scanf("%d", &pref_recap);
	if(ret == 0){
		printf("not valid input\n");
		goto rechoice;
	}

	if(pref_recap != 1 && pref_recap != 2 && pref_recap != 3){
		printf("not valid input\n");
		goto rechoice;
	}

	if(pref_recap == 1){
		if((strcmp(telefono, "null") == 0) || (strcmp(telefono, "NULL") == 0) || (strcmp(telefono, "") == 0)){
			printf("selected contact not valid\n");
			goto rechoice;
		}
	}else{
		if(pref_recap == 2){
			if((strcmp(cellulare, "null") == 0) || (strcmp(cellulare, "NULL") == 0) || (strcmp(cellulare, "") == 0)){
				printf("selected contact not valid\n");
				goto rechoice;
			}
		}else{
			if(pref_recap == 3){
				if((strcmp(email, "null") == 0) || (strcmp(email, "NULL") == 0) || (strcmp(email, "") == 0)){
					printf("selected contact not valid\n");
					goto rechoice;
```

```
                        }
                    }
                }
            }

// Prepare prepared statement
if(!setup_prepared_stmt(&prepared_stmt1, "call add_user(?,?,?,?,?,?,?,?,?,?,?,?,?)", conn)){
        finish_with_stmt_error(conn, prepared_stmt1, "Unable to initialize add_user statement\n", false);
}

// Prepare parameters
memset(param1, 0, sizeof(param1));

param1[0].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[0].buffer = username;
param1[0].buffer_length = strlen(username);

param1[1].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[1].buffer = password;
param1[1].buffer_length = strlen(password);

param1[2].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[2].buffer = ruolo;
param1[2].buffer_length = strlen(ruolo);

param1[3].buffer_type = MYSQL_TYPE_VAR_STRING;
param1[3].buffer = CF;
param1[3].buffer_length = strlen(CF);

param1[4].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[4].buffer = name;
param1[4].buffer_length = strlen(name);

param1[5].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[5].buffer = surname;
param1[5].buffer_length = strlen(surname);

param1[6].buffer_type = MYSQL_TYPE_DATE; // IN
param1[6].buffer = &date_birth;
param1[6].buffer_length = sizeof(date_birth);

param1[7].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[7].buffer = gender;
param1[7].buffer_length = strlen(gender);

param1[8].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[8].buffer = telefono;
param1[8].buffer_length = strlen(telefono);

param1[9].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[9].buffer = cellulare;
param1[9].buffer_length = strlen(cellulare);

param1[10].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param1[10].buffer = email;
param1[10].buffer_length = strlen(email);

param1[11].buffer_type = MYSQL_TYPE_LONG; // IN
param1[11].buffer = &pref_recap;
param1[11].buffer_length = sizeof(pref_recap);
```

```
	param1[12].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
	param1[12].buffer = username_g;
	param1[12].buffer_length = strlen(username_g);


	if(mysql_stmt_bind_param(prepared_stmt1, param1) != 0){
		finish_with_stmt_error(conn, prepared_stmt1, "Could not bind parameters for user inseriton\n", true);
		goto out;
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt1) != 0){
		print_stmt_error(prepared_stmt1, "An error occurred while adding the user");
		if(strcmp("22007", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n Invalid date of birth.. please retry");
			mysql_stmt_close(prepared_stmt1);
			goto retry_add_user;
		}

		if(strcmp("45014", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n Gender not valid.. please retry");
			mysql_stmt_close(prepared_stmt1);
			goto retry_add_user;
		}

		if(strcmp("45015", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n Email not valid.. please retry");
			mysql_stmt_close(prepared_stmt1);
			goto retry_add_user;
		}

		if(strcmp("23000", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n Invalid foreign key.. please retry");
			mysql_stmt_close(prepared_stmt1);
			goto retry_add_user;
		}
	}


	printf("\n Inserted user'CF: %s", CF);
	memset(idUtente_gl, 0, sizeof(idUtente_gl));
	strcpy(idUtente_gl, CF);

	printf("\n**** User and contacts correctly inserted in the library system****\n");
out:
	mysql_stmt_close(prepared_stmt1);

}


static void add_new_prestito(MYSQL *conn){


	MYSQL_STMT *prepared_stmt2;
	MYSQL_BIND param2[5];
	int idPrestito, ret;

retry_add_loan:
```

```
	// Prepare prepared statement
	if(!setup_prepared_stmt(&prepared_stmt2, "call add_loan(?,?,?,?,?)", conn)){
		finish_with_stmt_error(conn, prepared_stmt2, "Unable to initialize add_loan statement\n", false);
	}

	// Prepare parameters
	memset(param2, 0, sizeof(param2));

	param2[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param2[0].buffer = idUtente_gl;
	param2[0].buffer_length = strlen(idUtente_gl);

	param2[1].buffer_type = MYSQL_TYPE_LONG;
	param2[1].buffer = &idCopia_gl;
	param2[1].buffer_length = sizeof(idCopia_gl);

	param2[2].buffer_type = MYSQL_TYPE_LONG;
	param2[2].buffer = &Terminato_gl;
	param2[2].buffer_length = sizeof(Terminato_gl);

	param2[3].buffer_type = MYSQL_TYPE_LONG;
	param2[3].buffer = &Periodo_consultazione_gl;
	param2[3].buffer_length = sizeof(Periodo_consultazione_gl);

	param2[4].buffer_type = MYSQL_TYPE_LONG; // OUT
	param2[4].buffer = &idPrestito;
	param2[4].buffer_length = sizeof(idPrestito);


