Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

SISTEMA DI GESTIONE DI SALE CINEMATOGRAFICHE

0227617

Nicola Rossi

Indice

1. Descrizione del Minimondo

Si vuole realizzare il sistema informativo di una catena di cinema, che si occupi anche della gestione delle prenotazioni.

L'amministrazione della catena gestisce i cinema. Ciascun cinema ha un numero arbitrario di sale, identificato da un numero di sala. In ogni sala c'è un numero arbitrario di posti, ciascun individuato da una lettera per la fila ed un numero di posto.

In ogni sala vengono proiettati più film quotidianamente. Ciascun cinema può proiettare lo stesso film più volte in una giornata, in sale differenti. Ogni film ha una durata, un nome, è associato al cast degli attori protagonisti e una casa cinematografica. Lo stesso film, proiettato in orari differenti e in sale differenti, può avere un costo per il biglietto differente, proprio in relazione alla sala e all'orario in cui esso viene proiettato.

Il sistema di prenotazione è tale per cui gli utenti possono prenotarsi per una visione, scegliendo un posto disponibile. Dal momento dell'inizio della procedura di prenotazione, un cliente ha a disposizione 10 minuti per perfezionare la prenotazione. Dopo aver scelto il posto, al cliente è data la possibilità di inserire i dati relativi alla propria carta di credito (numero, intestatario, data di scadenza, codice CVV). Una volta inseriti questi dati, il sistema restituisce all'utente un codice di prenotazione.
Fino a 30 minuti dall'inizio della proiezione, il cliente ha la possibilità di annullare la sua prenotazione fornendo al sistema il codice di prenotazione.

La catena di cinema gestisce anche i propri dipendenti, divisi in maschere e proiezionisti. Gli amministratori della catena definiscono i turni di lavoro, di otto ore massimo. I turni sono gestiti su base settimanale. Un report permette di sapere agli amministratori della catena se qualche spettacolo è sprovvisto di proiezionista o se qualche cinema, in qualche fascia oraria di apertura, è sprovvisto di almeno due maschere per la verifica dei biglietti all'ingresso.

La verifica dei biglietti avviene da parte delle maschere mediante l'utilizzo del codice di prenotazione. Un biglietto utilizzato viene associato nel sistema al fatto che quel determinato posto, per quella determinata proiezione, è stato occupato dallo spettatore. Non è possibile utilizzare più volte lo stesso codice di prenotazione per accedere al cinema.

A fini statistici, gli amministratori possono generare dei report mensili che mostrano per ciascun cinema e ciascuna sala quante prenotazioni sono state confermate, quante sono state annullate, e quante prenotazioni confermate non sono state utilizzate per accedere al cinema.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
10,11, 15	Cliente	Utente	È più appropriato e coerente con la specifica data
17	Catena di cinema	Amministratore	È più appropriato e coerente con la specifica data
24	Spettatore	Utente	È più appropriato e coerente con la specifica data

Specifica disambiguata

Si vuole realizzare il sistema informativo di una catena di cinema, che si occupi anche della gestione delle prenotazioni.

L'amministrazione della catena gestisce i cinema. Ciascun cinema ha un numero arbitrario di sale, identificato da un numero di sala. In ogni sala c'è un numero arbitrario di posti, ciascun individuato da una lettera per la fila ed un numero di posto.

In ogni sala vengono proiettati più film quotidianamente. Ciascun cinema può proiettare lo stesso film più volte in una giornata, in sale differenti. Ogni film ha una durata, un nome, è associato al cast degli attori protagonisti e una casa cinematografica. Lo stesso film, proiettato in orari differenti e in sale differenti, può avere un costo per il biglietto differente, proprio in relazione alla sala e all'orario in cui esso viene proiettato.

Il sistema di prenotazione è tale per cui gli utenti possono prenotarsi per una visione, scegliendo un posto disponibile. Dal momento dell'inizio della procedura di prenotazione, un utente ha a disposizione 10 minuti per perfezionare la prenotazione. Dopo aver scelto il posto, all' utente è data la possibilità di inserire i dati relativi alla propria carta di credito (numero, intestatario, data di scadenza, codice CVV). Una volta inseriti questi dati, il sistema restituisce all'utente un codice di prenotazione.
Fino a 30 minuti dall'inizio della proiezione, l' utente ha la possibilità di annullare la sua prenotazione fornendo al sistema il codice di prenotazione.

L' amministratore gestisce anche i propri dipendenti, divisi in maschere e proiezionisti. Gli amministratori della catena definiscono i turni di lavoro, di otto ore massimo. I turni sono gestiti su base settimanale. Un report permette di far sapere agli amministratori della catena se qualche spettacolo è sprovvisto di proiezionista o se qualche cinema, in qualche fascia oraria di apertura, è sprovvisto di almeno due maschere per la verifica dei biglietti all'ingresso.

La verifica dei biglietti avviene da parte delle maschere mediante l'utilizzo del codice di prenotazione. Un biglietto utilizzato viene associato nel sistema al fatto che quel determinato posto,

per quella determinata proiezione, è stato occupato dall'utente. Non è possibile utilizzare più volte lo stesso codice di prenotazione per accedere al cinema.

