|  |  |
| --- | --- |
| manager |  |

Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

Sistema per la gestione di noleggio di videocassette e DVD

0243651

Pierpaolo Spaziani

**Indice**

[1. Descrizione del Minimondo 3](#_Toc606296459)

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

[3. Progettazione concettuale 5](#_Toc2081466291)

[4. Progettazione logica 6](#_Toc2147004904)

[5. Progettazione fisica 8](#_Toc518560220)

[Appendice: Implementazione 9](#_Toc403811585)

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | Si vuole realizzare un sistema informativo per la gestione di una catena di centri di servizio per il noleggio delle videocassette e DVD, tenendo conto delle seguenti informazioni. Ogni centro di servizio è identificato attraverso un codice numerico univoco; inoltre viene riportato l’indirizzo del centro, i contatti telefonici ed email, il nome di un responsabile. Il sistema mantiene le informazioni relative a tutte le persone impiegate presso la catena. Per ciascun impiegato sono noti il codice fiscale, il nome, il titolo di studio ed un recapito. Gli impiegati possono essere spostati da un centro all’altro a seconda delle esigenze; si vuole pertanto tenere traccia di tutti gli intervalli di tempo in cui un impiegato ha prestato servizio presso un centro e della carica che ha rivestito in quel periodo (per esempio, cassiere o commesso). Le informazioni sul personale sono gestite dai manager, che sono in grado di inserire nuovo personale nel sistema e di visualizzare report mensili ed annuali sulle ore lavorate e sui luoghi di lavoro. I turni di lavoro sono inseriti, su base mensile, sempre dai manager. I film disponibili presso la catena sono identificati dal titolo e dal nome del regista; inoltre sono noti l’anno in cui il film è stato girato, l’elenco degli attori principali del film, il costo corrente di noleggio della videocassetta ed eventualmente i film disponibili presso la catena di cui il film in questione rappresenta la versione “remake”. Per ogni film è nota la collocazione all’interno di ciascun centro di servizio. In particolare, sono noti il settore, la posizione all’interno del settore ed il numero di copie in cui il film è disponibile. Ciascun settore è identificato attraverso un codice numerico univoco all’interno del centro di servizi e dal codice del centro di servizio stesso. I film sono differenziati in classici e nuove uscite, ciascuna associata ad un costo di noleggio giornaliero differente. I clienti della catena, al momento del noleggio, ricevono una tessera cliente. Per ciascun cliente devono essere mantenute tutte le informazioni anagrafiche e viene associato anche un numero arbitrario di indirizzi e di recapiti (telefono, email, cellulare) a cui possono essere contattati. Quando un cliente effettua un noleggio, viene registrata la data entro cui il film dovrà essere restituito. Il personale della catena può gestire l’anagrafica dei clienti e gestire gli ordini. Inoltre, può visualizzare in ogni momento, per ciascun centro di servizio, quali titoli sono associati ad un noleggio scaduto e quali sono i clienti che hanno effettuato tali noleggi. |

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

|  |  |  |  |
| --- | --- | --- | --- |
| **Linea** | **Termine** | **Nuovo termine** | **Motivo correzione** |
| 6 | Recapito | Recapito Telefonico | Recapito è un termine ambiguo che non specifica la tipologia. |
| 28 | Film | Copia del Film | Il cliente noleggia una copia fisica del film. |

### Specifica disambiguata

|  |
| --- |
| Si vuole realizzare un sistema informativo per la gestione di una catena di centri di servizio per il noleggio delle videocassette e DVD, tenendo conto delle seguenti informazioni. Ogni centro di servizio è identificato attraverso un codice numerico univoco; inoltre viene riportato l’indirizzo del centro, i contatti telefonici ed email, il nome di un responsabile. Il sistema mantiene le informazioni relative a tutte le persone impiegate presso la catena. Per ciascun impiegato sono noti il codice fiscale, il nome, il titolo di studio ed un recapito telefonico. Gli impiegati possono essere spostati da un centro all’altro a seconda delle esigenze; si vuole pertanto tenere traccia di tutti gli intervalli di tempo in cui un impiegato ha prestato servizio presso un centro e della carica che ha rivestito in quel periodo (per esempio, cassiere o commesso). Le informazioni sul personale sono gestite dai manager, che sono in grado di inserire nuovo personale nel sistema e di visualizzare report mensili ed annuali sulle ore lavorate e sui luoghi di lavoro. I turni di lavoro sono inseriti, su base mensile, sempre dai manager. I film disponibili presso la catena sono identificati dal titolo e dal nome del regista; inoltre sono noti l’anno in cui il film è stato girato, l’elenco degli attori principali del film, il costo corrente di noleggio della videocassetta ed eventualmente i film disponibili presso la catena di cui il film in questione rappresenta la versione “remake”. Per ogni film è nota la collocazione all’interno di ciascun centro di servizio. In particolare, sono noti il settore, la posizione all’interno del settore ed il numero di copie in cui il film è disponibile. Ciascun settore è identificato attraverso un codice numerico univoco all’interno del centro di servizi e dal codice del centro di servizio stesso. I film sono differenziati in classici e nuove uscite, ciascuna associata ad un costo di noleggio giornaliero differente. I clienti della catena, al momento del noleggio, ricevono una tessera cliente. Per ciascun cliente devono essere mantenute tutte le informazioni anagrafiche e viene associato anche un numero arbitrario di indirizzi e di recapiti (telefono, email, cellulare) a cui possono essere contattati. Quando un cliente effettua un noleggio, viene registrata la data entro cui la copia del film dovrà essere restituita. Il personale della catena può gestire l’anagrafica dei clienti e gestire gli ordini. Inoltre, può visualizzare in ogni momento, per ciascun centro di servizio, quali titoli sono associati ad un noleggio scaduto e quali sono i clienti che hanno effettuato tali noleggi. |

## Glossario dei Termini

|  |  |  |  |
| --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Centro | Centro di servizio per il noleggio delle videocassette e DVD della catena. | Centro di servizio  Luogo di lavoro | Impiegati  Settore |
| Impiegati | Impiegati che lavorano per la catena di video noleggio.  I responsabili e i manager ne sono una specializzazione parziale. | Personale | Centro |
| Settore | Settore specifico di un centro della catena. |  | Centro  Film |
| Film | Film abilitato al noleggio nel sistema della catena. | Titoli | Settore  Noleggio |
| Noleggio | Noleggio da parte del cliente di una copia di un film. | Ordine | Film  Cliente |
| Cliente | Cliente della catena di noleggio. |  | Noleggio |

## Raggruppamento dei requisiti in insiemi omogenei

|  |
| --- |
| **Frasi relative a CENTRO** |
| Ogni centro di servizio è identificato attraverso un codice numerico univoco; inoltre viene riportato l’indirizzo del centro, i contatti telefonici ed email, il nome di un responsabile. |

|  |
| --- |
| **Frasi relative a IMPIEGATI** |
| Per ciascun impiegato sono noti il codice fiscale, il nome, il titolo di studio ed un recapito telefonico. Gli impiegati possono essere spostati da un centro all’altro a seconda delle esigenze; si vuole pertanto tenere traccia di tutti gli intervalli di tempo in cui un impiegato ha prestato servizio presso un centro e della carica che ha rivestito in quel periodo (per esempio, cassiere o commesso).  Le informazioni sul personale sono gestite dai manager, che sono in grado di inserire nuovo personale nel sistema e di visualizzare report mensili ed annuali sulle ore lavorate e sui luoghi di lavoro. I turni di lavoro sono inseriti, su base mensile, sempre dai manager.  Il personale della catena può gestire l’anagrafica dei clienti e gestire gli ordini. Inoltre, può visualizzare in ogni momento, per ciascun centro di servizio, quali titoli sono associati ad un noleggio scaduto e quali sono i clienti che hanno effettuato tali noleggi. |

|  |
| --- |
| **Frasi relative a SETTORE** |
| Ciascun settore è identificato attraverso un codice numerico univoco all’interno del centro di servizi e dal codice del centro di servizio stesso. |

|  |
| --- |
| **Frasi relative a FILM** |
| I film disponibili presso la catena sono identificati dal titolo e dal nome del regista; inoltre sono noti l’anno in cui il film è stato girato, l’elenco degli attori principali del film, il costo corrente di noleggio della videocassetta ed eventualmente i film disponibili presso la catena di cui il film in questione rappresenta la versione “remake”.  Per ogni film è nota la collocazione all’interno di ciascun centro di servizio. In particolare, sono noti il settore, la posizione all’interno del settore ed il numero di copie in cui il film è disponibile.  I film sono differenziati in classici e nuove uscite, ciascuna associata ad un costo di noleggio giornaliero differente. |

|  |
| --- |
| **Frasi relative a NOLEGGIO** |
| Quando un cliente effettua un noleggio, viene registrata la data entro cui la copia del film dovrà essere restituita. |

|  |
| --- |
| **Frasi relative a CLIENTE** |
| Per ciascun cliente devono essere mantenute tutte le informazioni anagrafiche e viene associato anche un numero arbitrario di indirizzi e di recapiti (telefono, email, cellulare) a cui possono essere contattati. |

# Progettazione concettuale

## Costruzione dello schema E-R

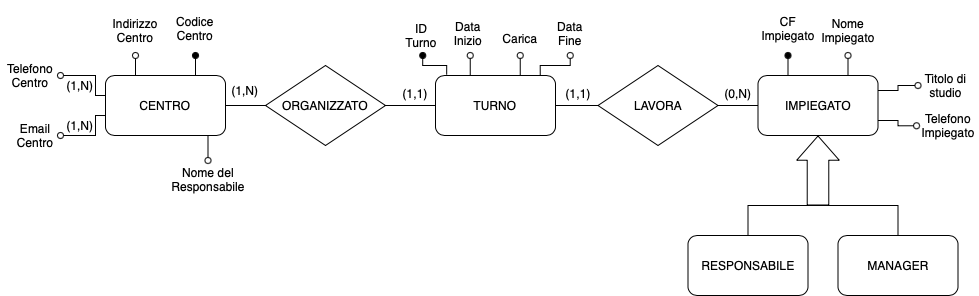
Per lo sviluppo dello schema E-R è stato scelto di utilizzare una strategia bottom-up.

Il primo passo è stato analizzare la specifica con 3 obiettivi:

isolare e risolvere le ambiguità e dubbi;

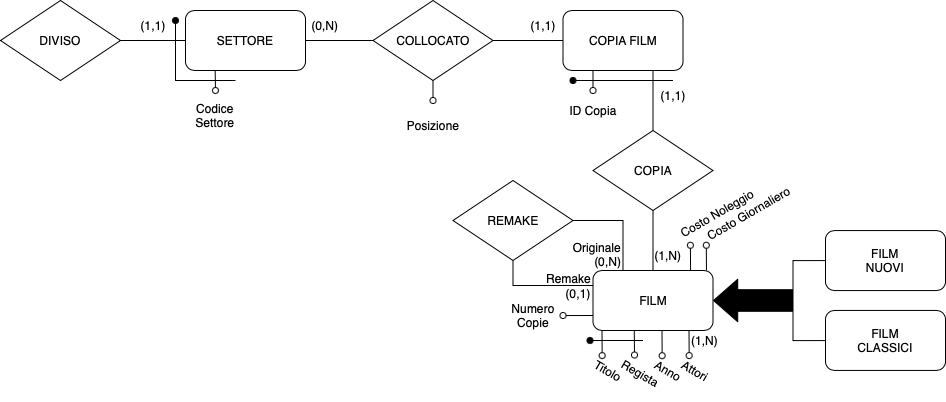
individuare le Entità che avrebbero costituito lo schema coi i relativi attributi;

suddividere la specifica in più sezioni per poterle sviluppare singolarmente ed unirle successivamente in un unico schema.

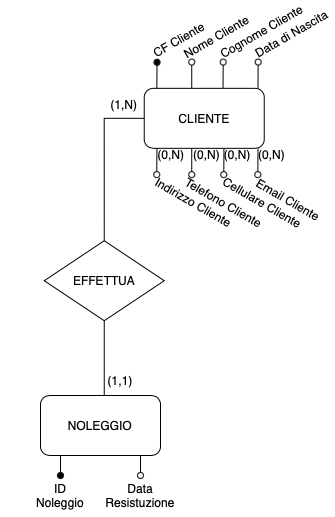
La prima sezione individuata e sviluppata è stata quella riguardante il Centro, gli Impiegati e i Turni:

Le entità Centro e Turno vengono identificate entrambe da un codice univoco, gli Impiegati invece dai loro Codici Fiscali. È stato deciso di considerare i Responsabili dei Centri e i Manager come specializzazioni degli Impiegati.

I Turni sono stati intesi come le giornate di lavoro, quindi quando i Manager organizzano i turni inseriscono nel sistema sia la data di inizio che la data di fine avendo così un intervallo di giorni in cui un determinato impiegato ha lavorato ricoprendo una determinata carica. In questo modo si hanno sia i Turni passati sia i Turni in corso.

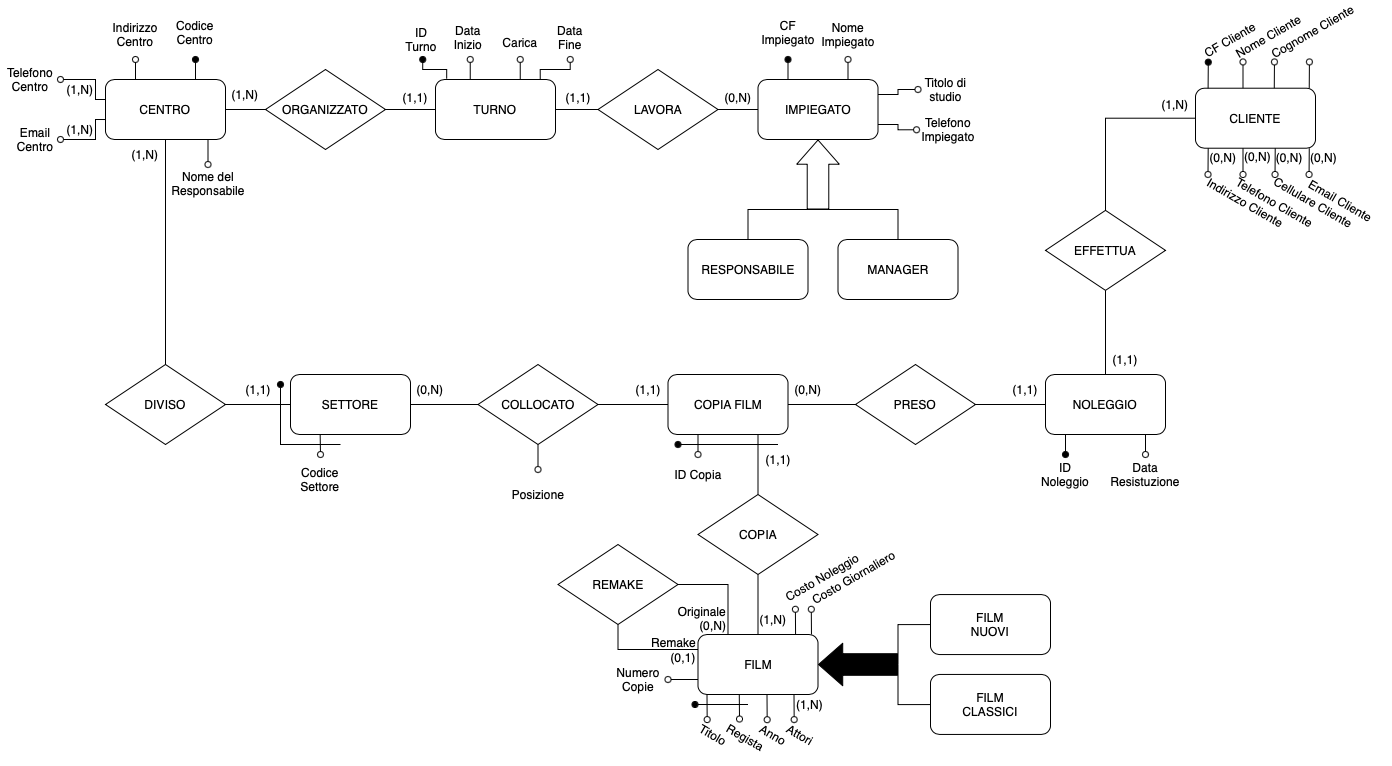
La seconda sezione è quella riguardante i Film ed i Settori:

La Copia del Film è un’istanza dell’entità Film e come tale è un’entità debole. È stato inoltre inserito un Numero di copia per identificarla insieme al relativo Film. I Film vengono identificati dal Titolo e dal Registra e divisi con una generalizzazione totale in Classici e Nuovi. Possiedono una relazione ricorsiva Remake la cui navigazione è facilitata dagli identificatori Originali e Remake. Il Settore è un’entità debole ed è identificato da un Codice unico nel Centro e dal Centro stesso di cui fa parte ed è messo in relazione con le Copie fisiche dei Film in base alla loro collocazione nei Settori stessi.

La terza ed ultima sezione è quella riguardante il Cliente ed il Noleggio:

Il Cliente viene identificato dal Codice Fiscale ed è in relazione con il Noleggio, identificato da un codice univoco e che ha come attributo la Data di restituzione della Copia del Film.

### Integrazione finale

I conflitti tra gli attributi delle entità, come il Codice Fiscale presente sia per gli Impiegati che per i Clienti, sono stati risolti da subito specificando nell’attributo l’entità a cui fanno riferimento.

## Regole aziendali

1. Gli Impiegati non possono avere più Turni nello stesso intervallo di giorni;
2. I Manager non possono lavorare nei Centri (non hanno Turni);
3. Si assume che i Centri siano aperti solo di pomeriggio dalle 16:00 alle 00:00 avendo quindi 8 ore di lavoro giornaliere;
4. I Film hanno un costo di noleggio base più un costo di noleggio giornaliero che parte dal giorno stesso del noleggio;
5. I Film Classici e Nuovi hanno un costo di noleggio giornaliero differente;
6. Si assume il costo di noleggio pari a 1€ per i Film Classici e 3€ per i Nuovi;
7. Si assume il costo giornaliero pari a 0,5€ per i Film Classici e 1€ per i Nuovi;
8. Si assume che ogni giorno extra oltre la scadenza della Data di Restituzione costi 1€.