	if(mysql_stmt_bind_param(prepared_stmt2, param2) != 0){
		finish_with_stmt_error(conn, prepared_stmt2, "Could not bind parameters for loan inseriton\n", true);
		goto out;
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt2) != 0){
		print_stmt_error(prepared_stmt2, "An error occurred while adding the loan");
		if(strcmp("45011", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
			printf("\n---> Invalid period of consultation");
			printf("\nHow long does the user want to consult the book? (1, 2 o 3 month max)\n");

rescan11:
			printf("Consultation period: \n");
			ret = scanf("%d", &Periodo_consultazione_gl);
			flush(stdin);
			if(ret == 0){
				printf("input non valido\n");
				goto rescan11;
			}

			mysql_stmt_close(prepared_stmt2);
			goto retry_add_loan;
		}

		if(strcmp("23000", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
			printf("\n---> Invalid foreign key selected");
			mysql_stmt_close(prepared_stmt2);
			return;
		}
```

```
	}

	// Get back the ID of the newly_added loan
	memset(param2, 0, sizeof(param2));
	param2[0].buffer_type = MYSQL_TYPE_LONG;
	param2[0].buffer = &idPrestito;
	param2[0].buffer_length = sizeof(idPrestito);

	if(mysql_stmt_bind_result(prepared_stmt2, param2)){
		finish_with_stmt_error(conn, prepared_stmt2, "Could not retrieve output parameter", true);
	}

	// Retrieve output parameter
	if(mysql_stmt_fetch(prepared_stmt2)){
		finish_with_stmt_error(conn, prepared_stmt2, "Could not buffer the results", true);

	}

	printf("\n**** Loan correctly added with ID: %d ****\n", idPrestito);

out:
	mysql_stmt_close(prepared_stmt2);


}

static void disponibilita_copie(MYSQL *conn){

	MYSQL_STMT *prepared_stmt;
	MYSQL_STMT *prepared_stmt5;
	MYSQL_BIND param5[3];
	MYSQL_BIND param[3];

	MYSQL_STMT *prepared_stmt3;
	MYSQL_BIND param3[2];

	int var_ret, status;
	int ret, idCopia, idBibliotecaPartenza;
	int pConsultazione;
	char risp;
	char buffer[46];

	memset(buffer,0, 46);

retry_av_copy:

	printf("\nPlease enter the title of book: ");
	getInput(46, buffer, false);
	////////////////////////////////////////////////////////////////////////////////////////////

	// Prepare prepared statement
	if(!setup_prepared_stmt(&prepared_stmt, "call recover_available_book_copies(?,?,?)", conn)){
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize recover_available_book_copies
statement\n", false);
	}

	// Prepare parameters
	memset(param, 0, sizeof(param));
```

```
	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = username_g;
	param[0].buffer_length = strlen(username_g);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = buffer;
	param[1].buffer_length = strlen(buffer);

	param[2].buffer_type = MYSQL_TYPE_LONG; // OUT
	param[2].buffer = &var_ret;
	param[2].buffer_length = sizeof(var_ret);

	if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for recover_available_book_copies\n", true);
		goto out;
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt) != 0){
		print_stmt_error(prepared_stmt, "An error occurred while recover_available_book_copies");
		goto out;
	}

	// Get back the ID of the newly_added user
	memset(param, 0, sizeof(param));
	param[0].buffer_type = MYSQL_TYPE_LONG;
	param[0].buffer = &var_ret;
	param[0].buffer_length = sizeof(var_ret);

	if(mysql_stmt_bind_result(prepared_stmt, param)){
		finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter", true);
	}

	// Retrieve output parameter
	if(mysql_stmt_fetch(prepared_stmt)){
		finish_with_stmt_error(conn, prepared_stmt, "Could not buffer the results", true);

	}

out:
	mysql_stmt_close(prepared_stmt);

	////////////////////////////////////////////////////////////////////////////////////////////////

	printf("Available copies: %d\n", var_ret);
	if(var_ret == 0){

		printf("\nThere are no copies available for this book..\nVerifying availability in other library of the circuit...");
		fflush(stdout);

		// Prepare stored procedure call
		if(!setup_prepared_stmt(&prepared_stmt3, "call transferable_copies(?,?)", conn)){
			finish_with_stmt_error(conn, prepared_stmt3, "Unable to initializa transferable_copies statement\n", false);
		}

		// Prepare parameters
		memset(param3, 0, sizeof(param3));
```

```
		param3[0].buffer_type = MYSQL_TYPE_VAR_STRING;
		param3[0].buffer = buffer;
		param3[0].buffer_length = strlen(buffer);

		param3[1].buffer_type = MYSQL_TYPE_VAR_STRING;
		param3[1].buffer = username_g;
		param3[1].buffer_length = strlen(username_g);

		if(mysql_stmt_bind_param(prepared_stmt3, param3) != 0){
			finish_with_stmt_error(conn, prepared_stmt3, "Could not bind parameters for copie_trasferibili_proc\n", true);
		}

		// Run procedure
		if(mysql_stmt_execute(prepared_stmt3) != 0){
			print_stmt_error(prepared_stmt3, "An error occurred while retrieving the transferable_copies\n");
			if(strcmp("45005", mysql_stmt_sqlstate(prepared_stmt3)) == 0){
				printf("\n---> There are no copy of this book in the circuit");
				return;
			}
		}

		do{
			if(conn->server_status & SERVER_PS_OUT_PARAMS){
				goto next3;
			}

			dump_result_set(conn, prepared_stmt3, "**** Transferable copies ****");

next3:
			status = mysql_stmt_next_result(prepared_stmt3);
			if(status > 0){
				finish_with_stmt_error(conn, prepared_stmt3, "Unexpected condition", true);
			}
		} while (status == 0);