A fini statistici, gli amministratori possono generare dei report mensili che mostrano per ciascun cinema e ciascuna sala quante prenotazioni sono state confermate, quante sono state annullate, e quante prenotazioni confermate non sono state utilizzate per accedere al cinema.

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
CINEMA	Struttura che possiede un numero arbitrario di sale e dove vengono proiettati i film		SALA, FILM,UTENTE
SALA	Identificata da un numero e contenente un numero arbitrario di posti. Inoltre in ogni sala possono essere proiettati più film		POSTO, PROIEZIONE, CINEMA
POSTO	Individuato da una lettera per la fila ed un numero di posto. Il posto può essere occupato tramite una prenotazione		PRENOTAZIONE, SALA
PRENOTAZIONE	Il sistema di prenotazione è tale per cui gli utenti possono prenotarsi per una visione, scegliendo un posto disponibile.		POSTO,UTENTE, FILM
UTENTE	Colui che tramite il sistema di prenotazione ha la possibilità di occupare un posto per una visione	CLIENTE, SPETTATORE	PRENOTAZIONE, CINEMA, TURNO
FILM	Spettacolo proiettato in		PROIEZIONE, CINEMA,TURNO,

	più sale		PRENOTAZIONE
PROIEZIONE	Esecuzione di un film in una sala		FILM, SALA
TURNO	Definito dall'amministratore per quel dipendente		FILM, UTENTE

Raggruppamento dei requisiti in insiemi omogenei

FRASI DI CARATTERE GENERALE
Si vuole realizzare il sistema informativo di una catena di cinema, che si occupi anche della gestione delle prenotazioni.

FRASI RELATIVE AGLI AMMINISTRATORI
L'amministratore della catena gestisce i cinema. L'amministratore gestisce anche i propri dipendenti. Inoltre gli amministratori definiscono il turno di lavoro, di otto ore massimo. Un report permette di far sapere agli amministratori della catena se qualche spettacolo è sprovvisto di proiezionista o se qualche cinema, in qualche fascia oraria di apertura, è sprovvisto di almeno due maschere per la verifica dei biglietti all'ingresso. Gli amministratori possono generare dei report mensili che mostrano per ciascun cinema e ciascuna sala quante prenotazioni sono state confermate, quante sono state annullate, e quante prenotazioni confermate non sono state utilizzate per accedere al cinema.

FRASI RELATIVE AL CINEMA
Ciascun cinema ha un numero arbitrario di sale e può proiettare lo stesso film più volte in una giornata, in sale differenti

FRASI RELATIVE ALLA SALA ED AI POSTI
La sala è identificata dal proprio numero. In ogni sala c'è un numero arbitrario di posti, ciascun individuato da una lettera per la fila ed un numero di posto. Inoltre in ogni sala vengono proiettati più film quotidianamente.

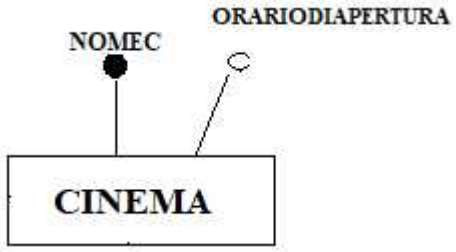
FRASI RELATIVE AL FILM
Ogni film ha una durata, un nome, è associato al cast degli attori protagonisti e una casa cinematografica. Lo stesso film, proiettato in orari differenti e in sale differenti, può avere un costo per il biglietto differente, proprio in relazione alla sala e all'orario in cui esso viene proiettato.

FRASI RELATIVE ALLA PRENOTAZIONE E ALL'UTENTE
Il sistema di prenotazione è tale per cui gli utenti possono prenotarsi per una visione, scegliendo un posto disponibile. Dal momento dell'inizio della procedura di prenotazione, un utente ha a disposizione 10 minuti per perfezionare la prenotazione. Dopo aver scelto il posto, all' utente è data la possibilità di inserire i dati relativi alla propria carta di credito (numero, intestatario, data di scadenza, codice CVV). Una volta inseriti questi dati, il sistema restituisce all'utente un codice di prenotazione. Fino a 30 minuti dall'inizio della proiezione, l' utente ha la possibilità di annullare la sua prenotazione fornendo al sistema il codice di prenotazione. Non è possibile utilizzare più volte lo stesso codice di prenotazione per accedere al cinema.

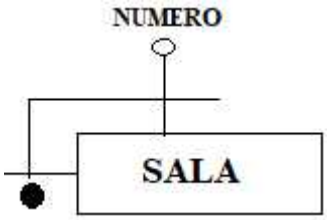
FRASI RELATIVE AI DIPENDENTI
I dipendenti sono divisi in maschere e proiezionisti. La verifica dei biglietti avviene da parte delle maschere mediante l'utilizzo del codice di prenotazione.