## Dizionario dei dati

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatori** |
| Centro | Centri di servizio per il noleggio delle videocassette e DVD della catena. | Codice Centro,  Nome del Responsabile,  Indirizzo Centro,  Telefono Centro,  Email Centro | Codice Centro |
| Impiegato | Impiegati che lavorano per la catena di video noleggio.  Comprende anche Responsabili e Manager. | Codice Fiscale Impiegato,  Nome Impiegato,  Titolo di studio,  Telefono Impiegato | Codice Fiscale Impiegato |
| Settore | Settori di un centro della catena. | Centro,  Codice Settore | Centro,  Codice Settore |
| Turno | Turni di lavoro. | ID Turno,  Data Inizio,  Data Fine  Carica | ID Turno |
| Film | Film abilitati al noleggio nella catena. | Titolo,  Regista,  Anno,  Attori,  Costo Noleggio,  Costo Giornaliero,  Numero Copie | Titolo,  Regista |
| Copia Film | Copia fisica di un film che l’utente noleggia. | Film,  Numero Copia | Film,  Numero Copia |
| Noleggio | Noleggio da parte del Cliente. | ID Noleggio,  Data Restituzione | ID Noleggio |
| Cliente | Cliente della catena di noleggio. | Codice Fiscale Cliente,  Nome Cliente,  Cognome Cliente,  Data di nascita,  Indirizzo Cliente,  Telefono Cliente,  Email Cliente,  Cellulare Cliente | Codice Fiscale Cliente |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| **Concetto nello schema** | **Tipo** | **Volume atteso** |
| CENTRO | E | 100 |
| IMPIEGATI | E | 1.500 (~ 15 ogni Centro) |
| RESPONSABILE | E | 100 (1 ogni Centro) |
| MANAGER | E | 10 |
| TURNO | E | 74.500  (~ 50 ogni Impiegato - Manager) |
| SETTORE | E | 500 (5 ogni Centro) |
| FILM | E | 200 |
| COPIA FILM | E | 100.000 (~ 5 ogni Film in ogni Centro)  (Caso Peggiore: 5 \* Film \* Centro) |
| CLIENTE | E | 3.000 (~ 30 ogni Centro) |
| NOLEGGIO | E | 100.000 (Caso Peggiore: = Copia Film) |
| Organizzato | R | 75.000 (= Turno) |
| Lavora | R | 1.490  (= Impiegati - Manager) |
| Diviso | R | 500 (= Settore) |
| Collocato | R | 100.000 (= Copia Film) |
| Remake | R | 19.999 (Caso Peggiore: Film - 1) |
| Copia | R | 100.000 (= Copia Film) |
| Preso | R | 3.000 (= Noleggio) |
| Effettua | R | 3.000 (= Noleggio) |

## Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| **Cod.** | **Descrizione** | **Frequenza attesa** |
| 1. 1 | Inserisci Impiegato | 100/mese (~ 1 in ogni Centro) |
| 1. 2 | Cambia Responsabile | 100/mese (~ 1 in ogni Centro) |
| 1. 4 | Visualizza report mensile dell’Impiegato | 1.490/mese (~ 1 in ogni Impiegato - Manager) |
|  | Visualizza report annuale dell’Impiegato | 1.490/mese (~ 1 in ogni Impiegato - Manager) |
|  | Inserisci turni lavoro | 1.490/mese (~ 1 in ogni Impiegato - Manager) |
|  | Visualizza i turni con i relativi Impiegati del giorno ‘X’ al Centro ‘Y’ | 100/giorno (~ 1 in ogni Centro) |
|  | Inserisci Film | 1/settimana |
|  | Aggiungi Copia Film | 1/mese |
|  | Mostra Film Nuovi | 1/settimana |
|  | Dequalifica Film da Nuovo a Classico | 1/settimana |
|  | Inserisci Cliente | 500/mese (5 in ogni Centro) |
|  | Modifica dati Cliente | 100/mese (1 in ogni Centro) |
|  | Visualizza Film con Noleggio scaduto | 100/giorno (1 in ogni Centro) |
|  | Visualizza Film disponibili al Noleggio | 100/giorno (1 in ogni Centro) |
|  | Effettua Noleggio | 3.000/giorno (30 in ogni Centro) |
|  | Restituisci Film | 3.000/giorno (30 in ogni Centro) |

## Costo delle operazioni

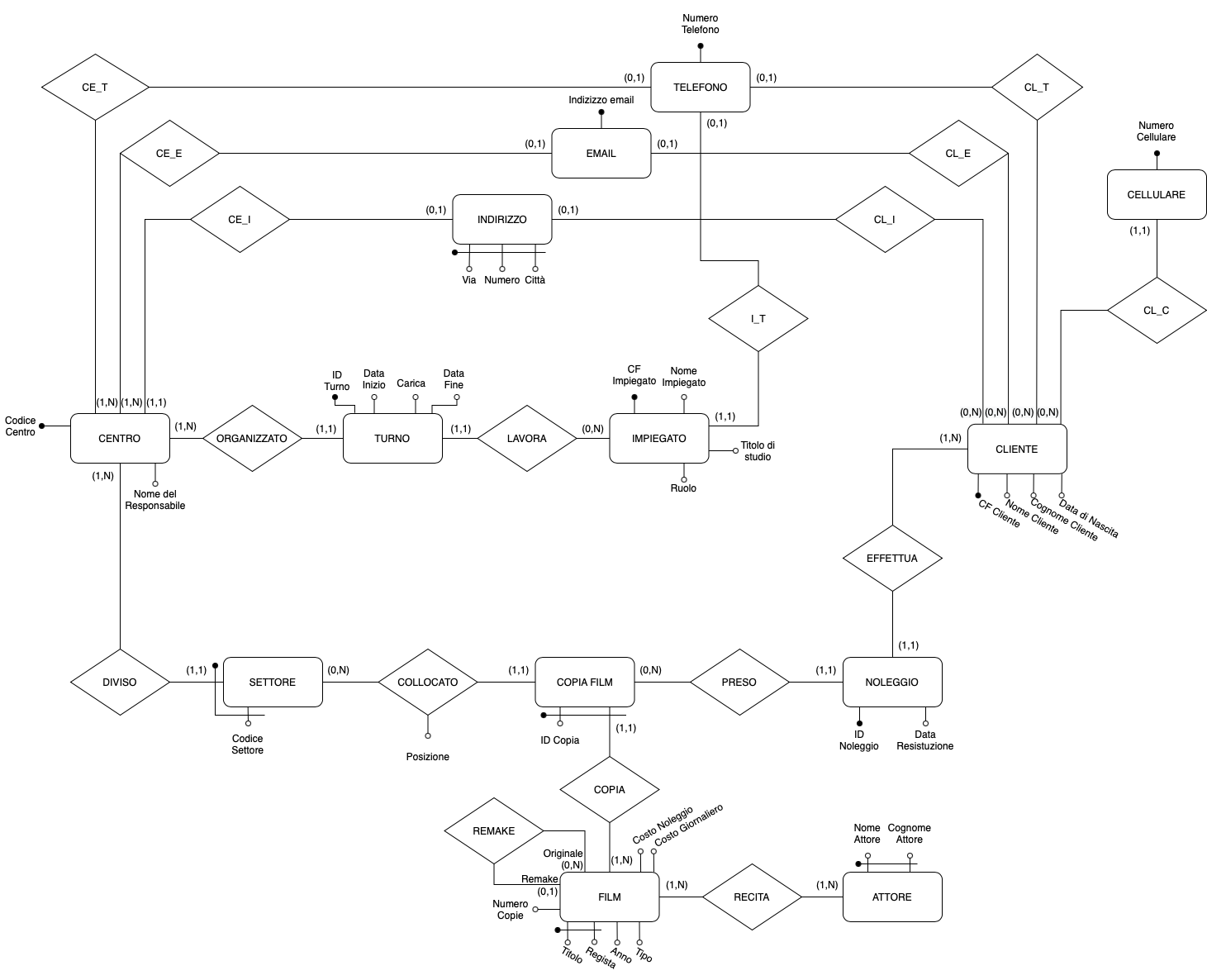
1. 1S Impiegato = 2 \* 100/mese = 200/mese
2. 1L Impiegato + 2S Impiegato + 1L Responsabile+ 2S Responsabile = (1+4+1+4) \* 100/mese = 1.000/mese
3. 1L Turno + 1L Impiegati = (1+1) \* 1.490/mese = 2980/mese
4. 1L Turno + 1L Impiegati = (1+1) \* 1.490/mese = 2980/mese
5. 1L Impiegato + 1S Turno = (1+2) \* 1.490/mese = 4.470/mese
6. 1L Turno + 1L Impiegati + 1L Centro = (1+1+1) \* 100/giorno = 300/giorno
7. 1S Film = 2 \* 1/settimana = 2/settimana
8. 1S Copia Film + 1S Collocato = (2+2) \* 1/mese = 400/mese
9. 1L Film = 1 \* 1/settimana = 1/settimana
10. 1L Film + 1S Film = (1+2) \* 1/settimana = 3/settimana
11. 1S Cliente = 1 \* 500/mese = 500/mese
12. 1L Cliente + 1S Cliente = (1+2) \* 100/mese = 300/mese
13. 1L Noleggio + 1L Copia Film = (1+1) \* 100/mese = 200/mese
14. 1L Noleggio + 1L Copia Film = (1+1) \* 100/mese = 200/mese
15. 1L Cliente + 1L Copia Film + 1S Noleggio = (1+1+2) \* 3.000/giorno = 12.000/giorno
16. 1S Noleggio = 2 \* 3.000/giorno = 6.000/giorno.

## Ristrutturazione dello schema E-R

Sono state eliminate le generalizzazioni optando per l’accorpamento delle entità figlie da parte del padre, ovvero è stato aggiunto l’attributo “Ruolo” (diverso da “Carica” presente in Turno) per l’entità Impiegato e l’attributo “Tipo” per l’entità Film.

Sono stati inoltre eliminati gli attributi multi-valore inserendo delle entità proprie: Attore, Indirizzo, Telefono, Cellulare, Email.

Sapendo che i Turni sono gestiti su base mensile, è stato deciso di non dividere l’entità Turno in ‘Turno Passato’ e ‘Turno da Fare’ (o anche Passato, Futuro e In Corso) poiché altrimenti la base di dati dovrebbe spostare ogni giorno le tuple di turni completati da una tabella all’altra (in caso di divisione a 3 anche i turni del giorno da Futuri a In Corso). Inoltre la maggior parte delle tuple sarebbero riferite a turni passati quindi, anche dividendo, ‘Turno Passato’ sarebbe comunque molto carico e il vantaggio sarebbe minimo.

Si è preferito quindi avere un’entità più corposa con poche operazioni mensili più “pesanti” rispetto ad entità multiple più leggere (relativamente) con operazioni giornaliere.

## Trasformazione di attributi e identificatori

Non sono presenti ridondanze tra i vari attributi essendo stati specificati sin da subito.

Sono stati sostituiti gli spazi nei nomi degli attributi con “\_” per non creare ambiguità.

Per le entità deboli sono state scelte le seguenti modifiche:

* Settore: Codice\_Centro → Centro\_Settore;
* Copia Film: Titolo, Regista → Titolo\_Copia, Regista\_Copia.

## Traduzione di entità e associazioni

Centro (**Codice\_Centro**, Responsabile, *Indirizzo\_Centro*, *Numero\_Civico*, *Sede*)

Indirizzo.Via → Centro.Indirizzo\_Centro

Indirizzo.Numero → Centro.Numero\_Civico

Indirizzo.Città → Centro.Sede

Turno (**ID\_Turno**, Data\_Inizio, Data\_Fine, Carica, *Centro\_Turno*, *Impiegato\_Turno*)

Centro.Codice\_Centro → Turno.Centro\_Tunro

Impiegato.CF\_Impiegato → Turno.Impiegato\_Turno

Impiegato (**CF\_Impiegato**, Nome\_Impiegato, Titolo\_di\_Studio, Ruolo, *Telefono*)

Telefono.Numero\_Telefono → Impiegato.Telefono

Settore (***Centro\_Settore***, **Codice\_Settore**)

Centro.Codice\_Centro → Settore.Centro\_Settore

Film (**Titolo**, **Regista**, Anno, Numero\_Copie, Costo\_Noleggio, Costo\_Giornaliero, Tipo)

Copia Film (**ID\_Copia**, ***Titolo\_Copia***, ***Regista\_Copia***, *Centro\_Copia*, *Settore\_Copia*, Posizione)

Film.Titolo → Copia\_Film.Titolo\_Copia

Film.Regista → Copia\_Film.Regista\_ Copia

Settore.Numero\_Centro → Copia\_Film.Centro\_ Copia

Settore.Codice\_Settore → Copia\_Film.Settore\_ Copia

Remake (***Titolo\_Remake***, ***Regista\_Remake***, *Titolo\_Originale*, *Regista\_Originale*)

Film.Titolo → Remake.Titolo\_Remake, Remake.Titolo\_Originale

Film.Regista → Remake.Regista\_Remake, Remake.Regista\_Originale

Attore (**Nome\_Attore**, **Cognome\_Attore**)

Recita (***Titolo\_Recita***, ***Regista\_Recita***, ***Nome\_Attore\_Recita***, ***Cognome\_Attore\_Recita***)

Film.Titolo → Recita.Titolo\_Recita

Film.Regista → Recita.Regista\_Recita

Attore.Nome\_Attore → Recita.Nome\_Attore\_Recita

Attore.Cognome\_Attore → Recita.Cognome\_Attore\_Recita

Cliente (**CF\_Cliente**, Nome\_Cliente, Cognome\_Cliente, Data\_di\_Nascita)

Noleggio (**ID\_Noleggio**, Data\_Restituzione, *Numero\_Copia*, *Titolo\_Noleggio*, *Regista\_Noleggio*, *Cliente\_Noleggio*)

Copia\_Film.ID\_Copia → Noleggio.Numero\_Copia

Copia\_Film.Titolo\_Film → Noleggio.Titolo\_Noleggio

Copia\_Film.Regista\_Film → Noleggio.Regista\_Noleggio

Cliente.CF\_Cliente → Noleggio.Cliente\_Noleggio

Indirizzo (**Via**, **Numero**, **Città**, *Cliente\_Indirizzo*)

Cliente.CF\_Cliente → Indirizzo.Cliente\_Indirizzo

Email (**Indirizzo\_Email**, *Centro\_Email*\*, *Cliente\_Email*\*)

Centro.Codice\_Centro → Indirizzo.Centro\_Email

Cliente.CF\_Cliente → Email.Cliente\_Email

Telefono (**Numero\_Telefono**, *Centro\_Telefono*\*, *Cliente\_Telefono*\*)

Centro.Codice\_Centro → Indirizzo.Centro\_Telefono

Cliente.CF\_Cliente → Telefono.Cliente\_Telefono

Cellulare (**Numero\_Cellulare**, *Cliente\_Cellulare*)

Cliente.CF\_Cliente → Cellulare.Cliente\_Cellulare

## Normalizzazione del modello relazionale

**1NF**

Il modello relazionale proposto rispetta le regole della prima forma normale, ovvero, per ogni relazione è presente una chiave primaria, non vi sono gruppo di attributi che si ripetono all’interno di una relazione, tutti i valori di un attributo sono dello stesso tipo e le colonne sono indivisibili.

**2NF**

La definizione di 2NF dice che relazioni in questa devono essere prima di tutto in 1NF ed inoltre ogni attributo non-chiave deve dipendere funzionalmente dall’intera chiave composta e non da una parte di essa. Per le relazioni che hanno una sola chiave non ci sono problemi, in quanto obbligatoriamente tutti gli attributi dipendono dall’intera chiave ed essendo valida la regola anche per le relazioni con chiavi composte, il modello visto precedentemente rispetta anche la 2NF.

**3NF**

Una relazione per essere in 3NF deve essere in 2NF ed inoltre tutti gli attributi non-chiave devono necessariamente dipendere funzionalmente dalla chiave e non da attributi non-chiave.

Tutte le relazioni dello schema rispecchiano tali caratteristiche.

# Progettazione fisica

## Utenti e privilegi

MANAGER:

Ha tutti i privilegi su tutte le tabelle.

IMPIEGATO:

Centro: SELECT

Impiegato: SELECT

Turno: SELECT

Settore: SELECT

Copia\_Film: SELECT – INSERT – UPDATE – DELETE

Film: SELECT – UPDATE

Remake: SELECT

Recita: SELECT

Attore: SELECT

Noleggio: SELECT – INSERT – UPDATE – DELETE

Cliente: SELECT – INSERT – UPDATE

Indirizzo: SELECT – INSERT – UPDATE – DELETE

Email: SELECT – INSERT – UPDATE – DELETE

Telefono: SELECT – INSERT – UPDATE – DELETE

Cellulare: SELECT – INSERT – UPDATE – DELETE

Sono stati scelti questi privilegi poiché gli Impiegati possono gestire i dati anagrafici dei Clienti, i loro recapiti e i Noleggi in corso. Per quanto riguarda le Copie dei Film possono inserirne di nuove, rimuoverle oppure modificarne i dati nel caso venissero cambiati di posizione. Potendo modificare il numero di copie disponibili devono poter aggiornare anche il relativo attributo nei Film.