		/* Get total rows in the query */


		mysql_stmt_close(prepared_stmt3);


rescan6:
		printf("\n Enter the library ID from which to transfer the copy of the book: ");
		ret = scanf("%d", &idBibliotecaPartenza);
		flush(stdin);
		if(ret == 0){
			printf("not valid input\n");
			goto rescan6;
		}

rescan7:
		printf("\n Enter the copy ID to transfer: ");
		ret = scanf("%d", &idCopia);
		flush(stdin);
		if(ret == 0){
			printf("not valid input\n");
			goto rescan7;
		}
```

```
		MYSQL_STMT *prepared_stmt4;
		MYSQL_BIND param4[4];

		// Prepare prepared statement
		if(!setup_prepared_stmt(&prepared_stmt4, "call update_copies(?,?,?,?)", conn)){
			finish_with_stmt_error(conn, prepared_stmt4, "Unable to initialize update_copies statement\n",
false);
		}

		// Prepare parameters
		memset(param4, 0, sizeof(param4));

		param4[0].buffer_type = MYSQL_TYPE_LONG;
		param4[0].buffer = &idCopia;
		param4[0].buffer_length = sizeof(idCopia);

		param4[1].buffer_type = MYSQL_TYPE_LONG;
		param4[1].buffer = &idBibliotecaPartenza;
		param4[1].buffer_length = sizeof(idBibliotecaPartenza);

		param4[2].buffer_type = MYSQL_TYPE_VAR_STRING;
		param4[2].buffer = username_g;
		param4[2].buffer_length = strlen(username_g);

		param4[3].buffer_type = MYSQL_TYPE_VAR_STRING;
		param4[3].buffer = buffer;
		param4[3].buffer_length = strlen(buffer);

		if(mysql_stmt_bind_param(prepared_stmt4, param4) != 0){
			finish_with_stmt_error(conn, prepared_stmt4, "Could not bind parameters for transfer
inseriton\n", true);
		}

		// Run procedure
		if(mysql_stmt_execute(prepared_stmt4) != 0){
			if(strcmp("45019", mysql_stmt_sqlstate(prepared_stmt4)) == 0){
				printf("\n Invalid selected item..please retry");
				mysql_stmt_close(prepared_stmt4);
				goto retry_av_copy;
			}
		}


		printf("\n**** copy correctly transferred ****\n");

		mysql_stmt_close(prepared_stmt4);

	}else{

		// copie disponibili trovate
		MYSQL_STMT *prepared_stmt1;
		MYSQL_BIND param1[2];

		// Prepare stored procedure call
		if(!setup_prepared_stmt(&prepared_stmt1, "call show_available_copies(?,?)", conn)){
			finish_with_stmt_error(conn, prepared_stmt1, "Unable to initializa show_available_copies
statement\n", false);
		}
```

```
		// Prepare parameters
		memset(param1, 0, sizeof(param1));

		param1[0].buffer_type = MYSQL_TYPE_VAR_STRING;
		param1[0].buffer = username_g;
		param1[0].buffer_length = strlen(username_g);

		param1[1].buffer_type = MYSQL_TYPE_VAR_STRING;
		param1[1].buffer = buffer;
		param1[1].buffer_length = strlen(buffer);


		if(mysql_stmt_bind_param(prepared_stmt1, param1) != 0){
			finish_with_stmt_error(conn, prepared_stmt1, "Could not bind parameters for show_available_copies\n", true);
		}

		// Run procedure
		if(mysql_stmt_execute(prepared_stmt1) != 0){
			print_stmt_error(prepared_stmt1, "(1) An error occurred while retrieving the show_available_copies\n");
			return;
		}

		do{
			if(conn->server_status & SERVER_PS_OUT_PARAMS){
				goto next4;
			}

			dump_result_set(conn, prepared_stmt1, "**** Available copies ****");

next4:
			status = mysql_stmt_next_result(prepared_stmt1);
			if(status > 0){
				finish_with_stmt_error(conn, prepared_stmt1, "Unexpected condition", true);
			}
		} while (status == 0);

		mysql_stmt_close(prepared_stmt1);

rescan9:
		printf("Enter the copy ID to deliver to the user: \n");
		ret = scanf("%d", &idCopia);
		flush(stdin);
		if(ret == 0){
			printf("not valid input");
			goto rescan9;
		}

		////////////////////////////////////////////////////////////////////////////

		if(!setup_prepared_stmt(&prepared_stmt5, "call check_inserted_copy_id(?,?,?)", conn)){
			finish_with_stmt_error(conn, prepared_stmt5, "Unable to initialize check_inserted_copy_id statement\n", false);
		}

		// Prepare parameters
		memset(param5, 0, sizeof(param5));
```

```
		param5[0].buffer_type = MYSQL_TYPE_LONG;
		param5[0].buffer = &idCopia;
		param5[0].buffer_length = sizeof(idCopia);

		param5[1].buffer_type = MYSQL_TYPE_VAR_STRING;
		param5[1].buffer = buffer;
		param5[1].buffer_length = strlen(buffer);

		param5[2].buffer_type = MYSQL_TYPE_VAR_STRING;
		param5[2].buffer = username_g;
		param5[2].buffer_length = strlen(username_g);

		if(mysql_stmt_bind_param(prepared_stmt5, param5) != 0){
			finish_with_stmt_error(conn, prepared_stmt5, "Could not bind parameters for request inseriton\n",
true);
		}