3. Progettazione concettuale

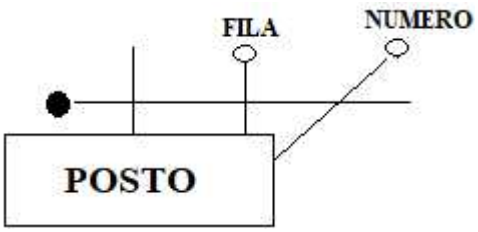
Costruzione dello schema E-R



Entità CINEMA con attributi:
NomeC (chiave): stringa univoca che identifica il cinema
OrarioDiApertura: orario di apertura del cinema

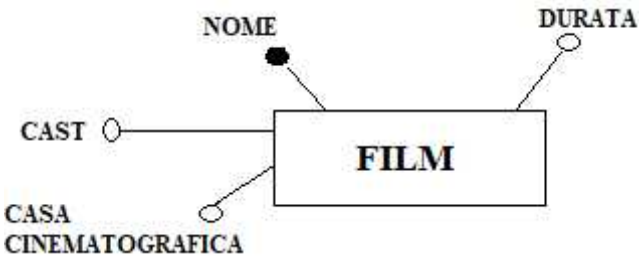


Entità SALA con attributi:
Numero (chiave): intero univoco che identifica una sala. Inoltre questa chiave funge da chiave esterna con cinema perché quel numero di sala può esserci in più cinema

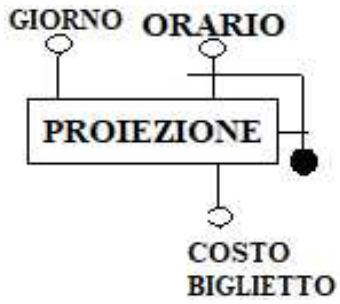


Entità POSTO con attributi:

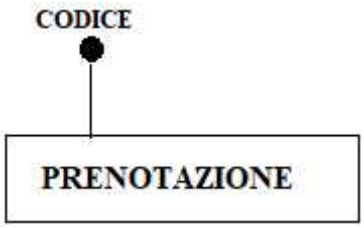
Fila e Numero (chiave): la prima chiave "fila" è una stringa che identifica in quale fila il posto è situato; la seconda chiave "numero" è un intero che identifica il numero di posto.
Inoltre queste chiavi fungono da chiave esterna con sala perché sia quella fila che quel numero di posto possono essere presenti in altre sale.



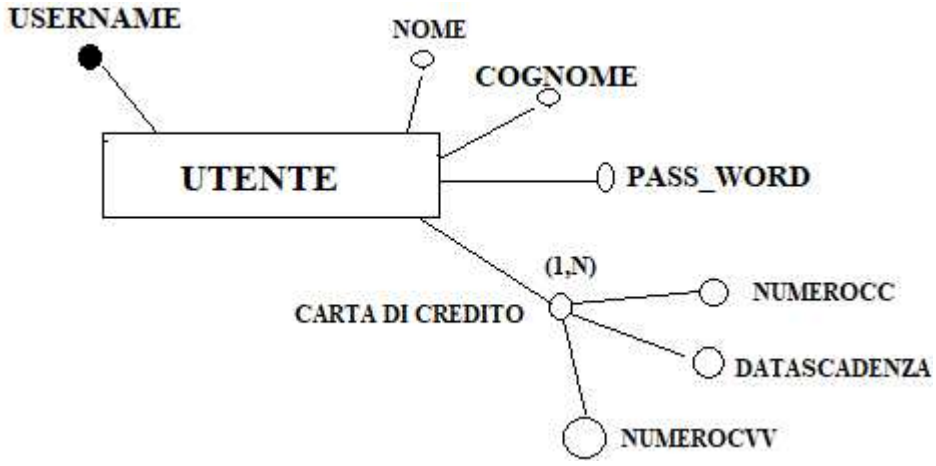
Entità FILM con attributi:
Nome (chiave): stringa univoca che identifica un film.
Durata: tempo di proiezione del film
Cast: numero di persone che partecipano alla produzione di un film
Casa cinematografica: nome dell'impresa che ha prodotto il film.



Entità PROIEZIONE con attributi:
Orario (chiave): tempo univoco che identifica l'orario di inizio del film
Giorno: data in cui viene proiettato il film
Costo Biglietto: prezzo del biglietto

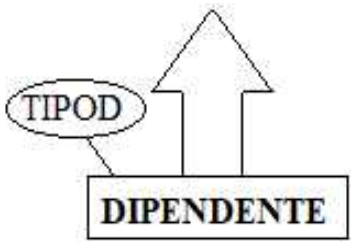


Entità PRENOTAZIONE con attributi:
Codice (chiave): intero che identifica una prenotazione

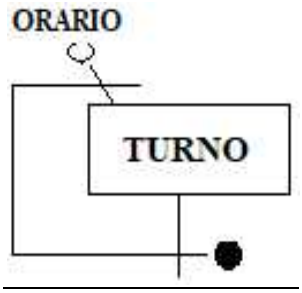


Entità UTENTE con attributi:

Username (chiave): stringa univoca inserita dall'utente in fase di registrazione.
Nome, cognome: informazioni dell'utente
Pass_word: password associata all'username dell'utente
Carta di credito con cardinalità (1.N): l'utente può avere più carte di credito ma è necessario pagare il posto con una. E' un attributo composto in cui viene inserito il numeroCC, la data di scadenza e il numeroCVV.