## Strutture di memorizzazione

|  |  |  |
| --- | --- | --- |
| **Tabella CENTRO** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[1]](#footnote-1)** |
| **Codice\_Centro** | INT | PK – NN – AI |
| **Responsabile** | VARCHAR(64) | NN |
| **Indirizzo\_Centro** | VARCHAR(64) | NN |
| **Numero\_Civico** | INT | NN |
| **Sede** | VARCHAR(64) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella IMPIEGATO** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **CF\_Impiegato** | VARCHAR(64) | PK – NN |
| **Nome\_Impiegato** | VARCHAR(64) | NN |
| **Titolo\_di\_Studio** | VARCHAR(64) | NN |
| **Telefono** | VARCHAR(64) | NN – UQ |
| **Ruolo** | ENUM(…) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella TURNO** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **ID\_Turno** | INT | PK – NN - AI |
| **Data\_Inizio** | DATE | NN |
| **Data\_Fine** | DATE | NN |
| **Carica** | ENUM(…) | NN |
| **Centro\_Turno** | INT | NN |
| **Impiegato\_Turno** | VARCHAR(64) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella SETTORE** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Centro\_Settore** | INT | PK – NN |
| **Codice\_Settore** | INT | PK – NN |

|  |  |  |
| --- | --- | --- |
| **Tabella COPIA FILM** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **ID\_Copia** | INT | PK – NN |
| **Titolo\_Copia** | VARCHAR(64) | PK – NN |
| **Regista\_Copia** | VARCHAR(64) | PK – NN |
| **Centro\_Copia** | INT | NN |
| **Settore\_Copia** | INT | NN |
| **Posizione** | VARCHAR(64) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella FILM** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Titolo** | VARCHAR(64) | PK – NN |
| **Regista** | VARCHAR(64) | PK – NN |
| **Anno** | YEAR(4) | NN |
| **Numero\_Copie** | INT | NN |
| **Costo\_Noleggio** | FLOAT | NN |
| **Costo\_Giornaliero** | FLOAT | NN |
| **Tipo** | ENUM(…) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella REMAKE** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Titolo\_Remake** | VARCHAR(64) | PK – NN |
| **Regista\_Remake** | VARCHAR(64) | PK – NN |
| **Titolo\_Originale** | VARCHAR(64) | NN |
| **Regista\_Originale** | VARCHAR(64) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella RECITA** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Titolo\_Recita** | VARCHAR(64) | PK – NN |
| **Regista\_Recita** | VARCHAR(64) | PK – NN |
| **Nome\_Attore\_Recita** | VARCHAR(64) | PK – NN |
| **Cognome\_Attore\_Recita** | VARCHAR(64) | PK – NN |

|  |  |  |
| --- | --- | --- |
| **Tabella ATTORE** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Nome\_Attore** | VARCHAR(64) | PK – NN |
| **Cognome\_Attore** | VARCHAR(64) | PK – NN |

|  |  |  |
| --- | --- | --- |
| **Tabella NOLEGGIO** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **ID\_Noleggio** | INT | PK – NN – AI |
| **Data\_Restituzione** | DATE | NN |
| **Numero\_Copia** | INT | NN |
| **Titolo\_Noleggio** | VARCHAR(64) | NN |
| **Regista\_Noleggio** | VARCHAR(64) | NN |
| **Cliente\_Noleggio** | VARCHAR(64) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella CLIENTE** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **CF\_Cliente** | VARCHAR(64) | PK – NN |
| **Nome\_Cliente** | VARCHAR(64) | NN |
| **Cognome\_Cliente** | VARCHAR(64) | NN |
| **Data\_di\_Nascita** | DATE | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella INDIRIZZO** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Via** | VARCHAR(64) | PK – NN |
| **Numero** | INT | PK – NN |
| **Città** | VARCHAR(64) | PK – NN |
| **Cliente\_Indirizzo** | VARCHAR(64) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella EMAIL** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Indirizzo\_Email** | VARCHAR(64) | PK – NN |
| **Centro\_Email** | INT |  |
| **Cliente\_Email** | VARCHAR(64) |  |

|  |  |  |
| --- | --- | --- |
| **Tabella TELEFONO** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Numero\_Telefono** | VARCHAR(64) | PK – NN |
| **Centro\_Telefono** | INT |  |
| **Cliente\_Telefono** | VARCHAR(64) |  |

|  |  |  |
| --- | --- | --- |
| **Tabella CELLULARE** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| **Numero\_Cellulare** | VARCHAR(64) | PK – NN |
| **Cliente\_Cellulare** | VARCHAR(64) | NN |

## Indici

Sono stati inseriti degli indici per le tabelle più corpose, quelle che nella tabella dei volumi ci aspettavamo avessero valori più elevati:

|  |  |
| --- | --- |
| **Tabella TURNO** | |
| **Indice idx\_turno\_centro\_impiegato** | **Tipo:** |
| Centro\_Turno – Impiegato\_Turno | IDX |

È stato utilizzato quest’indice poiché nella tabella Turno ci sono i turni (passati, presenti e futuri) per tutta la catena, con l’indicizzazione quando viene effettuata una ricerca, sia per i report o per visualizzare i turni, risulta più leggera.

|  |  |
| --- | --- |
| **Tabella COPIA\_FILM** | |
| **Indice idx\_copia\_centro\_film** | **Tipo:** |
| Centro\_Copia – Titolo\_Copia | IDX |

È stato utilizzato quest’indice poiché nella tabella Copia\_Film ci sono le copie dei film di tutta la catena, con l’indicizzazione la ricerca dei film disponibili per i singoli centri risulta più leggera.

|  |  |
| --- | --- |
| **Tabella NOLEGGIO** | |
| **Indice idx\_noleggio\_cliente** | **Tipo:** |
| Cliente\_Noleggio | IDX |

È stato utilizzato quest’indice poiché nella tabella Noleggio ci sono tutti i noleggi non restituiti di tutta la catena, con l’indicizzazione la ricerca dei noleggi di un cliente è più leggera.

## Trigger

## CENTRO

## Sono stati inseriti due trigger uguali sia BEFORE INSERT che BEFORE UPDATE per controllare che il nome del responsabile inserito o aggiornato sia di un Impiegato che lavora per la catena:

CREATE DEFINER = CURRENT\_USER TRIGGER `noleggio-film`.`Centro\_BEFORE\_INSERT` BEFORE INSERT ON `Centro` FOR EACH ROW

BEGIN

if not exists (select \* from `Impiegato` where `Nome\_Impiegato` = NEW.Responsabile) then

signal sqlstate '45000' set message\_text = 'Employee not found!';

end if;

END

CREATE DEFINER = CURRENT\_USER TRIGGER `noleggio-film`.`Centro\_BEFORE\_UPDATE` BEFORE UPDATE ON `Centro` FOR EACH ROW

BEGIN

if not exists (select \* from `Impiegato` where `Nome\_Impiegato` = NEW.Responsabile) then

signal sqlstate '45000' set message\_text = 'Employee not found!';

end if;

END

## NOLEGGIO

## È stato inserito un trigger BEFORE INSERT per controllare che la Data di Restituzione non sia già passata:

CREATE DEFINER = CURRENT\_USER TRIGGER `noleggio-film`.`Noleggio\_BEFORE\_INSERT` BEFORE INSERT ON `Noleggio` FOR EACH ROW

BEGIN

if (NEW.Data\_Restituzione < curdate()) then

signal sqlstate '45000' set message\_text = 'The return date is not valid!';

end if;

END

## TURNO

## È stato inserito un trigger BEFORE INSERT per controllare che l’Impiegato inserito per il Turno non sia un Manager, per controllare la validità delle date scelte e per impedire di inserire Turni sovrapposti:

CREATE DEFINER = CURRENT\_USER TRIGGER `noleggio-film`.`Turno\_BEFORE\_INSERT` BEFORE INSERT ON `Turno` FOR EACH ROW

BEGIN

if ((select `Ruolo` from `Impiegato`

where `CF\_Impiegato` = NEW.Impiegato\_Turno) <> 'Impiegato') then

signal sqlstate '45000' set message\_text = 'Managers cannot have workshifts!';

elseif (NEW.Data\_Inizio < curdate()) then

signal sqlstate '45000' set message\_text = 'Past workshifts cannot be insert!';

elseif (NEW.Data\_Inizio > NEW.Data\_Fine) then

signal sqlstate '45000' set message\_text = 'Invalid dates!';

elseif (select count(\*) from `Turno`

where `Impiegato\_Turno` = NEW.Impiegato\_Turno

and `Data\_Inizio` <= NEW.Data\_Inizio

and `Data\_Fine` >= NEW.Data\_Inizio) <> 0 then

signal sqlstate '45000' set message\_text = 'First day not valid, this employee has already a workshift in this day!';

elseif (select count(\*) from `Turno`

where `Impiegato\_Turno` = NEW.Impiegato\_Turno

and `Data\_Inizio` <= NEW.Data\_Fine

and `Data\_Fine` >= NEW.Data\_Fine) <> 0 then

signal sqlstate '45000' set message\_text = 'Last day not valid, this employee has already a workshift in this day!';

elseif (select count(\*) from `Turno`

where `Impiegato\_Turno` = NEW.Impiegato\_Turno

and `Data\_Inizio` >= NEW.Data\_Inizio

and `Data\_Inizio` <= NEW.Data\_Fine) <> 0 then

signal sqlstate '45000' set message\_text = 'This employee has already a smaller workshift in this days or the same!';

end if;

END

## Eventi

## Viste

## Stored Procedures e transazioni

**aggiungi\_attore\_recita**

* Aggiunge l’attore se non è già presente in Attore e inserisce la tupla film-attore in Recita;
* Livello di isolamento della transazione READ COMMITTED, interessa lavorare su dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `aggiungi\_attore\_recita` (in titolo varchar(64), in regista varchar(64), in nome varchar(64), in cognome varchar(64))

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

if not exists (select \* from `Attore` where `Nome\_Attore` = nome and `Cognome\_Attore` = cognome) then

insert into `Attore` values(nome, cognome);

end if;

insert into `Recita` values(titolo, regista, nome, cognome);

commit;

END

**aggiungi\_cellulare**

* Aggiunge un numero di cellulare per il cliente;
* Nessuna transazione poiché è una singola insert.

CREATE PROCEDURE `aggiungi\_cellulare` (in numero varchar(10), in cliente varchar(16))

BEGIN

insert into `Cellulare` values(numero, cliente);

END

**aggiungi\_cliente**

* Aggiunge un nuovo cliente;
* Nessuna transazione poiché è una singola insert.

CREATE PROCEDURE `aggiungi\_cliente` (in cf varchar(64), in nome varchar(64), in cognome varchar(64), in data\_nascita date)

BEGIN

insert into `Cliente` values(cf, nome, cognome, data\_nascita);

END

**aggiungi\_copia**

* Aggiunge una copia di un film;
* Livello di isolamento della transazione READ COMMITTED, interessa lavorare su dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `aggiungi\_copia` (in titolo varchar(64), in regista varchar(64), in centro int, in settore int, in posizione int)

BEGIN

declare num int;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

select count(\*) from `Copia\_Film`

where `Titolo\_Copia` = titolo and `Regista\_Copia` = regista

order by `ID\_Copia` desc

LIMIT 1 into num;

insert into `Copia\_Film` values (num+1,titolo, regista, centro, settore, posizione);

update `Film` set `Numero\_Copie` = `Numero\_Copie` + 1

where `Titolo` = titolo

and `Regista` = regista;

commit;

END

**aggiungi\_email**

* Aggiunge un’email per il centro o per il cliente;
* Nessuna transazione poiché è una singola insert.

CREATE PROCEDURE `aggiungi\_email` (in indirizzo varchar(64), in centro int, in cliente varchar(64), in tipo bool)

BEGIN

if tipo is true then

insert into `Email`(`Indirizzo\_Email`,`Centro\_Email`) values(indirizzo, centro);

else

insert into `Email`(`Indirizzo\_Email`,`Cliente\_Email`) values(indirizzo, cliente);

end if;

END

**aggiungi\_film**

* Aggiunge un nuovo film;
* Livello di isolamento della transazione READ UNCOMMITTED, utilizzata solo per garantire che le insert vengano entrambe o nessuna.

CREATE PROCEDURE `aggiungi\_film` (in titolo varchar(64), in regista varchar(64), in anno year, in nuovo bool, in remake bool, in tit\_or varchar(64), in reg\_or varchar(64))

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read uncommitted;

start transaction;

if nuovo is true then

insert into `Film` values(titolo,regista,anno,'0','3.0','0.5','New');

else

insert into `Film` values(titolo,regista,anno,'0','1.0','0.5','Classic');

end if;

if remake is true then

insert into `Remake` values(titolo,regista,tit\_or,reg\_or);

end if;

commit;

END

**aggiungi\_impiegato**

* Aggiunge un nuovo impiegato e crea l’utente per fare il login;
* Livello di isolamento della transazione READ UNCOMMITTED, utilizzata solo per garantire che le insert vengano entrambe o nessuna.

CREATE PROCEDURE `aggiungi\_impiegato` (in cf varchar(64), in nome varchar(64), in titolo\_studio varchar(64), in telefono varchar(10), in ruolo varchar(12), in pssw varchar(45))

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read uncommitted;

start transaction;

insert into Impiegato values(cf, nome, titolo\_studio, telefono, ruolo);

insert into utenti values(cf, MD5(pssw));

commit;

END

**aggiungi\_indirizzo**

* Aggiunge un indirizzo per il cliente;
* Nessuna transazione poiché è una singola insert.

CREATE PROCEDURE `aggiungi\_indirizzo` (in via varchar(64), in numero int, in citta varchar(64), in cliente varchar(64))

BEGIN

insert into `Indirizzo`(`Via`,`Numero`,`Città`,`Cliente\_Indirizzo`) values(via, numero, citta, cliente);

END

**aggiungi\_telefono**

* Aggiunge un telefono per un cliente o per un centro;
* Nessuna transazione poiché è una singola insert.

CREATE PROCEDURE `aggiungi\_telefono` (in numero varchar(10), in centro int, in cliente varchar(64), in tipo bool)

BEGIN

if tipo is true then

insert into `Telefono`(`Numero\_Telefono`,`Centro\_Telefono`) values(numero, centro);

else

insert into `Telefono`(`Numero\_Telefono`,`Cliente\_Telefono`) values(numero, cliente);

end if;

END

**aggiungi\_turno**

* Aggiunge un turno per un cliente;
* Nessuna transazione poiché è una singola insert.

CREATE PROCEDURE `aggiungi\_turno` (in data\_inizio date, in data\_fine date, in carica varchar(64), in centro int, in cf varchar(64))

BEGIN

insert into `Turno`(`Data\_Inizio`,`Data\_Fine`,`Carica`,`Centro\_Turno`,`Impiegato\_Turno`)

values(data\_inizio, data\_fine, carica, centro, cf);

END

**cambia\_responsabile**

* Cambia il responsabile di un centro;
* Nessuna transazione poiché è un singolo update.

CREATE PROCEDURE `cambia\_responsabile` (in centro int, in nome varchar(64))

BEGIN

update Centro set `Responsabile` = nome where `Codice\_Centro` = centro;

END

**dequalifica\_film**

* Cambia il film da nuovo a classico;
* Nessuna transazione poiché è un singolo update.

CREATE PROCEDURE `dequalifica\_film` (in titolo varchar(64), in regista varchar(64))

BEGIN

update Film set `Tipo` = 'Classic', `Costo\_Noleggio` = '1.0' where `Titolo` = titolo and `Regista` = regista;

END

**elimina\_cellulare**

* Elimina un cellulare;
* Nessuna transazione poiché è una singola delete.

CREATE PROCEDURE `elimina\_cellulare` (in numero bigint)

BEGIN

delete from `Cellulare` where `Numero\_Cellulare` = numero;

END

**elimina\_email**

* Elimina una email;
* Nessuna transazione poiché è una singola delete.

CREATE PROCEDURE `elimina\_email` (in cf varchar(64),in centro int, in indirizzo varchar(64), in tipo bool)

BEGIN

if tipo is true then

delete from `Email` where `Centro\_Email` = centro and `Indirizzo\_Email` = indirizzo;

else

delete from `Email` where `Cliente\_Email` = cf and `Indirizzo\_Email` = indirizzo;

end if;

END

**elimina\_indirizzo**

* Elimina un indirizzo;
* Nessuna transazione poiché è una singola delete.

CREATE PROCEDURE `elimina\_indirizzo` (in cf varchar(64), in via varchar(64), in numero int, in citta varchar(64))

BEGIN

delete from `Indirizzo`

where `Cliente\_Indirizzo` = cf and `Via` = via and `Numero` = numero and `Città` = citta;

END

**elimina\_telefono**

* Elimina un telefono;
* Nessuna transazione poiché è una singola delete.

CREATE PROCEDURE `elimina\_telefono` (in cf varchar(64), in centro int, in numero bigint, in tipo bool)

BEGIN

if tipo is true then

delete from `Telefono` where `Centro\_Telefono` = centro and `Numero\_Telefono` = numero;

else

delete from `Telefono` where `Cliente\_Telefono` = cf and `Numero\_Telefono` = numero;

end if;

END

**film\_disponibili\_noleggio**

* Mostra i film disponibili al noleggio per un centro;
* Livello di isolamento della transazione READ COMMITTED READ ONLY, interessa solo leggere dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `film\_disponibili\_noleggio` (in centro int)

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

select `ID\_Copia` as `Copy ID`,

`Titolo\_Copia` as `Film`,

`Regista\_Copia` as `Movie Director`,

`Anno` as `Year`,

`Settore\_Copia` as `Sector`,

`Posizione` as `Position`,

IFNULL(`Titolo\_Originale`,'') as `Remake of ..`,

IFNULL(`Regista\_Originale`,'') as `Directed by ..`

from `Copia\_Film` left join `Noleggio`

on `ID\_Copia` = `Numero\_Copia`

and `Titolo\_Copia` = `Titolo\_Noleggio`

and `Regista\_Copia` = `Regista\_Noleggio`

join `Film`

on `Titolo` = `Titolo\_Copia`

and `Regista` = `Regista\_Copia`

left join `Remake`

on `Titolo\_Copia` = `Titolo\_Remake`

and `Regista\_Copia` = `Regista\_Remake`

where `Centro\_Copia` = centro and `ID\_Noleggio` is null

order by `Titolo\_Copia`, `Regista\_Copia` asc;

commit;

END

**info\_cliente**

* Mostra le informazioni del cliente;
* Livello di isolamento della transazione READ COMMITTED READ ONLY, interessa solo leggere dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `info\_cliente` (in cliente varchar(64))

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

if not exists (select \* from `Cliente` where `CF\_Cliente` = cliente) then

signal sqlstate '45000' set message\_text = 'Client not found!';

end if;

select

`CF\_Cliente` as `Tax Code`,

`Nome\_Cliente` as `Name`,

`Cognome\_Cliente` as `Surname`,

`Data\_di\_Nascita` as `Birth Date`

from `Cliente` where `CF\_Cliente` = cliente;

select `Numero\_Cellulare` as `Cellular` from `Cellulare` where `Cliente\_Cellulare` = cliente;

select `Numero\_Telefono` as `Telephone` from `Telefono` where `Cliente\_Telefono` = cliente;

select `Città` as `City`, `Via` as `Street`, `Numero` as `House Number`

from `Indirizzo` where `Cliente\_Indirizzo` = cliente;

select `Indirizzo\_Email` as `Email/s` from `Email` where `Cliente\_Email` = cliente;

commit;

END

**login**

* Autorizza o meno l’utente;
* Nessuna transazione poiché gli utenti non possono essere modificati.