		// Run procedure
		if(mysql_stmt_execute(prepared_stmt5) != 0){
			if(strcmp("45018", mysql_stmt_sqlstate(prepared_stmt5)) == 0){
				printf("\n Inserted copy ID not found.. please retry");
				mysql_stmt_close(prepared_stmt5);
				goto retry_av_copy;
			}
		}

		mysql_stmt_close(prepared_stmt5);
		////////////////////////////////////////////////////////////////////////////


		printf("Enter the user information for the creation of the loan\n");

rescan10:
		printf("Is the user already registered in our library? (y/n)\n");
		ret = scanf("%c", &risp);
		if(ret == 0){
			printf("not valid input\n");
			goto rescan10;
		}

		if(risp != 'y' && risp != 'n' && risp != 'Y' && risp != 'N'){
			printf("not valid input\n");
			goto rescan10;
		}else{
			if(risp == 'y' || risp == 'Y'){


				MYSQL_STMT *prepared_stmt2;
				MYSQL_BIND param2[1];
				char idUtente[17];

				// Prepare stored procedure call
				if(!setup_prepared_stmt(&prepared_stmt2, "call show_library_users(?)", conn)){
					finish_with_stmt_error(conn, prepared_stmt2, "(1) Unable to initializa
show_library_users statement\n", false);
				}

				printf("\n");
				fflush(stdout);
```

```
                        // Prepare parameters
                        memset(param2, 0, sizeof(param2));

                        param2[0].buffer_type = MYSQL_TYPE_VAR_STRING;
                        param2[0].buffer = username_g;
                        param2[0].buffer_length = strlen(username_g);


                        if(mysql_stmt_bind_param(prepared_stmt2, param2) != 0){
                                finish_with_stmt_error(conn, prepared_stmt2, "Could not bind parameters for
show_library_users\n", true);
                        }


                        // Run procedure
                        if(mysql_stmt_execute(prepared_stmt2) != 0){
                                print_stmt_error(prepared_stmt2, "(2) An error occurred while retrieving the
show_library_users\n");
                                if(strcmp("45006", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
                                        printf("\n---> There are no users in this library");
                                        mysql_stmt_close(prepared_stmt2);
                                        return;
                                }
                        }

                        do{
                                if(conn->server_status & SERVER_PS_OUT_PARAMS){
                                        goto next;
                                }

                                dump_result_set(conn, prepared_stmt2, "**** Library users ****");

next:
                                status = mysql_stmt_next_result(prepared_stmt2);
                                if(status > 0){
                                        finish_with_stmt_error(conn,        prepared_stmt2,        "Unexpected
condition", true);
                                }
                        } while (status == 0);

                        /* Get total rows in the query */

                        mysql_stmt_close(prepared_stmt2);

                        flush(stdin);
                        printf("\nEnter the user CF: ");
                        memset(idUtente, 0, 17);

                        getInput(17, idUtente, false);
                        strcpy(idUtente_gl, idUtente);

                }else{

                        // procedura di iscrizione dell'utente alla biblioteca
                        add_new_user(conn);


                }
```

```
                }


                idCopia_gl = idCopia;
                Terminato_gl = 0;
                printf("---> idUtente: %s\t idCopia: %d\n", idUtente_gl, idCopia_gl);


                printf("\nHow long does the user want to consult the book? (1, 2 o 3 month max)\n");

rescan8:
                printf("Consultation period: \n");
                ret = scanf("%d", &pConsultazione);
                flush(stdin);
                if(ret == 0){
                        printf("input non valido\n");
                        goto rescan8;
                }


                Periodo_consultazione_gl = pConsultazione;
                add_new_prestito(conn);
        }

}




static void report_libri(MYSQL *conn){

        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];
        int status;

        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call loans_report(?)", conn)){
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initializa loans_report statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = username_g;
        param[0].buffer_length = strlen(username_g);


        if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for report prestiti\n", true);
        }

        // Run procedure
        if(mysql_stmt_execute(prepared_stmt) != 0){
                print_stmt_error(prepared_stmt, "An error occurred while retrieving the report_prestiti\n");
                return;
        }

        do{
                if(conn->server_status & SERVER_PS_OUT_PARAMS){
```

```
                goto next2;
        }

        dump_result_set(conn, prepared_stmt, "**** Ongoing loans ****");

next2:
        status = mysql_stmt_next_result(prepared_stmt);
        if(status > 0){
                finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
        }
    } while (status == 0);


    mysql_stmt_close(prepared_stmt);


}


static void contatta_utente(MYSQL *conn){

    MYSQL_STMT *prepared_stmt;
    MYSQL_BIND param[1];
    int status;

    // Prepare stored procedure call
    if(!setup_prepared_stmt(&prepared_stmt, "call user_addresses(?)", conn)){
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initializa user_addresses statement\n", false);
    }

    // Prepare parameters
    memset(param, 0, sizeof(param));

    param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[0].buffer = username_g;
    param[0].buffer_length = strlen(username_g);


    if(mysql_stmt_bind_param(prepared_stmt, param) != 0){
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for report prestiti\n", true);
    }