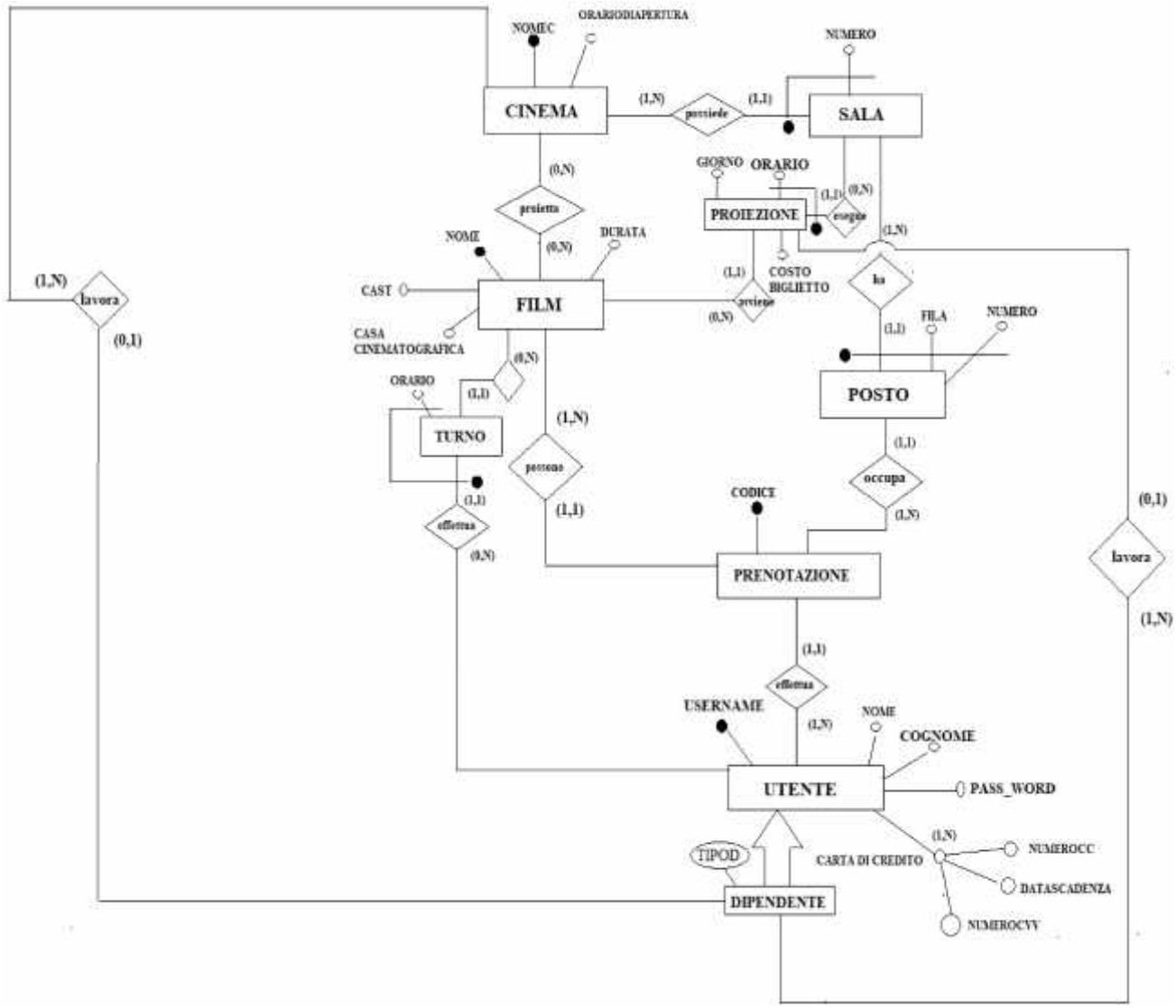


Dipendente: generalizzazione dell'utente
TipoD: tipo dipendente, ovvero maschera o proiezionista.



Entità TURNO con attributo:
Orario (chiave): fascia orario in cui può lavorare un dipendente.

Integrazione finale



Regole aziendali

Le regole aziendali sono utilizzate per rappresentare tutto quello che non si riesce ad inserire in un diagramma E-R

1) Il perfezionamento della prenotazione da parte di un cliente DEVE ESSERE di 10 minuti
2) Il tempo per annullare la prenotazione da parte di un cliente DEVE ESSERE di massimo 30 minuti dall'inizio della proiezione.

3) Il codice di prenotazione DEVE ESSERE utilizzato al massimo una volta per accedere allo stesso cinema.

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
CINEMA	Luogo che viene gestito dagli amministratori e dove vengono proiettati i film	NomeC., Orariodiapertura	NomeC
SALA	Parte di cinema dove vengono proiettati i film	Numero	Numero
FILM	Viene proiettato in orari differenti e in sale differenti e può avere un costo del biglietto differente	Nome, durata, cast, casa cinematografica	Nome
POSTO	Individuato da una lettera per fila e un numero	Fila, numero	Fila, numero
PRENOTAZIONE	Sistema che permette all'utente di prenotarsi per una visione	Codice	Codice
UTENTE	Colui che partecipa ad una visione tramite il sistema di prenotazione	Username, nome, cognome, pass_word	Username
TURNO	Fascia oraria per un dipendente	Orario	Orario
PROIEZIONE	Visione di un film in una sala	Orario, giorno, costo biglietto	Orario
DIPENDENTE	Colui che lavora nel cinema	Username, TipoD	Username