CREATE PROCEDURE `login` (in cf varchar(64), in psw varchar(45), out var\_role INT, out var\_name varchar(64))

BEGIN

declare user\_role ENUM('Impiegato','Manager');

select `Ruolo`

from `Impiegato` join `utenti` on `CF\_Impiegato` = `username`

where `username` = cf and `password` = MD5(psw) into user\_role;

if user\_role = 'Impiegato' then

set var\_role = 1;

select `Nome\_Impiegato` from `Impiegato` where `CF\_Impiegato` = cf into var\_name;

elseif user\_role = 'Manager' then

set var\_role = 2;

select `Nome\_Impiegato` from `Impiegato` where `CF\_Impiegato` = cf into var\_name;

else

set var\_role = 3;

end if;

END

**noleggia**

* Effettua un noleggio;
* Livello di isolamento della transazione READ COMMITTED, interessa lavorare su dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `noleggia` (in cliente varchar(64), in restituzione date, in centro int, in var\_film varchar(64), in var\_regista varchar(64), out days int, out c\_noleggio float, out c\_giornaliero float)

BEGIN

declare id int;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

select `ID\_Copia`

from `Copia\_Film` left join `Noleggio`

on `ID\_Copia` = `Numero\_Copia`

and `Titolo\_Copia` = `Titolo\_Noleggio`

and `Regista\_Copia` = `Regista\_Noleggio`

where `Centro\_Copia` = centro

and `Titolo\_Copia` = var\_film

and `Regista\_Copia` = var\_regista

and `ID\_Noleggio` is null

LIMIT 1 into id;

if id is null then

signal sqlstate '45000' set message\_text = 'No film copy found, check the available film list!';

else

insert into `Noleggio`(`Data\_Restituzione`,`Numero\_Copia`,`Titolo\_Noleggio`,`Regista\_Noleggio`,`Cliente\_Noleggio`)

values (restituzione, id, var\_film, var\_regista, cliente);

select datediff(restituzione,curdate())+1,`Costo\_Noleggio`,`Costo\_Giornaliero` from `Film`

where `Titolo` = var\_film and `Regista` = var\_regista into days, c\_noleggio, c\_giornaliero;

end if;

commit;

END

**noleggi\_cliente**

* Visualizza i noleggi attivi del cliente;
* Livello di isolamento della transazione READ COMMITTED READ ONLY, interessa solo leggere dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `noleggi\_cliente` (in cliente varchar(64))

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

select `ID\_Noleggio` as `Rent ID`,

`Titolo\_Noleggio` as `Film`,

`Regista\_Noleggio` as `Movie Director`,

`Data\_restituzione` as `Return Date`

from `Noleggio` where `Cliente\_Noleggio` = cliente;

commit;

END

**noleggi\_scaduti**

* Visualizza i noleggi scaduti del cliente;
* Livello di isolamento della transazione READ COMMITTED READ ONLY, interessa solo leggere dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `noleggi\_scaduti` (in centro int)

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

select `ID\_Noleggio` as `Rent ID`,

`Titolo\_Noleggio` as `Film`,

`Data\_Restituzione` as `Return Date`,

`Cliente\_Noleggio` as `Client`

from `Noleggio` join `Copia\_Film`

on `Numero\_Copia` = `ID\_Copia`

and `Titolo\_Noleggio` = `Titolo\_Copia`

and `Regista\_Noleggio` = `Regista\_Copia`

where `Data\_Restituzione` < curdate()

and `Centro\_Copia` = centro

order by `Data\_Restituzione` asc;

commit;

END

**report\_annuale**

* Visualizza il report annuale dell’impiegato;
* Livello di isolamento della transazione SERIALIZABLE, interessa lavorare su dati confermati, fare più select sulla stessa tabella con diversi range e non avere dati inconsistenti, avere inserimenti fantasma sarebbe un problema, quindi, serve il massimo livello di isolamento.

CREATE PROCEDURE `report\_annuale` (in cf varchar(64), in anno year, out ore int)

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

drop temporary table if exists `intervallo`;

create temporary table

`intervallo`(`Center` int,`Role` varchar(12),`First Day` date,`Last Day` date,`Tot. Day/s` int, `Tot. Hour/s` int);

set transaction isolation level serializable;

start transaction;

if not exists (select \* from `Impiegato` where `CF\_Impiegato` = cf) then

signal sqlstate '45000' set message\_text = 'Employee not found!';

end if;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

`Data\_Inizio`,

`Data\_Fine`,

datediff(`Data\_Fine`,`Data\_Inizio`)+1,

(datediff(`Data\_Fine`,`Data\_Inizio`)+1)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and year(`Data\_Inizio`) = anno

and year(`Data\_Fine`) = anno;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`, `Data\_Inizio`,

last\_day(`Data\_Inizio`),

datediff(last\_day(`Data\_Inizio`),

`Data\_Inizio`)+1,

(datediff(last\_day(`Data\_Inizio`),`Data\_Inizio`)+1)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and year(`Data\_Inizio`) = anno

and year(`Data\_Fine`) <> anno;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

ADDDATE(LAST\_DAY(SUBDATE(`Data\_Fine`, INTERVAL 1 MONTH)), 1),

`Data\_Fine`,

day(`Data\_Fine`),

day(`Data\_Fine`)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and year(`Data\_Fine`) = anno

and year(`Data\_Inizio`) <> anno;

select \* from `intervallo` order by `First Day` asc;

select sum(`Tot. Hour/s`) from `intervallo` into ore;

commit;

drop temporary table `intervallo`;

END

**report\_mensile**

* Visualizza il report mensile dell’impiegato;
* Livello di isolamento della transazione SERIALIZABLE, interessa lavorare su dati confermati, fare più select sulla stessa tabella con diversi range e non avere dati inconsistenti, avere inserimenti fantasma sarebbe un problema, quindi, serve il massimo livello di isolamento.

CREATE PROCEDURE `report\_mensile`(in cf varchar(64), in mese int, in anno year, out var\_ore int)

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

drop temporary table if exists `intervallo`;

create temporary table

`intervallo`(`Center` int,`Role` varchar(12),`First Day` date,`Last Day` date,`Tot. Day/s` int, `Tot. Hour/s` int);

set transaction isolation level serializable;

start transaction;

if not exists (select \* from `Impiegato` where `CF\_Impiegato` = cf) then

signal sqlstate '45000' set message\_text = 'Employee not found!';

end if;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

`Data\_Inizio`,

`Data\_Fine`,

datediff(`Data\_Fine`,`Data\_Inizio`)+1,

(datediff(`Data\_Fine`,`Data\_Inizio`)+1)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and month(`Data\_Inizio`) = mese

and year(`Data\_Inizio`) = anno

and month(`Data\_Fine`) = mese

and year(`Data\_Fine`) = anno;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`, `Data\_Inizio`,

last\_day(`Data\_Inizio`),

datediff(last\_day(`Data\_Inizio`),

`Data\_Inizio`)+1,

(datediff(last\_day(`Data\_Inizio`),`Data\_Inizio`)+1)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and month(`Data\_Inizio`) = mese

and year(`Data\_Inizio`) = anno

and month(`Data\_Fine`) <> mese

and year(`Data\_Fine`) = anno;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

ADDDATE(LAST\_DAY(SUBDATE(`Data\_Fine`, INTERVAL 1 MONTH)), 1),

`Data\_Fine`,

day(`Data\_Fine`),

day(`Data\_Fine`)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and month(`Data\_Fine`) = mese

and year(`Data\_Fine`) = anno

and month(`Data\_Inizio`) <> mese

and year(`Data\_Inizio`) = anno;

select \* from `intervallo` order by `First Day` asc;

select sum(`Tot. Hour/s`) from `intervallo` into ore;

commit;

drop temporary table `intervallo`;

END

**restituisci\_film**

* Restituisce il film noleggiato;
* Livello di isolamento della transazione READ COMMITTED, interessa lavorare su dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `restituisci\_film` (in id int, out prezzo\_extra int)

BEGIN

declare var\_data date;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

set prezzo\_extra = 0;

select `Data\_Restituzione` from `Noleggio` where `ID\_Noleggio` = id into var\_data;

if (var\_data < curdate()) then

set prezzo\_extra = datediff(curdate(),var\_data);

end if;

delete from `Noleggio` where `ID\_Noleggio` = id;

commit;

END

**turni\_del\_giorno**

* Visualizza i turni di certo giorno in un centro;
* Livello di isolamento della transazione READ COMMITTED READ ONLY, interessa solo leggere dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `turni\_del\_giorno` (in giorno date, in centro int)

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

select `Impiegato\_Turno`as `Tax Code`, `Nome\_Impiegato`as `Name`,`Carica` as `Role`

from `Turno` join `Impiegato` on `Impiegato\_Turno` = `CF\_Impiegato`

where `Data\_Inizio` <= giorno

and `Data\_Fine` >= giorno

and `Centro\_Turno` = centro;

commit;

END

**visualizza\_attori\_film**

* Visualizza gli attori del film;
* Livello di isolamento della transazione READ COMMITTED READ ONLY, interessa solo leggere dati confermati, la select viene eseguita un’unica volta quindi il lock di lettura può essere rilasciato subito.

CREATE PROCEDURE `visualizza\_attori\_film` (in titolo varchar(64), in regista varchar(64))

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

select `Nome\_Attore\_Recita` as `Name`,

`Cognome\_Attore\_Recita` as `Surname`

from `Recita` where `Titolo\_Recita` = titolo and `Regista\_Recita` = regista;

commit;

END

# Appendice: Implementazione

## Codice SQL per instanziare il database

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

-- Schema noleggio-film

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

CREATE SCHEMA IF NOT EXISTS `noleggio-film` DEFAULT CHARACTER SET utf8 ;

USE `noleggio-film` ;

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

-- Table `noleggio-film`.`Impiegato`

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

DROP TABLE IF EXISTS `noleggio-film`.`Impiegato` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Impiegato` (

`CF\_Impiegato` VARCHAR(64) NOT NULL,

`Nome\_Impiegato` VARCHAR(64) NOT NULL,

`Titolo\_di\_studio` VARCHAR(64) NOT NULL,

`Telefono` VARCHAR(64) NOT NULL,

`Ruolo` ENUM('Impiegato', 'Manager') NOT NULL,

PRIMARY KEY (`CF\_Impiegato`))

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Centro`

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

DROP TABLE IF EXISTS `noleggio-film`.`Centro` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Centro` (

`Codice\_Centro` INT NOT NULL AUTO\_INCREMENT,

`Responsabile` VARCHAR(64) NOT NULL,

`Indirizzo\_Centro` VARCHAR(64) NOT NULL,

`Numero\_Civico` INT NOT NULL,

`Sede` VARCHAR(64) NOT NULL,

PRIMARY KEY (`Codice\_Centro`))

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Turno`

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

DROP TABLE IF EXISTS `noleggio-film`.`Turno` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Turno` (

`iD\_Turno` INT NOT NULL AUTO\_INCREMENT,

`Data\_Inizio` DATE NOT NULL,

`Data\_Fine` DATE NOT NULL,

`Carica` ENUM('Cashier', 'Clerk', 'Warehouseman') NOT NULL,

`Centro\_Turno` INT NOT NULL,

`Impiegato\_Turno` VARCHAR(64) NOT NULL,

PRIMARY KEY (`iD\_Turno`),

INDEX `fk\_Turno\_Centro1\_idx` (`Centro\_Turno` ASC) VISIBLE,

INDEX `fk\_Turno\_Impiegato1\_idx` (`Impiegato\_Turno` ASC) VISIBLE,

INDEX `idx\_turno\_centro\_impiegato` (`Centro\_Turno` ASC, `Impiegato\_Turno` ASC) VISIBLE,

CONSTRAINT `fk\_Turno\_Centro1`

FOREIGN KEY (`Centro\_Turno`)

REFERENCES `noleggio-film`.`Centro` (`Codice\_Centro`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Turno\_Impiegato1`

FOREIGN KEY (`Impiegato\_Turno`)

REFERENCES `noleggio-film`.`Impiegato` (`CF\_Impiegato`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Settore`

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

DROP TABLE IF EXISTS `noleggio-film`.`Settore` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Settore` (

`Centro\_Settore` INT NOT NULL,

`Codice\_Settore` INT NOT NULL,

PRIMARY KEY (`Centro\_Settore`, `Codice\_Settore`),

INDEX `fk\_Settore\_Centro1\_idx` (`Centro\_Settore` ASC) VISIBLE,

CONSTRAINT `fk\_Settore\_Centro1`

FOREIGN KEY (`Centro\_Settore`)

REFERENCES `noleggio-film`.`Centro` (`Codice\_Centro`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Film`

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

DROP TABLE IF EXISTS `noleggio-film`.`Film` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Film` (

`Titolo` VARCHAR(64) NOT NULL,

`Regista` VARCHAR(64) NOT NULL,

`Anno` YEAR NOT NULL,

`Numero\_Copie` INT NOT NULL,

`Costo\_Noleggio` FLOAT NOT NULL,

`Costo\_Giornaliero` FLOAT NOT NULL,

`Tipo` ENUM('Classic', 'New') NOT NULL,

PRIMARY KEY (`Titolo`, `Regista`))

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Copia\_Film`

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

DROP TABLE IF EXISTS `noleggio-film`.`Copia\_Film` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Copia\_Film` (

`ID\_Copia` INT NOT NULL,

`Titolo\_Copia` VARCHAR(64) NOT NULL,

`Regista\_Copia` VARCHAR(64) NOT NULL,

`Centro\_Copia` INT NOT NULL,

`Settore\_Copia` INT NOT NULL,

`Posizione` INT NOT NULL,

PRIMARY KEY (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`),

INDEX `fk\_Copia\_Film\_Film1\_idx` (`Titolo\_Copia` ASC, `Regista\_Copia` ASC) VISIBLE,

INDEX `fk\_Copia\_Film\_Settore1\_idx` (`Centro\_Copia` ASC, `Settore\_Copia` ASC) VISIBLE,

INDEX `idx\_copia\_centro\_film` (`Centro\_Copia` ASC, `Titolo\_Copia` ASC) VISIBLE,

CONSTRAINT `fk\_Copia\_Film\_Film1`

FOREIGN KEY (`Titolo\_Copia` , `Regista\_Copia`)

REFERENCES `noleggio-film`.`Film` (`Titolo` , `Regista`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Copia\_Film\_Settore1`

FOREIGN KEY (`Centro\_Copia` , `Settore\_Copia`)

REFERENCES `noleggio-film`.`Settore` (`Centro\_Settore` , `Codice\_Settore`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Remake`

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

DROP TABLE IF EXISTS `noleggio-film`.`Remake` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Remake` (

`Titolo\_Remake` VARCHAR(64) NOT NULL,

`Regista\_Remake` VARCHAR(64) NOT NULL,

`Titolo\_Originale` VARCHAR(64) NOT NULL,

`Regista\_Originale` VARCHAR(64) NOT NULL,

PRIMARY KEY (`Titolo\_Remake`, `Regista\_Remake`),

INDEX `fk\_Remake\_Film2\_idx` (`Titolo\_Originale` ASC, `Regista\_Originale` ASC) VISIBLE,

CONSTRAINT `fk\_Remake\_Film1`

FOREIGN KEY (`Titolo\_Remake` , `Regista\_Remake`)

REFERENCES `noleggio-film`.`Film` (`Titolo` , `Regista`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Remake\_Film2`

FOREIGN KEY (`Titolo\_Originale` , `Regista\_Originale`)

REFERENCES `noleggio-film`.`Film` (`Titolo` , `Regista`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Attore`

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

DROP TABLE IF EXISTS `noleggio-film`.`Attore` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Attore` (

`Nome\_Attore` VARCHAR(64) NOT NULL,

`Cognome\_Attore` VARCHAR(64) NOT NULL,

PRIMARY KEY (`Nome\_Attore`, `Cognome\_Attore`))

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Recita`

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

DROP TABLE IF EXISTS `noleggio-film`.`Recita` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Recita` (

`Titolo\_Recita` VARCHAR(64) NOT NULL,

`Regista\_Recita` VARCHAR(64) NOT NULL,

`Nome\_Attore\_Recita` VARCHAR(64) NOT NULL,

`Cognome\_Attore\_Recita` VARCHAR(64) NOT NULL,

PRIMARY KEY (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`),

INDEX `fk\_Attore\_has\_Film\_Film1\_idx` (`Titolo\_Recita` ASC, `Regista\_Recita` ASC) VISIBLE,

INDEX `fk\_Attore\_has\_Film\_Attore1\_idx` (`Nome\_Attore\_Recita` ASC, `Cognome\_Attore\_Recita` ASC) VISIBLE,