    // Run procedure
    if(mysql_stmt_execute(prepared_stmt) != 0){
        print_stmt_error(prepared_stmt, "An error occurred while retrieving the report_prestiti\n");
        return;
    }

    do{
        if(conn->server_status & SERVER_PS_OUT_PARAMS){
                goto next2;
        }

        dump_result_set(conn, prepared_stmt, "**** User contacts  ****");

next2:
        status = mysql_stmt_next_result(prepared_stmt);
        if(status > 0){
                finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);
        }
```

```
	} while (status == 0);

	mysql_stmt_close(prepared_stmt);


}

static void restituzione_prestito(MYSQL *conn){

	MYSQL_STMT *prepared_stmt1;
	MYSQL_STMT *prepared_stmt2;

	MYSQL_BIND param1[1];
	MYSQL_BIND param2[9];

	int ret, status;

	char utente_cf[17];
	int idCopia, idPrestito, periodo_consultazione;
	int n_anno, n_mese, n_giorno;
	int ripiano, scaffale;
	float penale;
	MYSQL_TIME data_start;

	printf("\n Enter the user CF: ");
	getInput(17, utente_cf, false);

	if(!setup_prepared_stmt(&prepared_stmt1, "call show_user_loans(?)", conn)){
		finish_with_stmt_error(conn, prepared_stmt1, "Unable to initialize show_user_loans statement\n", false);
	}

	// Prepare parameters
	memset(param1, 0, sizeof(param1));

	param1[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param1[0].buffer = utente_cf;
	param1[0].buffer_length = strlen(utente_cf);

	if(mysql_stmt_bind_param(prepared_stmt1, param1) != 0){
		finish_with_stmt_error(conn, prepared_stmt1, "Could not bind parameters for request inseriton\n", true);
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt1) != 0){
		print_stmt_error(prepared_stmt1, "An error occurred while adding the request");
		if(strcmp("45003", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n---> There are no books on loan");
			return;
		}
	}

	do{
		if(conn->server_status & SERVER_PS_OUT_PARAMS){
			goto next;
		}

		dump_result_set(conn, prepared_stmt1, "**** User ongoing loans ****");

next:
		status = mysql_stmt_next_result(prepared_stmt1);
```

```
		if(status > 0){
			finish_with_stmt_error(conn, prepared_stmt1, "Unexpected condition", true);
		}
	} while (status == 0);


	/* Get total rows in the query */

	printf("\n Ci sono prestiti...");

	mysql_stmt_close(prepared_stmt1);

retry_loan_restitution:

	penale = 0.0;

retry:
	printf("\nEnter the copy ID to be returned: ");
	ret = scanf("%d", &idCopia);
	if(ret == 0){
		printf("not valid input\n");
		flush(stdin);
		goto retry;
	}

	flush(stdin);

retry_ngiorno:
	printf("\nEnter the loan start date:\n Enter the day (1-29,30,31): ");
	ret = scanf("%d", &n_giorno);
	if(ret == 0){
		printf("not valid input\n");
		flush(stdin);
		goto retry_ngiorno;
	}

retry_nmese:
	printf("\n Enter the month (1-12): ");
	ret = scanf("%d", &n_mese);
	if(ret == 0){
		printf("input non valido\n");
		flush(stdin);
		goto retry_nmese;
	}

retry_anno:
	printf("\n Enter the year: ");
	ret = scanf("%d", &n_anno);
	if(ret == 0){
		printf("not valid input\n");
		flush(stdin);
		goto retry_anno;
	}

	data_start.day = n_giorno;
	data_start.month = n_mese;
	data_start.year = n_anno;


retry2:
```

```
        printf("\nEnter the duration of the loan: ");
        ret = scanf("%d", &periodo_consultazione);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry2;
        }
retry3:
        printf("\nEnter the loan ID: ");
        ret = scanf("%d", &idPrestito);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry3;
        }
retry4:
        printf("\nEnter the tray: ");
        ret = scanf("%d", &ripiano);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry4;
        }

retry5:
        printf("\nEnter the shelf ");
        ret = scanf("%d", &scaffale);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry5;
        }

        if(!setup_prepared_stmt(&prepared_stmt2, "call loan_restitution(?,?,?,?,?,?,?,?)", conn)){
                finish_with_stmt_error(conn, prepared_stmt2, "Unable to initialize loan_restitution statement\n", false);
        }

        // Prepare parameters
        memset(param2, 0, sizeof(param2));

        param2[0].buffer_type = MYSQL_TYPE_LONG;
        param2[0].buffer = &idCopia;
        param2[0].buffer_length = sizeof(idCopia);

        param2[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param2[1].buffer = utente_cf;
        param2[1].buffer_length = strlen(utente_cf);

        param2[2].buffer_type = MYSQL_TYPE_DATE;
        param2[2].buffer = &data_start;
        param2[2].buffer_length = sizeof(data_start);

        param2[3].buffer_type = MYSQL_TYPE_LONG;
        param2[3].buffer = &periodo_consultazione;
        param2[3].buffer_length = sizeof(periodo_consultazione);

        param2[4].buffer_type = MYSQL_TYPE_LONG;
        param2[4].buffer = &idPrestito;
        param2[4].buffer_length = sizeof(idPrestito);
```