4. Progettazione logica

Volume dei dati

Concetto nello schema	Tipo[1]	Volume atteso
CINEMA	E	600
SALA	E	4200
FILM	E	21000
POSTO	E	1260000
PRENOTAZIONE	E	1000/al giorno
UTENTE	E	10/al giorno
PROIEZIONE	E	50/al giorno
TURNO	E	8/al giorno
POSSIEDE	R	7
PROIETTA (cinema-film)	R	350
HA	R	300
OCCUPA	R	100
EFFETTUA	R	3000
POSSONO	R	100/al giorno
LAVORA (dipendenteMaschera – cinema)	R	6/al giorno
ESEGUE	R	10/al giorno
AVVIENE	R	4/ogni settimana
EFFETTUA (turno – utente)	R	5/ogni settimana
LAVORA (dipendenteProiezionista-proiezione)	R	1/al giorno

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

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
OP1	Scegli posto	300/al giorno
OP2	Inserisci i dati relativi alla carta di credito dell'utente	200/al giorno
OP3	Restituire all'utente il codice di prenotazione	300/al giorno
OP4	Visualizza spettacolo sprovvisto di proiezionista, o se qualche cinema è sprovvisto di almeno due maschere per la verifica dei biglietti all'ingresso	10/al giorno
OP5	Visualizza numero di prenotazioni confermate	200/al giorno
OP6	Visualizza numero di prenotazioni annullate	70/al giorno
OP7	Visualizza numero di prenotazioni confermate ma non utilizzate per accedere al cinema	30/al giorno

Costo delle operazioni

OP1			
Concetto	**Costrutto**	**Accessi**	**Tipo**
POSTO	E	1	L

Costo: 1*300/al giorno = 300 accessi al giorno

OP5			
Concetto	**Costrutto**	**Accessi**	**Tipo**
PRENOTAZIONI CONFERMATE	E	1	L

Costo: 1*200/al giorno = 200 accessi al giorno

OP6			
Concetto	**Costrutto**	**Accessi**	**Tipo**
PRENOTAZIONI ANNULLATE	E	1	L

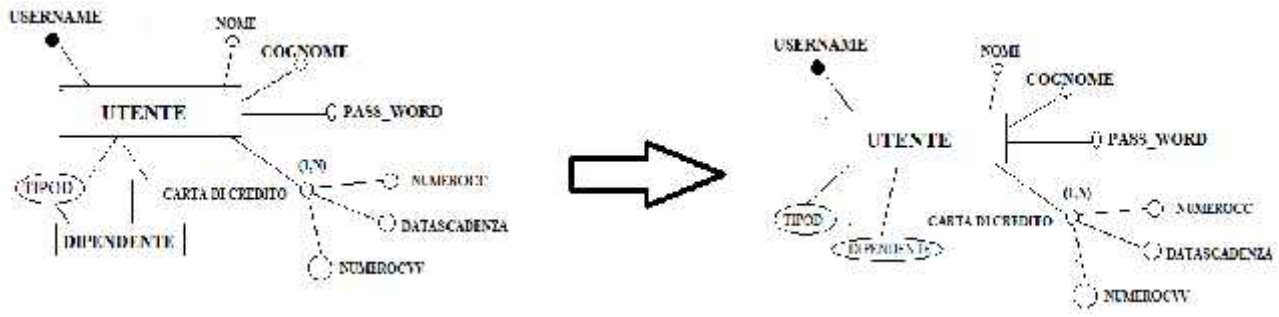
Costo: 1*70/al giorno = 70 accessi al giorno

OP7			
Concetto	**Costrutto**	**Accessi**	**Tipo**
PREOTAZIONI SPRECATE	E	1	L

Costo: 1*30/al giorno = 30 accessi al giorno

Ristrutturazione dello schema E-R

Elimino la generalizzazione sull'entità genitore "Utente" , incorporando in essa l'entità figlia "dipendente". Inoltre l'attributo dipendente può assumere il valore 0 o 1: 0 significa che non è un dipendente ma un utente, 1 che è un dipendente (maschera o proiezionista).

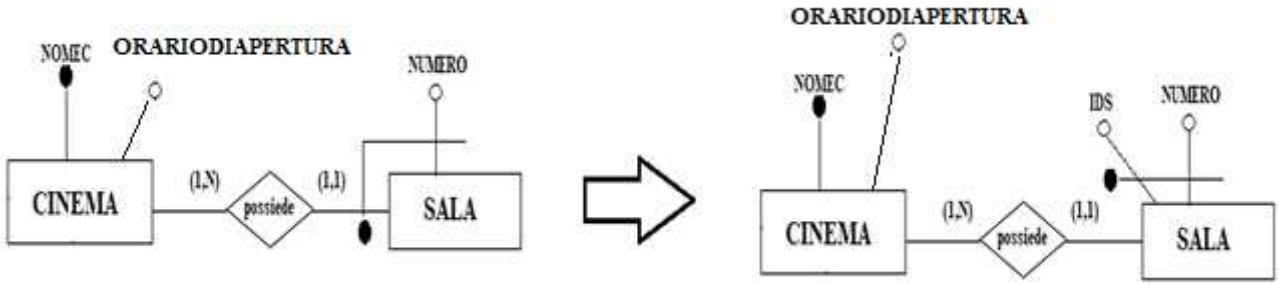


Identificatori primari:

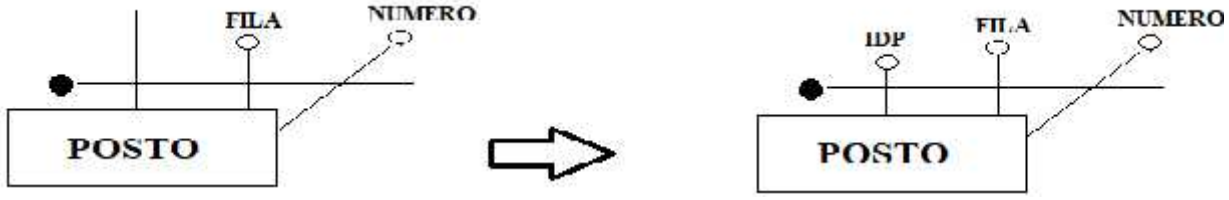
CINEMA → NomeC
SALA → Numero
POSTO → Numero, Fila
FILM → Nome
PRENOTAZIONE → Codice
UTENTE → Username
PROIETTA → film, cinema
PROIEZIONE → Orario
TURNO → Orario

Trasformazione di attributi e identificatori

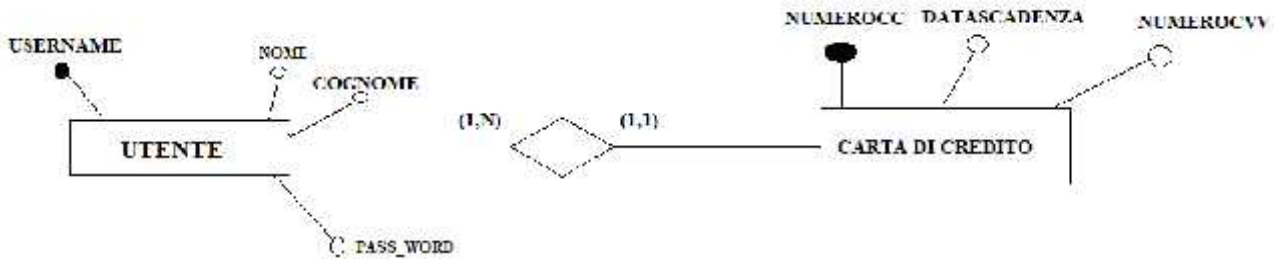
1) L'identificatore esterno presente nell'entità SALA viene sostituito con l'aggiunta dell'attributo- chiave "IDS" nell'entità "SALA".



2) L'identificatore esterno presente nell'entità POSTO viene sostituito con l'aggiunta dell'attributo-chiave "IDP" nell'entità POSTO.



3) L'attributo multi valore "CARTA DI CREDITO" dell'entità "UTENTE" viene sostituito dall'introduzione di un'associazione tra UTENTE e una nuova entità CARTA DI CREDITO.



Traduzione di entità e associazioni

CINEMA (NomeC, Orariodiapertura)
SALA (Numero, Cinema)
POSTO (Numero, Fila, Sala, Prenotazione)
PRENOTAZIONE (Codice, Utente,Tipo,Film)
UTENTE (Username, Nome, Cognome, Password, Dipendente, TipoD, Cinema)
FILM (Nome, Durata, Cast, Casa Cinematografica)
PROIETTA (Cinema, Film)
CARTADICREDITO(NumeroCC, DataScadenza, NumeroCVV, Utente)
TURNO (Orario, Utente, Film)
PROIEZIONE (Orario, Sala, Costo Biglietto, Giorno, Film, Utente_Proiezionista)

Vincoli di integrità:
SALA (Cinema) $\subseteq$ CINEMA (NomeC)
POSTO (Sala) $\subseteq$ SALA(Numero)
POSTO (Sala) $\subseteq$ SALA(Cinema)
POSTO (Prenotazione) $\subseteq$ PRENOTAZIONE (Codice)
PRENOTAZIONE (Utente) $\subseteq$ UTENTE (Username)
PRENOTAZIONE (Film) $\subseteq$ FILM (Nome)
PROIETTA (Cinema) $\subseteq$ CINEMA (NomeC)
PROIETTA (Film) $\subseteq$ FILM (Nome)
UTENTE (Cinema) $\subseteq$ CINEMA (NomeC)
TURNO (Utente) $\subseteq$ UTENTE (Username)
TURNO (Film) $\subseteq$ FILM(Nome)
PROIEZIONE (Sala) $\subseteq$ SALA (Numero, Cinema)
PROIEZIONE (Film) $\subseteq$ FILM(Nome)
CARTADICREDITO(Utente) $\subseteq$ UTENTE(Username)
PROIEZIONE(Utente_Proiezionista)) $\subseteq$ UTENTE(Username)

Normalizzazione del modello relazionale

Questa base dati non è in 1NF perché nell'entità "UTENTE" tutti i suoi attributi non sono semplici. Infatti in questa entità abbiamo l'attributo composto "CARTA DI CREDITO".
Per portare la relazione in 1NF occorre dividere la relazione in due relazioni separate.

UTENTE	USERNAME	NOME	COGNOME	PASS_WORD	TIPOD	DIPENDENTE

CARTE DI CREDITO	USERNAME	CARTA DI CREDITO

Per ricostruire la relazione originale è possibile compiere un join nella forma "utente" join "carte di credito".