CONSTRAINT `fk\_Attore\_has\_Film\_Attore1`

FOREIGN KEY (`Nome\_Attore\_Recita` , `Cognome\_Attore\_Recita`)

REFERENCES `noleggio-film`.`Attore` (`Nome\_Attore` , `Cognome\_Attore`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Attore\_has\_Film\_Film1`

FOREIGN KEY (`Titolo\_Recita` , `Regista\_Recita`)

REFERENCES `noleggio-film`.`Film` (`Titolo` , `Regista`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Cliente`

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

DROP TABLE IF EXISTS `noleggio-film`.`Cliente` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Cliente` (

`CF\_Cliente` VARCHAR(64) NOT NULL,

`Nome\_Cliente` VARCHAR(64) NOT NULL,

`Cognome\_Cliente` VARCHAR(64) NOT NULL,

`Data\_di\_Nascita` DATE NOT NULL,

PRIMARY KEY (`CF\_Cliente`))

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Noleggio`

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

DROP TABLE IF EXISTS `noleggio-film`.`Noleggio` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Noleggio` (

`ID\_Noleggio` INT NOT NULL AUTO\_INCREMENT,

`Data\_Restituzione` DATE NOT NULL,

`Numero\_Copia` INT NOT NULL,

`Titolo\_Noleggio` VARCHAR(64) NOT NULL,

`Regista\_Noleggio` VARCHAR(64) NOT NULL,

`Cliente\_Noleggio` VARCHAR(64) NOT NULL,

PRIMARY KEY (`ID\_Noleggio`),

INDEX `fk\_Noleggio\_Copia\_Film1\_idx` (`Numero\_Copia` ASC, `Titolo\_Noleggio` ASC, `Regista\_Noleggio` ASC) VISIBLE,

INDEX `fk\_Noleggio\_Cliente1\_idx` (`Cliente\_Noleggio` ASC) VISIBLE,

INDEX `idx\_noleggio\_cliente` (`Cliente\_Noleggio` ASC) VISIBLE,

CONSTRAINT `fk\_Noleggio\_Copia\_Film1`

FOREIGN KEY (`Numero\_Copia` , `Titolo\_Noleggio` , `Regista\_Noleggio`)

REFERENCES `noleggio-film`.`Copia\_Film` (`ID\_Copia` , `Titolo\_Copia` , `Regista\_Copia`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Noleggio\_Cliente1`

FOREIGN KEY (`Cliente\_Noleggio`)

REFERENCES `noleggio-film`.`Cliente` (`CF\_Cliente`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Indirizzo`

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

DROP TABLE IF EXISTS `noleggio-film`.`Indirizzo` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Indirizzo` (

`Via` VARCHAR(64) NOT NULL,

`Numero` INT NOT NULL,

`Città` VARCHAR(64) NOT NULL,

`Cliente\_Indirizzo` VARCHAR(64) NOT NULL,

PRIMARY KEY (`Via`, `Numero`, `Città`),

INDEX `fk\_Indirizzo\_Cliente1\_idx` (`Cliente\_Indirizzo` ASC) VISIBLE,

CONSTRAINT `fk\_Indirizzo\_Cliente1`

FOREIGN KEY (`Cliente\_Indirizzo`)

REFERENCES `noleggio-film`.`Cliente` (`CF\_Cliente`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Email`

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

DROP TABLE IF EXISTS `noleggio-film`.`Email` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Email` (

`Indirizzo\_Email` VARCHAR(64) NOT NULL,

`Centro\_Email` INT NULL,

`Cliente\_Email` VARCHAR(64) NULL,

PRIMARY KEY (`Indirizzo\_Email`),

INDEX `fk\_Email\_Centro1\_idx` (`Centro\_Email` ASC) VISIBLE,

INDEX `fk\_Email\_Cliente1\_idx` (`Cliente\_Email` ASC) VISIBLE,

CONSTRAINT `fk\_Email\_Centro1`

FOREIGN KEY (`Centro\_Email`)

REFERENCES `noleggio-film`.`Centro` (`Codice\_Centro`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Email\_Cliente1`

FOREIGN KEY (`Cliente\_Email`)

REFERENCES `noleggio-film`.`Cliente` (`CF\_Cliente`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Telefono`

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

DROP TABLE IF EXISTS `noleggio-film`.`Telefono` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Telefono` (

`Numero\_Telefono` VARCHAR(64) NOT NULL,

`Centro\_Telefono` INT NULL,

`Cliente\_Telefono` VARCHAR(64) NULL,

PRIMARY KEY (`Numero\_Telefono`),

INDEX `fk\_Telefono\_Centro1\_idx` (`Centro\_Telefono` ASC) VISIBLE,

INDEX `fk\_Telefono\_Cliente1\_idx` (`Cliente\_Telefono` ASC) VISIBLE,

CONSTRAINT `fk\_Telefono\_Centro1`

FOREIGN KEY (`Centro\_Telefono`)

REFERENCES `noleggio-film`.`Centro` (`Codice\_Centro`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Telefono\_Cliente1`

FOREIGN KEY (`Cliente\_Telefono`)

REFERENCES `noleggio-film`.`Cliente` (`CF\_Cliente`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`Cellulare`

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

DROP TABLE IF EXISTS `noleggio-film`.`Cellulare` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`Cellulare` (

`Numero\_Cellulare` VARCHAR(64) NOT NULL,

`Cliente\_Cellulare` VARCHAR(64) NOT NULL,

PRIMARY KEY (`Numero\_Cellulare`),

INDEX `fk\_Cellulare\_Cliente1\_idx` (`Cliente\_Cellulare` ASC) VISIBLE,

CONSTRAINT `fk\_Cellulare\_Cliente1`

FOREIGN KEY (`Cliente\_Cellulare`)

REFERENCES `noleggio-film`.`Cliente` (`CF\_Cliente`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `noleggio-film`.`utenti`

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

DROP TABLE IF EXISTS `noleggio-film`.`utenti` ;

CREATE TABLE IF NOT EXISTS `noleggio-film`.`utenti` (

`username` VARCHAR(64) NOT NULL,

`password` CHAR(32) NOT NULL,

INDEX `fk\_utenti\_Impiegato1\_idx` (`username` ASC) VISIBLE,

PRIMARY KEY (`username`),

CONSTRAINT `fk\_utenti\_Impiegato1`

FOREIGN KEY (`username`)

REFERENCES `noleggio-film`.`Impiegato` (`CF\_Impiegato`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

SET SQL\_MODE = '';

DROP USER IF EXISTS manager;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'manager' IDENTIFIED BY 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Impiegato` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Centro` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Turno` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Settore` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Film` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Copia\_Film` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Remake` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Attore` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Recita` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Cliente` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Noleggio` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Indirizzo` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Email` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Telefono` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`Cellulare` TO 'manager';

GRANT ALL ON TABLE `noleggio-film`.`utenti` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_turno` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`turni\_del\_giorno` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`report\_mensile` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`report\_annuale` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_impiegato` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`cambia\_responsabile` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_film` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_attore\_recita` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_copia` TO 'manager';

GRANT EXECUTE ON procedure `noleggio-film`.`dequalifica\_film` TO 'manager';

SET SQL\_MODE = '';

DROP USER IF EXISTS impiegato;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'impiegato' IDENTIFIED BY 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Impiegato` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Centro` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Turno` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Settore` TO 'impiegato';

GRANT UPDATE, SELECT ON TABLE `noleggio-film`.`Film` TO 'impiegato';

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE `noleggio-film`.`Copia\_Film` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Remake` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Attore` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`Recita` TO 'impiegato';

GRANT INSERT, UPDATE, SELECT ON TABLE `noleggio-film`.`Cliente` TO 'impiegato';

GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE `noleggio-film`.`Noleggio` TO 'impiegato';

GRANT SELECT, UPDATE, INSERT ON TABLE `noleggio-film`.`Indirizzo` TO 'impiegato';

GRANT SELECT, INSERT, UPDATE ON TABLE `noleggio-film`.`Email` TO 'impiegato';

GRANT SELECT, INSERT, UPDATE ON TABLE `noleggio-film`.`Telefono` TO 'impiegato';

GRANT UPDATE, SELECT, INSERT ON TABLE `noleggio-film`.`Cellulare` TO 'impiegato';

GRANT SELECT ON TABLE `noleggio-film`.`utenti` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`film\_disponibili\_noleggio` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`noleggia` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`restituisci\_film` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`noleggi\_scaduti` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`noleggi\_cliente` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`info\_cliente` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_cliente` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_cellulare` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_telefono` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_indirizzo` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`aggiungi\_email` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`elimina\_cellulare` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`elimina\_telefono` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`elimina\_indirizzo` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`elimina\_email` TO 'impiegato';

GRANT EXECUTE ON procedure `noleggio-film`.`visualizza\_attori\_film` TO 'impiegato';

SET SQL\_MODE = '';

DROP USER IF EXISTS login;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'login' IDENTIFIED BY 'login';

GRANT SELECT ON TABLE `noleggio-film`.`Impiegato` TO 'login';

GRANT SELECT ON TABLE `noleggio-film`.`utenti` TO 'login';

GRANT EXECUTE ON procedure `noleggio-film`.`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 `noleggio-film`.`Impiegato`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('LFSGGQ51L23I209P', 'Lilla', 'Diploma scuola superiore', '3063441097', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('TCCBRJ60M63B901D', 'Sabatino', 'Diploma scuola superiore', '963659956', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('RDUKTV70H53B099U', 'Loretta', 'Diploma universitario', '8585404637', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('PHFHDH29M44D775Z', 'Pomponio', 'Laurea di primo livello', '901158444', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('VNTLMN53H64D341L', 'Paride', 'Diploma scuola superiore', '809442152', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('SZRFVG51L10F922P', 'Manuel', 'Diploma scuola superiore', '8543492592', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('MTGCGA71A21E296T', 'Simonetta', 'Diploma scuola superiore', '3480206406', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('ZDBSNS49E65D802Z', 'Rocco', 'Diploma universitario', '3337430404', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('ZDGBFD71C54E658I', 'Tonino', 'Laurea di primo livello', '3333214552', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('TCTZSU32P45H336S', 'Nico', 'Diploma scuola superiore', '3480965031', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('PMMGMT59E43H407V', 'Mercedes', 'Diploma scuola superiore', '3334989814', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('WWVZMZ42B22E382I', 'Ida', 'Diploma scuola superiore', '3338352301', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('TRRRHR44S07F826R', 'Alfredo', 'Diploma scuola superiore', '3333832072', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('SJLMYF59H06F556C', 'Erika', 'Diploma scuola superiore', '3332318840', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('LDHDZP66S17I849M', 'Matteo', 'Diploma scuola superiore', '3295456513', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('CTXHJH90D01B207F', 'Salvatore', 'Diploma scuola superiore', '3296070108', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('LSNSND95E70G381R', 'Allegra', 'Diploma scuola superiore', '3480008086', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('NCGMWM61E19I417J', 'Ottone', 'Diploma scuola superiore', '3487990798', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('TXASHI84R46H343A', 'Simonetta', 'Laurea di primo livello', '3295909160', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('DFVSNS99D46C696T', 'Ramona', 'Diploma scuola superiore', '3294264123', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('ZFCVMT32S64L354D', 'Salvatore', 'Diploma scuola superiore', '3297933079', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('KZALNV44B14B493Z', 'Fabia', 'Diploma scuola superiore', '3488916306', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('JPNXZE37A05L551H', 'Eugenia', 'Diploma universitario', '3297689779', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('HCTGSG71R09H303V', 'Nadia', 'Laurea di primo livello', '3296017720', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('NVRZVM85B48B760M', 'Marcella', 'Diploma scuola superiore', '3295776004', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('VKYTMS69E12G364K', 'Gianni', 'Diploma scuola superiore', '3299800508', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('MCGZYZ61B52E106Y', 'Liana', 'Diploma universitario', '3276726326', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('YVWVFJ62D46D374X', 'Stefano', 'Laurea di primo livello', '3279386199', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('NXWFGG27T04A745E', 'Ilaria', 'Laurea di primo livello', '3274989264', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('CXECDP70A18D869S', 'Patrizio', 'Diploma universitario', '3279637887', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('QZGYWC69P24E746Q', 'Amedeo', 'Diploma scuola superiore', '3276019273', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('PNUGYA73A17I964P', 'Lina', 'Laurea di primo livello', '3271182174', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('LHCDFG87D47H046P', 'Liberto', 'Diploma scuola superiore', '3271406437', 'Impiegato');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('manager', 'Manager', 'Laurea di primo livello', '3271406439', 'Manager');

INSERT INTO `noleggio-film`.`Impiegato` (`CF\_Impiegato`, `Nome\_Impiegato`, `Titolo\_di\_studio`, `Telefono`, `Ruolo`) VALUES ('impiegato', 'Impiegato', 'Diploma scuola superiore', '3271406431', 'Impiegato');

COMMIT;

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

-- Data for table `noleggio-film`.`Centro`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Centro` (`Codice\_Centro`, `Responsabile`, `Indirizzo\_Centro`, `Numero\_Civico`, `Sede`) VALUES (1, 'Manuel', 'Via Calabria', 23, 'Roma');

INSERT INTO `noleggio-film`.`Centro` (`Codice\_Centro`, `Responsabile`, `Indirizzo\_Centro`, `Numero\_Civico`, `Sede`) VALUES (2, 'Simonetta', 'Via Sicilia', 224, 'Milano');

INSERT INTO `noleggio-film`.`Centro` (`Codice\_Centro`, `Responsabile`, `Indirizzo\_Centro`, `Numero\_Civico`, `Sede`) VALUES (3, 'Ida', 'Via Adige', 4, 'Napoli');

INSERT INTO `noleggio-film`.`Centro` (`Codice\_Centro`, `Responsabile`, `Indirizzo\_Centro`, `Numero\_Civico`, `Sede`) VALUES (4, 'Alfredo', 'Via Lecce', 45, 'Firenze');

INSERT INTO `noleggio-film`.`Centro` (`Codice\_Centro`, `Responsabile`, `Indirizzo\_Centro`, `Numero\_Civico`, `Sede`) VALUES (5, 'Erika', 'Via FIrenze', 79, 'Genova');

COMMIT;

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

-- Data for table `noleggio-film`.`Turno`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (1, '2020-03-25', '2020-03-30', 'Cashier', 1, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (2, '2020-01-13', '2020-01-13', 'Warehouseman', 5, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (3, '2020-02-08', '2020-02-18', 'Clerk', 1, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (4, '2020-04-05', '2020-04-09', 'Warehouseman', 5, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (5, '2020-04-18', '2020-04-22', 'Clerk', 1, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (6, '2020-06-16', '2020-06-26', 'Cashier', 2, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (7, '2020-06-18', '2020-06-22', 'Warehouseman', 1, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (8, '2020-07-03', '2020-07-07', 'Clerk', 3, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (9, '2020-08-02', '2020-08-03', 'Cashier', 4, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (10, '2020-08-18', '2020-08-19', 'Warehouseman', 5, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (11, '2020-08-22', '2020-08-25', 'Clerk', 1, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (12, '2020-09-07', '2020-09-10', 'Cashier', 4, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (13, '2020-09-15', '2020-09-18', 'Warehouseman', 3, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (14, '2020-10-04', '2020-10-14', 'Clerk', 4, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (15, '2020-11-05', '2020-11-25', 'Cashier', 3, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (16, '2020-12-06', '2020-12-16', 'Warehouseman', 4, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (17, '2021-01-03', '2021-01-06', 'Clerk', 2, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (18, '2021-01-14', '2021-01-16', 'Cashier', 5, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (19, '2021-01-29', '2021-01-29', 'Warehouseman', 3, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (20, '2021-02-09', '2021-02-11', 'Clerk', 1, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (21, '2021-02-26', '2021-02-28', 'Cashier', 2, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (22, '2021-03-09', '2021-03-13', 'Warehouseman', 5, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (23, '2021-04-14', '2021-04-24', 'Clerk', 2, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (24, '2021-07-20', '2021-07-21', 'Cashier', 3, 'impiegato');

INSERT INTO `noleggio-film`.`Turno` (`iD\_Turno`, `Data\_Inizio`, `Data\_Fine`, `Carica`, `Centro\_Turno`, `Impiegato\_Turno`) VALUES (25, '2021-08-10', '2021-08-18', 'Warehouseman', 1, 'impiegato');

COMMIT;

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

-- Data for table `noleggio-film`.`Settore`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (1, 1);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (1, 2);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (1, 3);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (1, 5);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (1, 4);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (2, 1);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (2, 2);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (2, 3);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (2, 4);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (2, 5);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (3, 1);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (3, 2);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (3, 3);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (3, 4);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (3, 5);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (4, 1);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (4, 2);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (4, 3);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (5, 1);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (5, 2);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (5, 3);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (4, 5);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (4, 4);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (5, 4);

INSERT INTO `noleggio-film`.`Settore` (`Centro\_Settore`, `Codice\_Settore`) VALUES (5, 5);

COMMIT;

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

-- Data for table `noleggio-film`.`Film`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Il padrino', 'Francis Ford Coppola', 1972, 5, 1, 0.5, 'Classic');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('I predatori dell\'arca perduta', 'Steven Spielberg', 1981, 5, 1, 0.5, 'Classic');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('L\'Impero colpisce ancora', 'Irvin Kershner', 1980, 5, 1, 0.5, 'Classic');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Alien', 'Ridley Scott', 1979, 5, 1, 0.5, 'Classic');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Lo squalo', 'Steven Spielberg', 1975, 5, 1, 0.5, 'Classic');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Iron Man', 'Jon Favreau', 2008, 5, 3, 1, 'New');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Iron Man 2', 'Jon Favreau', 2010, 5, 3, 1, 'New');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Iron Man 3', 'Shane Black', 2013, 5, 3, 1, 'New');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('The Avengers', 'Joe Johnston', 2012, 5, 3, 1, 'New');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Guardiani della Galassia', 'James Gunn', 2014, 5, 3, 1, 'New');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Scarface', 'Brian De Palma', 1983, 5, 1, 0.5, 'Classic');