```
	param2[5].buffer_type = MYSQL_TYPE_LONG;
	param2[5].buffer = &ripiano;
	param2[5].buffer_length = sizeof(ripiano);

	param2[6].buffer_type = MYSQL_TYPE_LONG;
	param2[6].buffer = &scaffale;
	param2[6].buffer_length = sizeof(scaffale);

	param2[7].buffer_type = MYSQL_TYPE_FLOAT; // OUT
	param2[7].buffer = &penale;
	param2[7].buffer_length = sizeof(penale);


	if(mysql_stmt_bind_param(prepared_stmt2, param2) != 0){
		finish_with_stmt_error(conn, prepared_stmt2, "Could not bind parameters for loan inseriton\n", true);
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt2) != 0){
		print_stmt_error(prepared_stmt2, "An error occurred while adding the loan");
		if(strcmp("45017", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
			printf("\n---> Inserted information not valid");
			mysql_stmt_close(prepared_stmt2);
			goto retry_loan_restitution;
		}

		if(strcmp("23000", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
			printf("\n---> Invalid foreign key");
			mysql_stmt_close(prepared_stmt2);
			goto retry_loan_restitution;
		}
	}

	// Get back the ID of the newly_added loan
	memset(param2, 0, sizeof(param2));
	param2[0].buffer_type = MYSQL_TYPE_FLOAT;
	param2[0].buffer = &penale;
	param2[0].buffer_length = sizeof(penale);

	if(mysql_stmt_bind_result(prepared_stmt2, param2)){
		finish_with_stmt_error(conn, prepared_stmt2, "Could not retrieve output parameter", true);
	}

	// Retrieve output parameter
	if(mysql_stmt_fetch(prepared_stmt2)){
		finish_with_stmt_error(conn, prepared_stmt2, "Could not buffer the results", true);

	}

	printf("\n**** Copy correctly returned with a penalty to be paid of: %.2f euro ****\n", penale);


	mysql_stmt_close(prepared_stmt2);



}
```

```
void run_as_bibliotecario(MYSQL *conn, char username[46]){

	char options[7] = {'1', '2', '3', '4', '5', '6', '7'};
	char op;

	printf("Switching to libriarian role...\n");

	if(!parse_config("users/bibliotecario.json", &conf)){
		fprintf(stderr, "Unable to load libriarian 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);
	}

	strcpy(username_g, username);
	printf("--> %s", username_g);

	while(true){

		printf("\033[2J\033[H");
		printf("*** What should I do for you? ***\n\n");
		printf("1) Disease request\n");
		printf("2) Add user\n");
		printf("3) Copies availability\n");
		printf("4) Book on loan report\n");
		printf("5) Contact users\n");
		printf("6) Loan restitution\n");
		printf("7) Quit\n");

		op = multiChoice("Select an option", options, 7);

		switch(op) {
			case '1':

				disease_request(conn);
				break;

			case '2':

				add_new_user(conn);
				break;

			case '3':

				disponibilita_copie(conn);
				break;

			case '4':

				report_libri(conn);
				break;

			case '5':

				contatta_utente(conn);
				break;
```

```
                        case '6':

                                restituzione_prestito(conn);
                                break;

                        case '7':
                                return;

                        default:
                                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                                abort();
                }

                printf("\npremere invio per continuare..\n");
                getchar();