Non essendoci, nelle relazioni, attributi che dipendono solo da una parte della chiave primaria, posso considerare il modello già in forma 2NF.

5. Progettazione fisica

Utenti e privilegi

```
CREATE USER 'amministratore' IDENTIFIED BY 'amministratore';

GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`generaReportMensile` TO 'amministratore';
GRANT ALL ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` TO
'amministratore';
GRANT SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile` TO
'amministratore';
GRANT ALL ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` TO
'amministratore';
GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`vediReportSprovvistoProiezionista` TO
'amministratore';
GRANT SELECT ON TABLE
`GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista` TO
'amministratore';
GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`VediReportSprovvistoMaschera` TO
'amministratore';
GRANT SELECT ON TABLE
`GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere` TO
'amministratore';
```

Creo l’user “amministratore” che amministra il sistema.

```
CREATE USER 'login' IDENTIFIED BY 'login';

GRANT ALL ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` TO 'login';
GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`checkDipendente` TO 'login';
GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`checkUser` TO
'login';
```

Creo l’user ‘login’ che ha il compito di interfacciarsi con la base di dati per gestire la registrazione di un nuovo utente e verificare se un utente che accede è registrato. Inoltre gestisce la registrazione di un dipendente.

```
CREATE USER 'utente' IDENTIFIED BY 'utente';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`ScegliPosto`
TO 'utente';
GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`PrenotareUnaVisione` TO 'utente';
GRANT DELETE, INSERT, GRANT OPTION, SELECT ON TABLE
`GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` TO 'utente';
```

```
GRANT INSERT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` TO
'utente';
GRANT INSERT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`POSTO` TO 'utente';
GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`InserireCarta`
TO 'utente';
GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`annullarePrenotazione` TO 'utente';
GRANT EXECUTE ON procedure
`GESTIONESALECINEMATOGRAFICHE`.`checkBiglietto` TO 'utente';
GRANT SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` TO
'utente';
```

Creo l'user 'utente' che identifica l'utilizzatore finale del sistema. Ha la possibilità di eliminare, inserire una PRENOTAZIONE, inserire i dati della CARTA DI CREDITO, inserire il POSTO. Inoltre se l'utente è un dipendente posso controllare se il codice del biglietto è valido o meno

Strutture di memorizzazione

Tabella CINEMA		
Attributo	**Tipo di dato**	**Attributi[2]**
NomeC	VARCHAR(45)	PK,NN
Orariodiapertura	TIME	NN

Tabella SALA		
Attributo	**Tipo di dato**	**Attributi[3]**
Numero	INT	PK,NN
Cinema_NomeC	VARCHAR(45)	PK,NN

[2] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[3] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella PROIETTAFC		
Attributo	**Tipo di dato**	**Attributi[4]**
Film_Nome	VARCHAR(45)	PK,NN
Cinema_NomeC	VARCHAR(45)	PK,NN

Tabella FILM		
Attributo	**Tipo di dato**	**Attributi[5]**
Nome	VARCHAR(45)	PK,NN
Durata	VARCHAR(45)	NN
Cast	INT	NN
CasaCinematografica	VARCHAR(45)	NN

[4] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[5] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella PROIEZIONE		
Attributo	**Tipo di dato**	**Attributi[6]**
Orario	DATETIME	PK,NN
Giorno	DATE	NN
Costo Biglietto	INT	NN
Sala_Numero	INT	PK,NN
Sala_Cinema_NomeC	VARCHAR(45)	PK,NN
Film_Nome	VARCHAR(45)	
Utente_Proiezionista	VARCHAR(45)	

Tabella POSTO		
Attributo	**Tipo di dato**	**Attributi[7]**
NumeroP	VARCHAR(45)	PK,NN
Fila	VARCHAR(45)	PK,NN
Sala_Numero	INT	PK,NN
Sala_Cinema_NomeC	VARCHAR(45)	PK,NN
Prenotazione_Codice	INT	NN

[6] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[7] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella PRENOTAZIONE		
Attributo	**Tipo di dato**	**Attributi[8]**
Codice	INT	PK,NN
Utente_Username	VARCHAR(45)	NN
Tipo	VARCHAR(45)	NN
Film_Nome	VARCHAR(45)	NN

Tabella UTENTE		
Attributo	**Tipo di dato**	**Attributi[9]**
Username	VARCHAR(45)	PK,NN
Nome	VARCHAR(45)	NN
Cognome	VARCHAR(45)	NN
Pass_Word	VARCHAR(45)	NN
Dipendente	VARCHAR(45)	NN
TipoD	VARCHAR(45)	NN
Cinema_NomeC	VARCHAR(45)	

[8] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[9] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella CARTA DI CREDITO		
Attributo	**Tipo di dato**	**Attributi[10]**
NumeroCC	BIGINT(16)	PK,NN
DataScandenza	DATE	NN
NumeroCVV	INT	NN,UN
Utente_Username	VARCHAR(45)	NN

Tabella TURNO		
Attributo	**Tipo di dato**	**Attributi[11]**
Orario	INT	PK,NN
Film_Nome	VARCHAR(45)	NN
Utente_Dipendente	VARCHAR(45)	PK,NN