INSERT INTO `noleggio-film`.`Film` (`Titolo`, `Regista`, `Anno`, `Numero\_Copie`, `Costo\_Noleggio`, `Costo\_Giornaliero`, `Tipo`) VALUES ('Scarface - Lo sfregiato', 'Howard Hawks', 1932, 0, 1, 0.5, 'Classic');

COMMIT;

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

-- Data for table `noleggio-film`.`Copia\_Film`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Il padrino', 'Francis Ford Coppola', 1, 1, 4);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Il padrino', 'Francis Ford Coppola', 1, 1, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Il padrino', 'Francis Ford Coppola', 1, 2, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Il padrino', 'Francis Ford Coppola', 2, 2, 5);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Il padrino', 'Francis Ford Coppola', 3, 2, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'I predatori dell\'arca perduta', 'Steven Spielberg', 1, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'I predatori dell\'arca perduta', 'Steven Spielberg', 1, 2, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'I predatori dell\'arca perduta', 'Steven Spielberg', 4, 4, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'I predatori dell\'arca perduta', 'Steven Spielberg', 4, 4, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'I predatori dell\'arca perduta', 'Steven Spielberg', 2, 3, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'L\'Impero colpisce ancora', 'Irvin Kershner', 1, 4, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'L\'Impero colpisce ancora', 'Irvin Kershner', 3, 2, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'L\'Impero colpisce ancora', 'Irvin Kershner', 3, 4, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'L\'Impero colpisce ancora', 'Irvin Kershner', 2, 3, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'L\'Impero colpisce ancora', 'Irvin Kershner', 5, 4, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Alien', 'Ridley Scott', 1, 4, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Alien', 'Ridley Scott', 1, 2, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Alien', 'Ridley Scott', 5, 2, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Alien', 'Ridley Scott', 5, 2, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Alien', 'Ridley Scott', 4, 1, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Lo squalo', 'Steven Spielberg', 1, 1, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Lo squalo', 'Steven Spielberg', 1, 4, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Lo squalo', 'Steven Spielberg', 1, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Lo squalo', 'Steven Spielberg', 5, 1, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Lo squalo', 'Steven Spielberg', 4, 2, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Iron Man', 'Jon Favreau', 1, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Iron Man', 'Jon Favreau', 1, 4, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Iron Man', 'Jon Favreau', 1, 2, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Iron Man', 'Jon Favreau', 1, 1, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Iron Man', 'Jon Favreau', 4, 4, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Iron Man 2', 'Jon Favreau', 1, 4, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Iron Man 2', 'Jon Favreau', 3, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Iron Man 2', 'Jon Favreau', 2, 5, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Iron Man 2', 'Jon Favreau', 5, 1, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Iron Man 2', 'Jon Favreau', 4, 2, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Iron Man 3', 'Shane Black', 3, 1, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Iron Man 3', 'Shane Black', 1, 5, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Iron Man 3', 'Shane Black', 1, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Iron Man 3', 'Shane Black', 5, 5, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Iron Man 3', 'Shane Black', 3, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'The Avengers', 'Joe Johnston', 1, 5, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'The Avengers', 'Joe Johnston', 1, 3, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'The Avengers', 'Joe Johnston', 1, 2, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'The Avengers', 'Joe Johnston', 1, 1, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'The Avengers', 'Joe Johnston', 2, 1, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Guardiani della Galassia', 'James Gunn', 1, 3, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Guardiani della Galassia', 'James Gunn', 1, 3, 5);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Guardiani della Galassia', 'James Gunn', 3, 4, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Guardiani della Galassia', 'James Gunn', 4, 4, 2);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Guardiani della Galassia', 'James Gunn', 2, 3, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (1, 'Scarface', 'Brian De Palma', 1, 3, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (2, 'Scarface', 'Brian De Palma', 2, 3, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (3, 'Scarface', 'Brian De Palma', 3, 2, 1);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (4, 'Scarface', 'Brian De Palma', 4, 2, 3);

INSERT INTO `noleggio-film`.`Copia\_Film` (`ID\_Copia`, `Titolo\_Copia`, `Regista\_Copia`, `Centro\_Copia`, `Settore\_Copia`, `Posizione`) VALUES (5, 'Scarface', 'Brian De Palma', 5, 3, 5);

COMMIT;

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

-- Data for table `noleggio-film`.`Remake`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Remake` (`Titolo\_Remake`, `Regista\_Remake`, `Titolo\_Originale`, `Regista\_Originale`) VALUES ('Scarface', 'Brian De Palma', 'Scarface - Lo sfregiato', 'Howard Hawks');

COMMIT;

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

-- Data for table `noleggio-film`.`Attore`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Marlon', 'Brando');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Alfredo James', 'Pacino');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Harrison', 'Ford');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Kaen', 'Allen');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Mark', 'Hamill');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Carrie', 'Fisher');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Sigourney', 'Weaver');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('John', 'Hurt');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Roy', 'Scheider');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Robert', 'Shaw');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Robert', 'Downey Jr');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Jon', 'Favreau');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Chris', 'Pratt');

INSERT INTO `noleggio-film`.`Attore` (`Nome\_Attore`, `Cognome\_Attore`) VALUES ('Dave', 'Bautista');

COMMIT;

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

-- Data for table `noleggio-film`.`Recita`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Il padrino', 'Francis Ford Coppola', 'Marlon', 'Brando');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Il padrino', 'Francis Ford Coppola', 'Alfredo James', 'Pacino');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('I predatori dell\'arca perduta', 'Steven Spielberg', 'Harrison', 'Ford');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('I predatori dell\'arca perduta', 'Steven Spielberg', 'Kaen', 'Allen');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('L\'Impero colpisce ancora', 'Irvin Kershner', 'Mark', 'Hamill');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('L\'Impero colpisce ancora', 'Irvin Kershner', 'Carrie', 'Fisher');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Alien', 'Ridley Scott', 'Sigourney', 'Weaver');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Alien', 'Ridley Scott', 'John', 'Hurt');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Lo squalo', 'Steven Spielberg', 'Roy', 'Scheider');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Lo squalo', 'Steven Spielberg', 'Robert', 'Shaw');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Iron Man', 'Jon Favreau', 'Robert', 'Downey Jr');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Iron Man', 'Jon Favreau', 'Jon', 'Favreau');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Iron Man 2', 'Jon Favreau', 'Robert', 'Downey Jr');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Iron Man 2', 'Jon Favreau', 'Jon', 'Favreau');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Iron Man 3', 'Shane Black', 'Robert', 'Downey Jr');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Iron Man 3', 'Shane Black', 'Jon', 'Favreau');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('The Avengers', 'Joe Johnston', 'Robert', 'Downey Jr');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('The Avengers', 'Joe Johnston', 'Jon', 'Favreau');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Guardiani della Galassia', 'James Gunn', 'Chris', 'Pratt');

INSERT INTO `noleggio-film`.`Recita` (`Titolo\_Recita`, `Regista\_Recita`, `Nome\_Attore\_Recita`, `Cognome\_Attore\_Recita`) VALUES ('Guardiani della Galassia', 'James Gunn', 'Dave', 'Bautista');

COMMIT;

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

-- Data for table `noleggio-film`.`Cliente`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('PLAMNF59P19L369O', 'Antonella', 'Federico', '1987-04-24');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('MNTXJR93M45H334J', 'Carlo', 'Spadafora', '1989-12-31');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('RWPXTC80M21C781I', 'Francesco', 'Nolcini', '1995-07-15');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('SVFBRZ61D21A929T', 'Federica', 'Carullo', '1991-11-06');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('ZSCPMH79R48D935B', 'Antonina', 'Spadafora', '1963-04-24');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('PHCXPT46R22L414Q', 'Ciro', 'Pertini', '1965-10-05');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('GNKHKB35H13L582K', 'Raffaella', 'Duodo', '1968-05-30');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('RWLMVF77L09B866V', 'Gianpietro', 'Pertini', '1991-11-06');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('NTNYCF95L02B856M', 'Alessia', 'Iannucci', '1987-04-24');

INSERT INTO `noleggio-film`.`Cliente` (`CF\_Cliente`, `Nome\_Cliente`, `Cognome\_Cliente`, `Data\_di\_Nascita`) VALUES ('cliente', 'Cliente', 'Cliente', '1997-10-21');

COMMIT;

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

-- Data for table `noleggio-film`.`Noleggio`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Noleggio` (`ID\_Noleggio`, `Data\_Restituzione`, `Numero\_Copia`, `Titolo\_Noleggio`, `Regista\_Noleggio`, `Cliente\_Noleggio`) VALUES (1, '2021-08-20', 1, 'Il padrino', 'Francis Ford Coppola', 'cliente');

INSERT INTO `noleggio-film`.`Noleggio` (`ID\_Noleggio`, `Data\_Restituzione`, `Numero\_Copia`, `Titolo\_Noleggio`, `Regista\_Noleggio`, `Cliente\_Noleggio`) VALUES (2, '2021-08-28', 1, 'Iron Man', 'Jon Favreau', 'cliente');

INSERT INTO `noleggio-film`.`Noleggio` (`ID\_Noleggio`, `Data\_Restituzione`, `Numero\_Copia`, `Titolo\_Noleggio`, `Regista\_Noleggio`, `Cliente\_Noleggio`) VALUES (3, '2021-09-05', 1, 'I predatori dell\'arca perduta', 'Steven Spielberg', 'cliente');

INSERT INTO `noleggio-film`.`Noleggio` (`ID\_Noleggio`, `Data\_Restituzione`, `Numero\_Copia`, `Titolo\_Noleggio`, `Regista\_Noleggio`, `Cliente\_Noleggio`) VALUES (4, '2021-09-06', 1, 'Iron Man 2', 'Jon Favreau', 'cliente');

INSERT INTO `noleggio-film`.`Noleggio` (`ID\_Noleggio`, `Data\_Restituzione`, `Numero\_Copia`, `Titolo\_Noleggio`, `Regista\_Noleggio`, `Cliente\_Noleggio`) VALUES (5, '2021-09-09', 1, 'L\'Impero colpisce ancora', 'Irvin Kershner', 'cliente');

INSERT INTO `noleggio-film`.`Noleggio` (`ID\_Noleggio`, `Data\_Restituzione`, `Numero\_Copia`, `Titolo\_Noleggio`, `Regista\_Noleggio`, `Cliente\_Noleggio`) VALUES (6, '2021-09-13', 1, 'Lo squalo', 'Steven Spielberg', 'cliente');

COMMIT;

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

-- Data for table `noleggio-film`.`Indirizzo`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via Latina', 7, 'Roma', 'cliente');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via Gela', 32, 'Milano', 'cliente');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via dei Selci', 543, 'Roma', 'PLAMNF59P19L369O');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via Nazionale', 13, 'Roma', 'PLAMNF59P19L369O');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via Torino', 71, 'Milano', 'MNTXJR93M45H334J');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via Roma', 65, 'Napoli', 'RWPXTC80M21C781I');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via Luigi X', 34, 'Napoli', 'GNKHKB35H13L582K');

INSERT INTO `noleggio-film`.`Indirizzo` (`Via`, `Numero`, `Città`, `Cliente\_Indirizzo`) VALUES ('Via del Monte', 4, 'Napoli', 'RWLMVF77L09B866V');

COMMIT;

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

-- Data for table `noleggio-film`.`Email`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('kulas.sam@yahoo.com', 1, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('pbernhard@hotmail.com', 1, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('quentin.collier@hotmail.com', 2, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('mayer.kirstin@yahoo.com', 3, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('langosh.mafalda@hotmail.com', 4, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('hilda.borer@hotmail.com', 5, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('bria43@pollich.info', 3, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('irunolfsson@rosenbaum.org', 2, NULL);

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('dexter91@hotmail.com', NULL, 'cliente');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('jerrod.collins@yahoo.com', NULL, 'cliente');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('lera29@gleason.info', NULL, 'PLAMNF59P19L369O');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('qschmidt@stamm.net', NULL, 'MNTXJR93M45H334J');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('casper43@okon.com', NULL, 'RWPXTC80M21C781I');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('torphy.aurore@pfannerstill.com', NULL, 'SVFBRZ61D21A929T');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('maxwell.fisher@zulauf.net', NULL, 'ZSCPMH79R48D935B');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('macejkovic.lucius@hotmail.com', NULL, 'PHCXPT46R22L414Q');

INSERT INTO `noleggio-film`.`Email` (`Indirizzo\_Email`, `Centro\_Email`, `Cliente\_Email`) VALUES ('immanuel10@hauck.com', NULL, 'GNKHKB35H13L582K');

COMMIT;

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

-- Data for table `noleggio-film`.`Telefono`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('064646337', 1, NULL);

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('066887148', 1, NULL);

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('03312337970', 2, NULL);

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('0813774713', 3, NULL);

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('0551486236', 4, NULL);

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('0100945696', 5, NULL);

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('068703410', NULL, 'cliente');

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('068969857', NULL, 'SVFBRZ61D21A929T');

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('064470176', NULL, 'PHCXPT46R22L414Q');

INSERT INTO `noleggio-film`.`Telefono` (`Numero\_Telefono`, `Centro\_Telefono`, `Cliente\_Telefono`) VALUES ('066558153', NULL, 'MNTXJR93M45H334J');

COMMIT;

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

-- Data for table `noleggio-film`.`Cellulare`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3230991995', 'PLAMNF59P19L369O');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3237389271', 'PLAMNF59P19L369O');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3238631306', 'PLAMNF59P19L369O');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3235162253', 'MNTXJR93M45H334J');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3253052758', 'MNTXJR93M45H334J');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3253128134', 'RWPXTC80M21C781I');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3256388222', 'SVFBRZ61D21A929T');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3254486702', 'cliente');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3298270288', 'cliente');

INSERT INTO `noleggio-film`.`Cellulare` (`Numero\_Cellulare`, `Cliente\_Cellulare`) VALUES ('3299254632', 'NTNYCF95L02B856M');

COMMIT;

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

-- Data for table `noleggio-film`.`utenti`

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

START TRANSACTION;

USE `noleggio-film`;

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('RDUKTV70H53B099U', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('ZDBSNS49E65D802Z', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('JPNXZE37A05L551H', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('MCGZYZ61B52E106Y', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('CXECDP70A18D869S', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('PHFHDH29M44D775Z', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('ZDGBFD71C54E658I', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('TXASHI84R46H343A', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('HCTGSG71R09H303V', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('YVWVFJ62D46D374X', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('NXWFGG27T04A745E', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('PNUGYA73A17I964P', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('LFSGGQ51L23I209P', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('TCCBRJ60M63B901D', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('VNTLMN53H64D341L', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('SZRFVG51L10F922P', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('MTGCGA71A21E296T', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('TCTZSU32P45H336S', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('PMMGMT59E43H407V', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('WWVZMZ42B22E382I', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('TRRRHR44S07F826R', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('SJLMYF59H06F556C', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('LDHDZP66S17I849M', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('CTXHJH90D01B207F', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('LSNSND95E70G381R', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('NCGMWM61E19I417J', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('DFVSNS99D46C696T', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('ZFCVMT32S64L354D', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('KZALNV44B14B493Z', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('NVRZVM85B48B760M', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('VKYTMS69E12G364K', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('QZGYWC69P24E746Q', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('LHCDFG87D47H046P', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('manager', '0c88028bf3aa6a6a143ed846f2be1ea4');

INSERT INTO `noleggio-film`.`utenti` (`username`, `password`) VALUES ('impiegato', '0c88028bf3aa6a6a143ed846f2be1ea4');

COMMIT;

## Codice del Front-End

**login.json**

{

"host": "localhost",

"username": "login",

"password": "login",

"port": 3306,

"database": "noleggio-film"

}

**manager.json**

{

"host": "localhost",

"username": "manager",

"password": "manager",

"port": 3306,

"database": "noleggio-film"

}

**impiegato.json**

{

"host": "localhost",

"username": "impiegato",

"password": "impiegato",

"port": 3306,

"database": "noleggio-film"

}

**main.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <mysql.h>

#include <mysql\_time.h>

#include "defines.h"

#include "manager.c"

#include "impiegato.c"

#include "inout.c"

#include "utils.c"

#include "parse.c"

typedef enum {

IMPIEGATO = 1,

MANAGER,

FAILED\_LOGIN

} role\_t;

struct configuration conf;

static MYSQL \*conn;

char var\_nome[64];

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; // IN

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);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // OUT

param[3].buffer = var\_nome;

param[3].buffer\_length = strlen(var\_nome);

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\_LONG; // OUT

param[0].buffer = &role;

param[0].buffer\_length = sizeof(role);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // OUT

param[1].buffer = var\_nome;

param[1].buffer\_length = 64;

if(mysql\_stmt\_bind\_result(login\_procedure, param)) {

print\_stmt\_error(login\_procedure, "Could not retrieve output parameter");

goto err;

}

// Retrieve output parameter

if(mysql\_stmt\_fetch(login\_procedure)) {

print\_stmt\_error(login\_procedure, "Could not buffer results");

goto err;

}

mysql\_stmt\_close(login\_procedure);

return role;

err:

mysql\_stmt\_close(login\_procedure);

err2:

return FAILED\_LOGIN;

}

int main(void) {

role\_t role;

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, "mysql\_real\_connect() failed\n");

mysql\_close (conn);

exit(EXIT\_FAILURE);