        }


}
```

//

utente.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#include "defines.h"

#define flush(stdin) while(getchar() != '\n')

char username_g[46];

static void loan_restitution(MYSQL *conn){

        MYSQL_STMT *prepared_stmt1;
        MYSQL_STMT *prepared_stmt2;

        MYSQL_BIND param1[1];
        MYSQL_BIND param2[9];

        int ret, rows_count, status;
        int idCopia, idPrestito, periodo_consultazione;
        int n_anno, n_mese, n_giorno;
        MYSQL_TIME data_start;
        int ripiano, scaffale;
        float penale;

        if(!setup_prepared_stmt(&prepared_stmt1, "call show_pending_loans(?)", conn)){
                finish_with_stmt_error(conn, prepared_stmt1, "Unable to initialize show_pending_loans statement\n",
false);
        }

        // Prepare parameters
```

```
	memset(param1, 0, sizeof(param1));

	param1[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param1[0].buffer = username_g;
	param1[0].buffer_length = strlen(username_g);

	if(mysql_stmt_bind_param(prepared_stmt1, param1) != 0){
		finish_with_stmt_error(conn, prepared_stmt1, "Could not bind parameters for request inseriton\n", true);
	}

	// Run procedure
	if(mysql_stmt_execute(prepared_stmt1) != 0){
		print_stmt_error(prepared_stmt1, "An error occurred while adding the request");
		if(strcmp("45003", mysql_stmt_sqlstate(prepared_stmt1)) == 0){
			printf("\n---> There are no books on loan");
			return;
		}
	}

	do{
		if(conn->server_status & SERVER_PS_OUT_PARAMS){
			goto next;
		}

		dump_result_set(conn, prepared_stmt1, "**** User ongoing loans ****");

next:
		status = mysql_stmt_next_result(prepared_stmt1);
		if(status > 0){
			finish_with_stmt_error(conn, prepared_stmt1, "Unexpected condition", true);
		}
	} while (status == 0);


	/* Get total rows in the query */
	rows_count = mysql_stmt_num_rows(prepared_stmt1);
	fprintf(stdout, " total rows in SELECT statement: %d\n", rows_count);

	printf("\n There are loans...");

	mysql_stmt_close(prepared_stmt1);

retry_loan_restitution:

retry:
	printf("\nInsert the ID of the copie to be returned: ");
	ret = scanf("%d", &idCopia);
	if(ret == 0){
		printf("not valid input\n");
		flush(stdin);
		goto retry;
	}

	flush(stdin);

retry_ngiorno:
	printf("\nInsert the start date of the load: \n Insert the day: ");
	ret = scanf("%d", &n_giorno);
	if(ret == 0){
		printf("not valid input\n");
```

```
                flush(stdin);
                goto retry_ngiorno;
        }

retry_nmese:
        printf("\n Insert the month: ");
        ret = scanf("%d", &n_mese);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry_nmese;
        }

retry_anno:
        printf("\n Insert the year: ");
        ret = scanf("%d", &n_anno);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry_anno;
        }

        data_start.day = n_giorno;
        data_start.month = n_mese;
        data_start.year = n_anno;


retry2:
        printf("\nInsert the duration of the load: ");
        ret = scanf("%d", &periodo_consultazione);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry2;
        }
retry3:
        printf("\nInsert the loan's ID: ");
        ret = scanf("%d", &idPrestito);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry3;
        }
retry4:
        printf("\nInsert the tray: ");
        ret = scanf("%d", &ripiano);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry4;
        }

retry5:
        printf("\nInserire lo shelf: ");
        ret = scanf("%d", &scaffale);
        if(ret == 0){
                printf("not valid input\n");
                flush(stdin);
                goto retry5;
        }
```

```
printf("\n INFO: %d - %s - (%d-%d-%d) - %d - %d", idCopia, username_g, data_start.day, data_start.month, data_start.year, periodo_consultazione, idPrestito);

if(!setup_prepared_stmt(&prepared_stmt2, "call loan_restitution_user(?,?,?,?,?,?,?,?)", conn)){
        finish_with_stmt_error(conn, prepared_stmt2, "Unable to initialize loan_restitution statement\n", false);
}

// Prepare parameters
memset(param2, 0, sizeof(param2));

param2[0].buffer_type = MYSQL_TYPE_LONG;
param2[0].buffer = &idCopia;
param2[0].buffer_length = sizeof(idCopia);

param2[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param2[1].buffer = username_g;
param2[1].buffer_length = strlen(username_g);

param2[2].buffer_type = MYSQL_TYPE_DATE;
param2[2].buffer = &data_start;
param2[2].buffer_length = sizeof(data_start);

param2[3].buffer_type = MYSQL_TYPE_LONG;
param2[3].buffer = &periodo_consultazione;
param2[3].buffer_length = sizeof(periodo_consultazione);

param2[4].buffer_type = MYSQL_TYPE_LONG;
param2[4].buffer = &idPrestito;
param2[4].buffer_length = sizeof(idPrestito);

param2[5].buffer_type = MYSQL_TYPE_LONG;
param2[5].buffer = &ripiano;
param2[5].buffer_length = sizeof(ripiano);

param2[6].buffer_type = MYSQL_TYPE_LONG;
param2[6].buffer = &scaffale;
param2[6].buffer_length = sizeof(scaffale);

param2[7].buffer_type = MYSQL_TYPE_FLOAT; // OUT
param2[7].buffer = &penale;
param2[7].buffer_length = sizeof(penale);


if(mysql_stmt_bind_param(prepared_stmt2, param2) != 0){
        finish_with_stmt_error(conn, prepared_stmt2, "Could not bind parameters \n", true);
}

// Run procedure
if(mysql_stmt_execute(prepared_stmt2) != 0){
        print_stmt_error(prepared_stmt2, "An error occurred while adding the loan");
        if(strcmp("45017", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
                printf("\n---> Inserted information not valid");
                mysql_stmt_close(prepared_stmt2);
                goto retry_loan_restitution;
        }

        if(strcmp("23000", mysql_stmt_sqlstate(prepared_stmt2)) == 0){
                printf("\n---> Invalid foreign key");
```

```
                        mysql_stmt_close(prepared_stmt2);
                        goto retry_loan_restitution;
                }
        }

        // Get back the ID of the newly_added loan
        memset(param2, 0, sizeof(param2));
        param2[0].buffer_type = MYSQL_TYPE_FLOAT;
        param2[0].buffer = &penale;
        param2[0].buffer_length = sizeof(penale);

        if(mysql_stmt_bind_result(prepared_stmt2, param2)){
                finish_with_stmt_error(conn, prepared_stmt2, "Could not retrieve output parameter", true);
        }

        // Retrieve output parameter
        if(mysql_stmt_fetch(prepared_stmt2)){
                finish_with_stmt_error(conn, prepared_stmt2, "Could not buffer the results", true);

        }

        printf("\n**** Copy correctly returned with a penalty to be paid of: %.2f euro ****\n", penale);

        mysql_stmt_close(prepared_stmt2);


}



void run_as_utente(MYSQL *conn, char username[46]){

        char options[2] = {'1', '2'};
        char op;

        printf("Switching to user role...\n");

        if(!parse_config("users/utente.json", &conf)){
                fprintf(stderr, "Unable to load user 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);
        }

        strcpy(username_g, username);
        printf("--> %s", username_g);

        while(true){

                printf("\033[2J\033[H");
                printf("*** What should I do for you? ***\n\n");
                printf("1) Loan restitution\n");
                printf("2) Quit\n");

                op = multiChoice("Select an option", options, 2);

                switch(op) {
```

```
                        case '1':

                                loan_restitution(conn);
                                break;

                        case '2':
                                return;

                        default:
                                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                                abort();
                }

                printf("\npremere invio per continuare..\n");
                getchar();

        }
}
```

//

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_lenght = true;

        *stmt = mysql_stmt_init(conn);
        if(*stmt == NULL){
                print_error(conn, "Could not initialize statement handler");
```