}

printf("\033[2J\033[H");

printf("██╗ ██╗███████╗██╗ ██████╗ ██████╗ ███╗ ███╗███████╗\n");

printf("██║ ██║██╔════╝██║ ██╔════╝██╔═══██╗████╗ ████║██╔════╝\n");

printf("██║ █╗ ██║█████╗ ██║ ██║ ██║ ██║██╔████╔██║█████╗\n");

printf("██║███╗██║██╔══╝ ██║ ██║ ██║ ██║██║╚██╔╝██║██╔══╝\n");

printf("╚███╔███╔╝███████╗███████╗╚██████╗╚██████╔╝██║ ╚═╝ ██║███████╗\n");

printf(" ╚══╝╚══╝ ╚══════╝╚══════╝ ╚═════╝ ╚═════╝ ╚═╝ ╚═╝╚══════╝\n\n");

printf("████████╗ ██████╗ ██████╗ ██╗ ██╗██████╗\n");

printf("╚══██╔══╝██╔═══██╗ ██╔══██╗██║ ██║██╔══██╗\n");

printf(" ██║ ██║ ██║ ██║ ██║██║ ██║██║ ██║\n");

printf(" ██║ ██║ ██║ ██║ ██║╚██╗ ██╔╝██║ ██║\n");

printf(" ██║ ╚██████╔╝ ██████╔╝ ╚████╔╝ ██████╔╝\n");

printf(" ╚═╝ ╚═════╝ ╚═════╝ ╚═══╝ ╚═════╝\n\n");

printf(" ██████╗███████╗███╗ ██╗████████╗███████╗██████╗\n");

printf("██╔════╝██╔════╝████╗ ██║╚══██╔══╝██╔════╝██╔══██╗\n");

printf("██║ █████╗ ██╔██╗ ██║ ██║ █████╗ ██████╔╝\n");

printf("██║ ██╔══╝ ██║╚██╗██║ ██║ ██╔══╝ ██╔══██╗\n");

printf("╚██████╗███████╗██║ ╚████║ ██║ ███████╗██║ ██║\n\n");

printf("Tax Code: ");

getInput(128, conf.username, false);

printf("Password: ");

getInput(128, conf.password, true);

role = attempt\_login(conn, conf.username, conf.password);

switch(role) {

case MANAGER:

run\_as\_manager(conn,var\_nome);

break;

case IMPIEGATO:

run\_as\_impiegato(conn,var\_nome);

break;

case FAILED\_LOGIN:

fprintf(stderr, "Invalid credentials\n");

exit(EXIT\_FAILURE);

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

printf("Bye!\n");

mysql\_close(conn);

return 0;

}

**manager.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

static void add\_workshift(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[5];

char options[3] = {'1','2','3'};

MYSQL\_TIME start;

MYSQL\_TIME end;

int dd,mm,yy;

char carica[64];

char centro\_c[4];

int centro;

char cf[64];

printf("\nEmployee Tax Code: ");

getInput(63, cf, false);

char op = multiChoice("Cashier, Clerk or Warehouseman?: ", options, 3);

switch(op) {

case '1':

strcpy(carica,"Cashier");

break;

case '2':

strcpy(carica,"Clerk");

break;

case '3':

strcpy(carica,"Warehouseman");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

in\_date:

printf("Start Date (dd-mm-yyyy): ");

scanf("%2d-%2d-%4d",&dd,&mm,&yy);

fflush(stdin);

if (date\_check(dd,mm,yy) == false){

printf("\nInvalid date, try again!\n\n");

goto in\_date;

}

start.day = dd;

start.month = mm;

start.year = yy;

fin\_date:

printf("End Date (dd-mm-yyyy): ");

scanf("%2d-%2d-%4d",&dd,&mm,&yy);

fflush(stdin);

if (date\_check(dd,mm,yy) == false){

printf("\nInvalid date, try again!\n\n");

goto fin\_date;

}

end.day = dd;

end.month = mm;

end.year = yy;

printf("Center ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_turno(?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize workshift insertion statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_DATE;

param[0].buffer = &start;

param[0].buffer\_length = sizeof(start);

param[1].buffer\_type = MYSQL\_TYPE\_DATE;

param[1].buffer = &end;

param[1].buffer\_length = sizeof(end);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = carica;

param[2].buffer\_length = strlen(carica);

param[3].buffer\_type = MYSQL\_TYPE\_LONG;

param[3].buffer = &centro;

param[3].buffer\_length = sizeof(centro);

param[4].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[4].buffer = cf;

param[4].buffer\_length = strlen(cf);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters for add workshift\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while adding the workshift.");

} else {

printf("\nWorkshift correctly added!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void show\_workshifts(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

MYSQL\_TIME var\_day;

int dd,mm,yy;

char centro\_c[4];

int centro;

l\_date:

printf("\nDate (dd-mm-yyyy): ");

scanf("%2d-%2d-%4d",&dd,&mm,&yy);

fflush(stdin);

if (date\_check(dd,mm,yy) == false){

printf("\nInvalid date, try again!\n\n");

goto l\_date;

}

var\_day.day = dd;

var\_day.month = mm;

var\_day.year = yy;

printf("Center ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call turni\_del\_giorno(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize show workshifts statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_DATE;

param[0].buffer = &var\_day;

param[0].buffer\_length = sizeof(var\_day);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &centro;

param[1].buffer\_length = sizeof(centro);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters for show workshifts\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred showing the workshifts.");

}

dump\_result\_set(conn, prepared\_stmt, "\nHere are the employees on duty:");

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void montly\_report(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[4];

char cf[64];

char mm\_c[64];

char yy\_c[64];

int mm;

short int yy;

int ore;

printf("\nEmployee Tax Code: ");

getInput(63, cf, false);

l\_month:

printf("Month [1-12]: ");

getInput(2, mm\_c, false);

if (isNumber(mm\_c) == 0){

printf("Invalid digit!\n”);

goto l\_month;

}

mm = atoi(mm\_c);

if (mm < 1 || mm > 12) {

printf("\nInvalid month, try again!\n\n");

goto l\_month;

}

l\_year:

printf("Year: ");

getInput(4, yy\_c, false);

if (isNumber(yy\_c) == 0){

printf("Invalid digit!\n”);

goto l\_year;

}

yy = atoi(yy\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call report\_mensile(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize monlty report statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf;

param[0].buffer\_length = strlen(cf);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &mm;

param[1].buffer\_length = sizeof(mm);

param[2].buffer\_type = MYSQL\_TYPE\_SHORT;

param[2].buffer = &yy;

param[2].buffer\_length = sizeof(yy);

param[3].buffer\_type = MYSQL\_TYPE\_LONG;

param[3].buffer = &ore;

param[3].buffer\_length = sizeof(ore);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters for montly report\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred showing montly report.");

goto end;

}

dump\_result\_set(conn, prepared\_stmt, "\nHere is the report:");

if (mysql\_stmt\_next\_result(prepared\_stmt) > 0){

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unexpected condition", true);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ore;

param[0].buffer\_length = sizeof(ore);

if(mysql\_stmt\_bind\_result(prepared\_stmt, param)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not retrieve output parameter", true);

}

if(mysql\_stmt\_fetch(prepared\_stmt)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not buffer results", true);

}

printf("\nThis employee collected %d hour/s of work!\n", ore);

end:

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void annual\_report(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char cf[64];

char year\_c[64];

short int year;

int var\_ore;

printf("\nEmployee Tax Code: ");

getInput(64, cf, false);

l\_year:

printf("Year: ");

getInput(4, year\_c, false);

if (isNumber(year\_c) == 0){

printf("Invalid digit!\n”);

goto l\_year;

}

year = atoi(year\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call report\_annuale(?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize annual report statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf;

param[0].buffer\_length = strlen(cf);

param[1].buffer\_type = MYSQL\_TYPE\_SHORT;;

param[1].buffer = &year;

param[1].buffer\_length = sizeof(year);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &var\_ore;

param[2].buffer\_length = sizeof(var\_ore);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters for annual report\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred showing annual report.");

goto end;

}

dump\_result\_set(conn, prepared\_stmt, "\nHere is the report:");

if (mysql\_stmt\_next\_result(prepared\_stmt) > 0){

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unexpected condition", true);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &var\_ore;

param[0].buffer\_length = sizeof(var\_ore);

if(mysql\_stmt\_bind\_result(prepared\_stmt, param)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not retrieve output parameter", true);

}

if(mysql\_stmt\_fetch(prepared\_stmt)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not buffer results", true);

}

printf("\nThis employee collected %d hour/s of work!\n", var\_ore);

end:

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void add\_employee(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[6];

char options[2] = {'1','2'};

char cf[64];

char nome[64];

char titolo\_studio[64];

char ruolo[64];

char numero\_c[10];

char pssw[64];

printf("\nEmployee Tax Code: ");

getInput(64, cf, false);

printf("Employee Name: ");

getInput(64, nome, false);

printf("Password for login: ");

getInput(64, pssw, false);

char op = multiChoice("Manager or Employee?: ", options, 2);

switch(op) {

case '1':

sprintf(ruolo,"Manager");

break;

case '2':

sprintf(ruolo,"Impiegato");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

printf("Qualification: ");

getInput(64, titolo\_studio, false);

l\_t1:

printf("\nTelephone Number: ");

getInput(10, numero\_c, false);

if (isNumber(numero\_c) == 0){

printf("Invalid digit!\n”);

goto l\_t1;

}

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_impiegato(?, ?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize employee insertion statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf;

param[0].buffer\_length = strlen(cf);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = nome;

param[1].buffer\_length = strlen(nome);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = titolo\_studio;

param[2].buffer\_length = strlen(titolo\_studio);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = numero\_c;

param[3].buffer\_length = strlen(numero\_c);

param[4].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[4].buffer = ruolo;

param[4].buffer\_length = strlen(ruolo);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[5].buffer = pssw;

param[5].buffer\_length = strlen(pssw);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters for add employee\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while adding the employee.");

} else {

printf("\nEmployee correctly added!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void change\_person\_charge(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char nome[64];

char centro\_c[4];

int centro;

printf("Center ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

printf("Employee Name: ");

getInput(64, nome, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call cambia\_responsabile(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize change person in charge statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &centro;

param[0].buffer\_length = sizeof(centro);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = nome;

param[1].buffer\_length = strlen(nome);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to change person in charge\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while changing the person in charge.");

} else {

printf("\nThe person in charge has been changed successfully!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void add\_film(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[7];

char options[2] = {'1','2'};

char titolo[64];

char regista[64];

int anno;

bool nuovo;

bool remake = false;

char titolo\_orignale[64];

char regista\_originale[64];

char actors\_c[16];

int actors;

char name[64];

char surname[64];

printf("\nFilm Title: ");

getInput(64, titolo, false);

printf("Movie Director: ");

getInput(64, regista, false);

printf("Production Year: ");

scanf("%4d",&anno);

fflush(stdin);

char op = multiChoice("Is a New Film or a Classic one?", options, 2);

switch(op) {

case '1':

nuovo = true;

break;

case '2':

nuovo = false;

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

op = multiChoice("Is this a Remake or an Orinal? ", options, 2);

switch(op) {

case '1':

remake = true;

printf("Which is the original one?: ");

getInput(64, titolo\_orignale, false);

printf("Directed by?: ");

getInput(64, regista\_originale, false);

break;

case '2':

remake = false;

strcpy(titolo\_orignale,"");

strcpy(regista\_originale,"");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_film(?, ?, ?, ?, ?, ? ,?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize add film statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = titolo;

param[0].buffer\_length = strlen(titolo);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = regista;

param[1].buffer\_length = strlen(regista);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &anno;

param[2].buffer\_length = sizeof(anno);

param[3].buffer\_type = MYSQL\_TYPE\_TINY;

param[3].buffer = &nuovo;

param[3].buffer\_length = sizeof(nuovo);

param[4].buffer\_type = MYSQL\_TYPE\_TINY;

param[4].buffer = &remake;

param[4].buffer\_length = sizeof(remake);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[5].buffer = titolo\_orignale;

param[5].buffer\_length = strlen(titolo\_orignale);

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[6].buffer = regista\_originale;

param[6].buffer\_length = strlen(regista\_originale);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to add the film\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while adding the film.");

} else {

printf("\nThe Film has been added successfully!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

op = multiChoice("\nDo you wanna add actors too? (1 = Yes / 2 = No)", options, 2);

switch(op) {

case '1':

act:

printf("How many?: ");

getInput(15, actors\_c, false);

if (isNumber(actors\_c) == 0){

printf("Invalid digit!\n”);

goto act;

}

actors = atoi(actors\_c);

for (int i=0; i<actors; i++){

printf("\nActor %d Name: ", i+1);

getInput(64, name, false);

printf("Actor %d Surname: ", i+1);

getInput(64, surname, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_attore\_recita(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize add actor statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = titolo;

param[0].buffer\_length = strlen(titolo);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = regista;

param[1].buffer\_length = strlen(regista);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = name;

param[2].buffer\_length = strlen(name);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = surname;

param[3].buffer\_length = strlen(surname);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to add the actor\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while adding the actor.");

} else {

printf("\nActor and Acting added successfully!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

}

break;

case '2':

printf("\nOk, no actors added.\n");

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

mysql\_stmt\_close(prepared\_stmt);

}

static void add\_film\_copy(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[5];

char titolo[64];

char regista[64];

char centro\_c[4];

int centro;

char settore\_c[4];

int settore;

char posizione\_c[4];

int posizione;

printf("\nFilm Title: ");

getInput(64, titolo, false);

printf("Movie Director: ");

getInput(64, regista, false);

printf("Center ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

printf("Sector ID: ");

getInput(4, settore\_c, false);

settore = atoi(settore\_c);

printf("Position: ");

getInput(4, posizione\_c, false);

posizione = atoi(posizione\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_copia(?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize add copy statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = titolo;

param[0].buffer\_length = strlen(titolo);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = regista;

param[1].buffer\_length = strlen(regista);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &centro;

param[2].buffer\_length = sizeof(centro);

param[3].buffer\_type = MYSQL\_TYPE\_LONG;

param[3].buffer = &settore;

param[3].buffer\_length = sizeof(settore);

param[4].buffer\_type = MYSQL\_TYPE\_LONG;

param[4].buffer = &posizione;

param[4].buffer\_length = sizeof(posizione);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to add the copy\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while adding the copy.");

} else {

printf("\nThe copy has been added successfully!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void qualifies\_film(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char titolo[64];

char regista[64];

printf("\nFilm Title: ");

getInput(64, titolo, false);

printf("Movie Director: ");

getInput(64, regista, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call dequalifica\_film(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize qualifies film statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = titolo;

param[0].buffer\_length = strlen(titolo);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = regista;

param[1].buffer\_length = strlen(regista);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to qualifies the film\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while qualifying the film.");

} else {

printf("\nThe Film has been qualified to Classic successfully!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

void run\_as\_manager(MYSQL \*conn, char\* var\_nome){

char options[10] = {'1','2','3','4','5','6','7','8','9','0'};

char op;

if(!parse\_config("users/manager.json", &conf)) {

fprintf(stderr, "\nUnable to load manager configuration\n");

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, conf.db\_username, conf.db\_password, conf.database)) {

fprintf(stderr, "\nmysql\_change\_user() failed\n");

exit(EXIT\_FAILURE);

}

printf("\nWelcome back %s!", var\_nome);

while(true) {

printf("\n\nAVAILABLE COMMANDS FOR MANAGERS:\n");

printf("-------------------------------------------\n");

printf("1) Add workshift\n");

printf("2) Show workshifts of day 'X' of center 'Y'\n");

printf("3) Monthly report\n");

printf("4) Annual report\n");

printf("5) Add new Employee\n");

printf("6) Change person in charge of the center\n");

printf("7) Add film\n");

printf("8) Add a film copy\n");

printf("9) Qualifies film to classic\n");

printf("0) QUIT\n");

op = multiChoice("\nSelect an option", options, 10);

switch(op) {

case '1':

add\_workshift(conn);

break;

case '2':

show\_workshifts(conn);

break;

case '3':

montly\_report(conn);

break;

case '4':

annual\_report(conn);

break;

case '5':

add\_employee(conn);

break;

case '6':

change\_person\_charge(conn);

break;

case '7':

add\_film(conn);

break;

case '8':

add\_film\_copy(conn);

break;

case '9':

qualifies\_film(conn);

break;

case '0':

return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

**impiegato.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include "defines.h"

static void available\_films(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char centro\_c[4];

int centro;

printf("\nCenter ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call film\_disponibili\_noleggio(?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize show available films statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &centro;

param[0].buffer\_length = sizeof(centro);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to show available films\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while showing available films.");

}

dump\_result\_set(conn, prepared\_stmt, "\nHere are the available films:");

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void rent\_film(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[8];

char cliente[64];

MYSQL\_TIME restituzione;

int dd,mm,yy;

char centro\_c[4];

int centro;

char film[64];

char regista[64];

float costo\_noleggio;

float costo\_giornaliero;

int num\_giorni;

printf("\nClient Tax Code: ");

getInput(64, cliente, false);

printf("Film: ");

getInput(64, film, false);

printf("Movie Director: ");

getInput(64, regista, false);

printf("Center ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

l\_date:

printf("Return date (dd-mm-yyyy): ");

scanf("%2d-%2d-%4d",&dd,&mm,&yy);

fflush(stdin);

if (date\_check(dd,mm,yy) == false){

printf("\nInvalid date, try again!\n\n");

goto l\_date;

}

restituzione.day = dd;

restituzione.month = mm;

restituzione.year = yy;

if(!setup\_prepared\_stmt(&prepared\_stmt, "call noleggia(?, ?, ?, ?, ?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize rent statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cliente;

param[0].buffer\_length = strlen(cliente);

param[1].buffer\_type = MYSQL\_TYPE\_DATE;

param[1].buffer = &restituzione;

param[1].buffer\_length = sizeof(restituzione);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &centro;

param[2].buffer\_length = sizeof(centro);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = film;

param[3].buffer\_length = strlen(film);

param[4].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[4].buffer = regista;

param[4].buffer\_length = strlen(regista);

param[5].buffer\_type = MYSQL\_TYPE\_LONG;

param[5].buffer = &num\_giorni;

param[5].buffer\_length = sizeof(num\_giorni);

param[6].buffer\_type = MYSQL\_TYPE\_FLOAT;

param[6].buffer = &costo\_noleggio;

param[6].buffer\_length = sizeof(costo\_noleggio);

param[7].buffer\_type = MYSQL\_TYPE\_FLOAT;

param[7].buffer = &costo\_giornaliero;

param[7].buffer\_length = sizeof(costo\_giornaliero);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to rent the film\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while renting the film.");

goto end;