```
		return false;
	}

	if(mysql_stmt_prepare(*stmt, statement, strlen(statement)) != 0){
		print_stmt_error(*stmt, "Could not prepare statement");
		return false;
	}

	mysql_stmt_attr_set(*stmt, STMT_ATTR_UPDATE_MAX_LENGTH, &update_lenght);

	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);

	for (i = 0; i < mysql_num_fields (res_set); i++) {
		field = mysql_fetch_field (res_set);
		col_len = strlen(field->name);

		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;
        MYSQL_TIME *date;
        MYSQL_TIME *time;
        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);

                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 output buffers\n", true);
	}
}

if(mysql_stmt_bind_result(stmt, rs_bind)) {
	finish_with_stmt_error(conn, stmt, "Unable to bind output 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_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;

                                case MYSQL_TYPE_TIME:
                                        time = (MYSQL_TIME *)rs_bind[i].buffer;
                                        printf(" %02d:%02d:%02d         |", time->hour, time->minute, time-
>second);
                                        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);
        }
}
```

//

inout.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <pthread.h>
#include <signal.h>
#include <stdbool.h>

#include "defines.h"

// Per la gestione dei segnali
static volatile sig_atomic_t signo;
typedef struct sigaction sigaction_t;
static void handler(int s);

char *getInput(unsigned int lung, char *stringa, bool hide)
{
        char c;
        unsigned int i;

        // Dichiara le variabili necessarie ad un possibile mascheramento dell'input
        sigaction_t sa, savealrm, saveint, savehup, savequit, saveterm;
        sigaction_t savetstp, savettin, savettou;
        struct termios term, oterm;

        if(hide) {
                // Svuota il buffer
                (void) fflush(stdout);

                // Cattura i segnali che altrimenti potrebbero far terminare il programma, lasciando l'utente senza output
sulla shell
                sigemptyset(&sa.sa_mask);
                sa.sa_flags = SA_INTERRUPT; // Per non resettare le system call
                sa.sa_handler = handler;
                (void) sigaction(SIGALRM, &sa, &savealrm);
                (void) sigaction(SIGINT, &sa, &saveint);
                (void) sigaction(SIGHUP, &sa, &savehup);
                (void) sigaction(SIGQUIT, &sa, &savequit);
                (void) sigaction(SIGTERM, &sa, &saveterm);
                (void) sigaction(SIGTSTP, &sa, &savetstp);
                (void) sigaction(SIGTTIN, &sa, &savettin);
                (void) sigaction(SIGTTOU, &sa, &savettou);

                // Disattiva l'output su schermo
                if (tcgetattr(fileno(stdin), &oterm) == 0) {
                        (void) memcpy(&term, &oterm, sizeof(struct termios));
                        term.c_lflag &= ~(ECHO|ECHONL);
```

```
                (void) tcsetattr(fileno(stdin), TCSAFLUSH, &term);
        } else {
                (void) memset(&term, 0, sizeof(struct termios));
                (void) memset(&oterm, 0, sizeof(struct termios));
        }
}

// Acquisisce da tastiera al più lung - 1 caratteri
for(i = 0; i < lung; i++) {
        (void) fread(&c, sizeof(char), 1, stdin);
        if(c == '\n') {
                stringa[i] = '\0';
                break;
        } else
                stringa[i] = c;

        // Gestisce gli asterischi
        if(hide) {
                if(c == '\b') // Backspace
                        (void) write(fileno(stdout), &c, sizeof(char));
                else
                        (void) write(fileno(stdout), "*", sizeof(char));
        }
}

// Controlla che il terminatore di stringa sia stato inserito
if(i == lung - 1)
        stringa[i] = '\0';

// Se sono stati digitati più caratteri, svuota il buffer della tastiera
if(strlen(stringa) >= lung) {
        // Svuota il buffer della tastiera
        do {
                c = getchar();
        } while (c != '\n');
}

if(hide) {
        //L'a capo dopo l'input
        (void) write(fileno(stdout), "\n", 1);

        // Ripristina le impostazioni precedenti dello schermo
        (void) tcsetattr(fileno(stdin), TCSAFLUSH, &oterm);

        // Ripristina la gestione dei segnali
        (void) sigaction(SIGALRM, &savealrm, NULL);
        (void) sigaction(SIGINT, &saveint, NULL);
        (void) sigaction(SIGHUP, &savehup, NULL);
        (void) sigaction(SIGQUIT, &savequit, NULL);
        (void) sigaction(SIGTERM, &saveterm, NULL);
        (void) sigaction(SIGTSTP, &savetstp, NULL);
        (void) sigaction(SIGTTIN, &savettin, NULL);
        (void) sigaction(SIGTTOU, &savettou, NULL);

        // Se era stato ricevuto un segnale viene rilanciato al processo stesso
        if(signo)
                (void) raise(signo);
}

return stringa;
```

```
}

// Per la gestione dei segnali
static void handler(int s) {
	signo = s;
}


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;
		getInput(1, &c, false);

		// 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)
{

	// Genera la stringa delle possibilità
	char *possib = 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);

		char c;
		getInput(1, &c, false);

		// Controlla se è un carattere valido
		for(i = 0; i < num; i++) {
			if(c == choices[i])
				return c;
		}
	}
}
```

//

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) || /* 0-9 */
									(js[parser->pos] >= 65 && js[parser->pos]
<= 70) || /* A-F */
									(js[parser->pos] >= 97 && js[parser->pos]
<= 102))) { /* a-f */
							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;
}

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;
}
```

//