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &num\_giorni;

param[0].buffer\_length = sizeof(num\_giorni);

param[1].buffer\_type = MYSQL\_TYPE\_FLOAT;

param[1].buffer = &costo\_noleggio;

param[1].buffer\_length = sizeof(costo\_noleggio);

param[2].buffer\_type = MYSQL\_TYPE\_FLOAT;

param[2].buffer = &costo\_giornaliero;

param[2].buffer\_length = sizeof(costo\_giornaliero);

if(mysql\_stmt\_bind\_result(prepared\_stmt, param)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not retrieve output parameter", true);

}

if(mysql\_stmt\_fetch(prepared\_stmt)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not buffer results", true);

}

printf("\nFilm successfully rented!\n\nThe price is €%.2f for rental and €%.2f per day.\n\nTotal Cost: €%.2f.\n", costo\_noleggio, costo\_giornaliero, costo\_noleggio + (costo\_giornaliero \* num\_giorni));

end:

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void return\_film(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char noleggio\_c[4];

int noleggio;

int prezzo;

printf("\nRent ID: ");

getInput(4, noleggio\_c, false);

noleggio = atoi(noleggio\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call restituisci\_film(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize return film statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &noleggio;

param[0].buffer\_length = sizeof(noleggio);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &prezzo;

param[1].buffer\_length = sizeof(prezzo);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to return film\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while return film.");

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &prezzo;

param[0].buffer\_length = sizeof(prezzo);

if(mysql\_stmt\_bind\_result(prepared\_stmt, param)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not retrieve output parameter", true);

}

if(mysql\_stmt\_fetch(prepared\_stmt)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Could not buffer results", true);

}

if (prezzo != 0) {

printf("\nRental returned successfully, but the film was supposed to be returned %d day/s ago!\n\nThe fee is €1 for each extra day.\n\nThe client has to pay an additional €%d.\n", prezzo, prezzo);

} else {

printf("\nRental returned successfully!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void expired\_rentals(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char centro\_c[4];

int centro;

printf("\nCenter ID: ");

getInput(4, centro\_c, false);

centro = atoi(centro\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call noleggi\_scaduti(?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize show expired rentals statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &centro;

param[0].buffer\_length = sizeof(centro);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to show expired rentals\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while showing expired rentals.");

}

dump\_result\_set(conn, prepared\_stmt, "\nHere are the expired rentals:");

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void client\_rentals(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char cliente[64];

printf("\nClient Tax Code: ");

getInput(64, cliente, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call noleggi\_cliente(?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize show client rentals statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = &cliente;

param[0].buffer\_length = strlen(cliente);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to show client rentals\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while showing client rentals.");

}

dump\_result\_set(conn, prepared\_stmt, "\nHere are the client rentals:");

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void client\_info(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

int status;

int i = 0;

char cliente[64];

printf("\nClient Tax Code: ");

getInput(64, cliente, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call info\_cliente(?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize show client infos statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = &cliente;

param[0].buffer\_length = strlen(cliente);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to show client infos\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while showing client infos.");

goto end;

}

do {

if(conn->server\_status & SERVER\_PS\_OUT\_PARAMS) {

goto next;

}

if (i == 0){

dump\_result\_set(conn, prepared\_stmt, "\nHere are the client infos:");

} else if (i == 1){

dump\_result\_set(conn, prepared\_stmt, "");

} else if (i == 2){

dump\_result\_set(conn, prepared\_stmt, "");

} else if (i == 3){

dump\_result\_set(conn, prepared\_stmt, "");

} else if (i == 4){

dump\_result\_set(conn, prepared\_stmt, "");

}

i++;

next:

status = mysql\_stmt\_next\_result(prepared\_stmt);

if (status > 0){

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unexpected condition", true);

}

} while (status == 0);

end:

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void add\_client(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[4];

char cliente[64];

char nome[64];

char cognome[64];

MYSQL\_TIME data\_nascita;

int dd,mm,yy;

printf("\nClient Tax Code: ");

getInput(64, cliente, false);

printf("Client Name: ");

getInput(64, nome, false);

printf("Client Surname: ");

getInput(64, cognome, false);

l\_date:

printf("Birth Date (dd-mm-yyyy): ");

scanf("%2d-%2d-%4d",&dd,&mm,&yy);

fflush(stdin);

if (date\_check(dd,mm,2222) == false){

printf("\nInvalid date, try again!\n\n");

goto l\_date;

}

data\_nascita.day = dd;

data\_nascita.month = mm;

data\_nascita.year = yy;

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_cliente(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize add client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = &cliente;

param[0].buffer\_length = strlen(cliente);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = &nome;

param[1].buffer\_length = strlen(nome);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = &cognome;

param[2].buffer\_length = strlen(cognome);

param[3].buffer\_type = MYSQL\_TYPE\_DATE;

param[3].buffer = &data\_nascita;

param[3].buffer\_length = sizeof(data\_nascita);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to add client\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while adding client.");

} else {

printf("\nClient correctly added!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void modify\_client(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[4];

char options[8] = {'1','2','3','4','5','6','7','8'};

char cliente[64];

char numero\_c[11];

int numero\_i;

int centro = 0;

char via[64];

char citta[64];

bool tipo = false;

printf("\nClient Tax Code: ");

getInput(64, cliente, false);

char op = multiChoice("\n1) Add Cellular Number\n2) Add Telephone Number\n3) Add Address\n4) Add Email\n5) Remove Cellular Number\n6) Remove Telephone Number\n7) Remove Address\n8) Remove Email\n\nWhich info you wanna modify?: ", options, 8);

switch(op) {

case '1':

l\_c1:

printf("\nCellular Number: ");

getInput(10, numero\_c, false);

if (isNumber(numero\_c) == 0){

printf("Invalid digit!\n”);

goto l\_c1;

}

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_cellulare(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = numero\_c;

param[0].buffer\_length = strlen(numero\_c);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = cliente;

param[1].buffer\_length = strlen(cliente);

break;

case '2':

l\_t1:

printf("\nTelephone Number: ");

getInput(10, numero\_c, false);

if (isNumber(numero\_c) == 0){

printf("Invalid digit!\n”);

goto l\_t1;

}

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_telefono(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = numero\_c;

param[0].buffer\_length = strlen(numero\_c);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &centro;

param[1].buffer\_length = sizeof(centro);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = cliente;

param[2].buffer\_length = strlen(cliente);

param[3].buffer\_type = MYSQL\_TYPE\_TINY;

param[3].buffer = &tipo;

param[3].buffer\_length = sizeof(tipo);

break;

case '3':

printf("\nCity: ");

getInput(64, citta, false);

printf("Street: ");

getInput(64, via, false);

printf("House Number: ");

getInput(2, numero\_c, false);

numero\_i = atoi(numero\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_indirizzo(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = via;

param[0].buffer\_length = strlen(via);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &numero\_i;

param[1].buffer\_length = sizeof(numero\_i);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = citta;

param[2].buffer\_length = strlen(citta);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = cliente;

param[3].buffer\_length = strlen(cliente);

break;

case '4':

printf("\nEmail: ");

getInput(64, via, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call aggiungi\_email(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = via;

param[0].buffer\_length = strlen(via);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &centro;

param[1].buffer\_length = sizeof(centro);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = cliente;

param[2].buffer\_length = strlen(cliente);

param[3].buffer\_type = MYSQL\_TYPE\_TINY;

param[3].buffer = &tipo;

param[3].buffer\_length = sizeof(tipo);

break;

case '5':

l\_c2:

printf("\nCellular Number: ");

getInput(10, numero\_c, false);

if (isNumber(numero\_c) == 0){

printf("Invalid digit!\n”);

goto l\_c2;

}

if(!setup\_prepared\_stmt(&prepared\_stmt, "call elimina\_cellulare(?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = numero\_c;

param[0].buffer\_length = strlen(numero\_c);

break;

case '6':

l\_t2:

printf("\nTelephone Number: ");

getInput(10, numero\_c, false);

if (isNumber(numero\_c) == 0){

printf("Invalid digit!\n”);

goto l\_t2;

}

if(!setup\_prepared\_stmt(&prepared\_stmt, "call elimina\_telefono(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cliente;

param[0].buffer\_length = strlen(cliente);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &centro;

param[1].buffer\_length = sizeof(centro);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = numero\_c;

param[2].buffer\_length = strlen(numero\_c);

param[3].buffer\_type = MYSQL\_TYPE\_TINY;

param[3].buffer = &tipo;

param[3].buffer\_length = sizeof(tipo);

break;

case '7':

printf("\nCity: ");

getInput(64, citta, false);

printf("\nStreet: ");

getInput(64, via, false);

printf("\nHouse Number: ");

getInput(2, numero\_c, false);

numero\_i = atoi(numero\_c);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call elimina\_indirizzo(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cliente;

param[0].buffer\_length = strlen(cliente);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = via;

param[1].buffer\_length = strlen(via);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &numero\_i;

param[2].buffer\_length = sizeof(numero\_i);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = citta;

param[3].buffer\_length = strlen(citta);

break;

case '8':

printf("\nEmail: ");

getInput(64, via, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call elimina\_email(?, ?, ?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize modify client statement\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cliente;

param[0].buffer\_length = strlen(cliente);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &centro;

param[1].buffer\_length = sizeof(centro);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = via;

param[2].buffer\_length = strlen(via);

param[3].buffer\_type = MYSQL\_TYPE\_TINY;

param[3].buffer = &tipo;

param[3].buffer\_length = sizeof(tipo);

break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to modify client\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while modifying client.");

} else {

printf("\nClient correctly motified!\n");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

static void film\_actors(MYSQL \*conn){

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char titolo[64];

char regista[64];

printf("\nFilm Title: ");

getInput(64, titolo, false);

printf("Movie Director: ");

getInput(64, regista, false);

if(!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_attori\_film(?, ?)", conn)) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nUnable to initialize get film info\n", false);

}

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = titolo;

param[0].buffer\_length = strlen(titolo);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = regista;

param[1].buffer\_length = strlen(regista);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "\nCould not bind parameters to get the film infos\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error (prepared\_stmt, "\nAn error occurred while getting the film infos.");

} else {

dump\_result\_set(conn, prepared\_stmt, "\nHere is the actor/s list:");

}

mysql\_stmt\_free\_result(prepared\_stmt);

for(; mysql\_next\_result(conn) == 0;)

mysql\_stmt\_close(prepared\_stmt);

}

void run\_as\_impiegato(MYSQL \*conn, char\* var\_nome){

char options[10] = {'1','2','3','4','5','6','7','8','9','0'};

char op;

if(!parse\_config("users/impiegato.json", &conf)) {

fprintf(stderr, "Unable to load impiegato 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);

}

printf("\nWelcome back %s!", var\_nome);

while(true) {

printf("\n\nAVAILABLE COMMANDS FOR EMPLOYERS:\n");

printf("---------------------------------\n");

printf("1) Show available films\n");

printf("2) Rent a film\n");

printf("3) Return film\n");

printf("4) Show expired rentals\n");

printf("5) Show client rentals\n");

printf("6) Client info\n");

printf("7) Add new client\n");

printf("8) Modify client info\n");

printf("9) Actors of the film\n");

printf("0) QUIT\n");

op = multiChoice("\nSelect an option", options, 10);

switch(op) {

case '1':

available\_films(conn);

break;

case '2':

rent\_film(conn);

break;

case '3':

return\_film(conn);

break;

case '4':

expired\_rentals(conn);

break;

case '5':

client\_rentals(conn);

break;

case '6':

client\_info(conn);

break;

case '7':

add\_client(conn);

break;

case '8':

modify\_client(conn);

break;

case '9':

film\_actors(conn);

break;

case '0':

return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

**defines.h**

#pragma once

#include <stdbool.h>

#include <mysql.h>

#include <ctype.h>

static bool date\_check(int dd, int mm, int yy){

if(yy>=2000 && yy<=9999){

if(mm>=1 && mm<=12){

if((dd>=1 && dd<=31) && (mm==1 || mm==3 || mm==5 || mm==7 || mm==8 || mm==10 || mm==12)){

return true;

} else if((dd>=1 && dd<=30) && (mm==4 || mm==6 || mm==9 || mm==11)){

return true;

} else if((dd>=1 && dd<=28) && (mm==2)){

return true;

} else if(dd==29 && mm==2 && (yy%400==0 ||(yy%4==0 && yy%100!=0))){

return true;

} else{

return false;

}

} else {

return false;

}

} else {

return false;

}

}

int isNumber(char s[]){

for (int i = 0; s[i]!= '\0'; i++) {

if (isdigit(s[i]) == 0)

return 0;

}

return 1;

}

struct configuration {

char \*host;

char \*db\_username;

char \*db\_password;

unsigned int port;

char \*database;

char username[128];

char password[128];

};

extern struct configuration conf;

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

extern char \*getInput(unsigned int lung, char \*stringa, bool hide);

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

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

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

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

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

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

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

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

extern void run\_as\_manager(MYSQL \*conn, char\* var\_nome);

extern void run\_as\_impiegato(MYSQL \*conn, char\* var\_nome);

**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 = 0;

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 pi˘ 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 pi˘ 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 Ë stata data

if(c == '\0') { // getInput() non puÚ 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 possibilit‡

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 Ë un carattere valido

for(i = 0; i < num; i++) {

if(c == choices[i])

return c;

}

}

}

**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: %s\n",

mysql\_stmt\_error (stmt));

}

}

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

{

fprintf (stderr, "%s\n", message);

if (conn != NULL) {

#if MYSQL\_VERSION\_ID >= 40101

fprintf (stderr, "Error %u (%s): %s\n",

mysql\_errno (conn), mysql\_sqlstate(conn), mysql\_error (conn));

#else

fprintf (stderr, "Error %u: %s\n",

mysql\_errno (conn), mysql\_error (conn));

#endif

}

}

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

{

bool update\_length = true;

\*stmt = mysql\_stmt\_init(conn);

if (\*stmt == NULL)

{

print\_error(conn, "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\_length);

return true;

}

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

{

print\_error(conn, message);

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

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

{

print\_stmt\_error(stmt, message);

if(close\_stmt) mysql\_stmt\_close(stmt);

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

static void print\_dashes(MYSQL\_RES \*res\_set)

{

MYSQL\_FIELD \*field;

unsigned int i, j;

mysql\_field\_seek(res\_set, 0);

putchar('+');

for (i = 0; i < mysql\_num\_fields(res\_set); i++) {

field = mysql\_fetch\_field(res\_set);

for (j = 0; j < field->max\_length + 2; j++)

putchar('-');

putchar('+');

}

putchar('\n');

}

static void dump\_result\_set\_header(MYSQL\_RES \*res\_set)

{

MYSQL\_FIELD \*field;

unsigned long col\_len;

unsigned int i;

/\* determine column display widths -- requires result set to be \*/

/\* generated with mysql\_store\_result(), not mysql\_use\_result() \*/

mysql\_field\_seek (res\_set, 0);

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;

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:

attr\_size = sizeof(short int);

break;

case MYSQL\_TYPE\_YEAR:

attr\_size = sizeof(short int);

break;

case MYSQL\_TYPE\_LONG:

attr\_size = sizeof(long int);

break;

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(" %02d-%02d-%d |", date->day, date->month, date->year);

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\_YEAR:

case MYSQL\_TYPE\_TINY:

printf(" %-\*d |", (int)fields[i].max\_length, \*(int \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_LONGLONG:

printf(" %-\*lld |", (int)fields[i].max\_length, \*(long long int \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_NEWDECIMAL:

printf(" %-\*.02lf |", (int)fields[i].max\_length, \*(float\*) rs\_bind[i].buffer);

break;

default:

printf("ERROR: Unhandled type (%d)\n", rs\_bind[i].buffer\_type);

abort();

}

}

putchar('\n');

print\_dashes(rs\_metadata);

}

mysql\_free\_result(rs\_metadata); /\* free metadata \*/

/\* free output buffers \*/

for (i = 0; i < num\_fields; i++) {

free(rs\_bind[i].buffer);

}

free(rs\_bind);

}

}

**parse.c**

#include <stddef.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

#define BUFF\_SIZE 4096

static char config[BUFF\_SIZE];

typedef enum {

JSMN\_UNDEFINED = 0,

JSMN\_OBJECT = 1,

JSMN\_ARRAY = 2,

JSMN\_STRING = 3,

JSMN\_PRIMITIVE = 4

} jsmntype\_t;

enum jsmnerr {

JSMN\_ERROR\_NOMEM = -1,

JSMN\_ERROR\_INVAL = -2,

JSMN\_ERROR\_PART = -3

};

typedef struct {

jsmntype\_t type;

int start;

int end;

int size;

#ifdef JSMN\_PARENT\_LINKS

int parent;

#endif

} jsmntok\_t;

typedef struct {

unsigned int pos;

unsigned int toknext;

int toksuper;

} jsmn\_parser;

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;

}

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;

}

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

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

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;

}

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++;

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

char c = js[parser->pos];

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;

}

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

int i;

parser->pos++;

switch (js[parser->pos]) {

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

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

break;

case 'u':

parser->pos++;

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

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

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

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

parser->pos = start;

return JSMN\_ERROR\_INVAL;

}

parser->pos++;

}

parser->pos--;

break;

default:

parser->pos = start;

return JSMN\_ERROR\_INVAL;

}

}

}

parser->pos = start;

return JSMN\_ERROR\_PART;

}

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;

}

}

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

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' :

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

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

default:

return JSMN\_ERROR\_INVAL;

#endif

}

}

if (tokens != NULL) {

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

if (tokens[i].start != -1 && tokens[i].end == -1) {

return JSMN\_ERROR\_PART;

}

}

}

return count;

}

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);

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];

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;

}

if (r < 1 || t[0].type != JSMN\_OBJECT) {

printf("Object expected\n");

return 0;

}

for (i = 1; i < r; i++) {

if (jsoneq(config, &t[i], "host") == 0) {

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;

}

1. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-1)