Indici

Tabella CINEMA	
Indice	**Tipo[12]:**
PRIMARY	PR

[10] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[11] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

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

Tabella SALA	
Indice	**Tipo[13]:**
PRIMARY	PR
Fk_SALA_CINEMA1_idx	INDEX

Tabella FILM	
Indice	**Tipo[14]:**
PRIMARY	PR

Tabella PROIETTAFC	
Indice	**Tipo[15]:**
PRIMARY	PR
Fk_FILM_has_CINEMA_CINEMA1_idx	INDEX
Fk_FILM_has_CINEMA_FILM1_idx	INDEX

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

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

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

Tabella POSTO	
Indice	**Tipo[16]:**
PRIMARY	PR
Fk_POSTO_SALA1_idx	INDEX
Fk_POSTO_PRENOTAZIONE1_idx	INDEX

Tabella UTENTE	
Indice	**Tipo[17]:**
PRIMARY	PR
Fk_UTENTE_CINEMA1_idx	INDEX

Tabella PRENOTAZIONE	
Indice	**Tipo[18]:**
PRIMARY	PR
Fk_PRENOTAZIONE_UTENTE1_idx	INDEX
Fk_PRENOTAZIONE_FILM1_idx	INDEX

Tabella CARTA DI CREDITO	
Indice	**Tipo[19]:**
PRIMARY	PR
Fk_CARTADICREDITO_UTENTE1_idx	INDEX

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

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

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

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

Tabella TURNO	
Indice	**Tipo[20]:**
PRIMARY	PR
Fk_TURNO_FILM1_idx	INDEX
Fk_TURNO_UTENTE1_idx	INDEX

Tabella PROIEZIONE	
Indice	**Tipo[21]:**
PRIMARY	PR
Fk_PROIEZIONE_SALA1_idx	INDEX
Fk_PROIEZIONE_FILM1_idx	INDEX
Fk_PROIEZIONE_UTENTE1_idx	INDEX

Viste

Creo la vista “ReportMensile” col fine di avere in un’unica tabella le informazioni richieste nella specifica all’amministratore della base di dati. Dal join tra prenotazione e posto estraggo tutti i codici e tramite il group by ottengo per ciascun cinema e ciascuna sala quante prenotazioni(tramite il count(tipo)) sono state confermate, annullate e non utilizzate.

```
-- View `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile`;
DROP VIEW IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile` ;
USE `GESTIONESALECINEMATOGRAFICHE`;
CREATE  OR REPLACE VIEW `ReportMensile` AS
```

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

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

```
    SELECT
        sala_cinema_nomec,
        sala_numero,
        tipo,
        COUNT(tipo) AS conteggio
    FROM
        prenotazione
            JOIN
        posto
    WHERE
        codice = prenotazione_codice
    GROUP BY sala_cinema_nomec , sala_numero , tipo;
```

Creo la vista “ReportSprovvistoMaschere” col fine di far sapere all’amministratore se qualche cinema è sprovvisto di almeno due maschere.

```
-- View `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere`
-- -----------------------------------------------------
DROP TABLE IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere`;
DROP VIEW IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere` ;
USE `GESTIONESALECINEMATOGRAFICHE`;
CREATE  OR REPLACE VIEW `ReportSprovvistoMaschere` AS
    SELECT
        NOMEC AS CINEMA
    FROM
        CINEMA
    WHERE
        NOMEC NOT IN (SELECT
                NOMEC AS CINEMA
```

```
            FROM
                UTENTE
                    JOIN
                CINEMA
            WHERE
                NOMEC = CINEMA_NOMEC
            GROUP BY NOMEC
            HAVING COUNT(NOMEC) >= 2);
```

Creo la vista “ReportSprovvistoProiezionista” col fine di far sapere all’amministratore se qualche spettacolo è sprovvisto di proiezionista. Infatti dal where prendo il film che non ha un dipendente proiezionista.

```
-- View `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista`
-- ------------------------------------------------
DROP TABLE IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista`;
DROP VIEW IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista` ;
USE `GESTIONESALECINEMATOGRAFICHE`;
CREATE  OR REPLACE VIEW `ReportSprovvistoProiezionista` AS
    SELECT
        FILM_NOME AS FILM
    FROM
        PROIEZIONE
    WHERE
        FILM_NOME NOT IN (SELECT
                FILM_NOME AS FILM
            FROM
                UTENTE
                    JOIN
                PROIEZIONE ON USERNAME = UTENTE_PROIEZIONISTA);
```

Stored Procedures e transazioni

Con questa procedura verifico se l'utente è regolarmente iscritto al servizio

```
-- procedure checkUser
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`checkUser`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `checkUser` (IN user VARCHAR(45), pass VARCHAR(45))
BEGIN
     if (user not in (Select USERNAME from UTENTE)) then
           signal sqlstate '45000'
        SET message_text = 'Non registrato';
     end if;
    if (md5(pass) <> (select PASS_WORD from UTENTE where user =
USERNAME)) then
           signal sqlstate '45000'
        SET message_text = 'password errata';
     end if;

END$$

DELIMITER ;
```

Questa procedura permette all'amministratore di generare un report mensile per quanto riguarda le prenotazioni.

```
-- procedure generaReportMensile
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`generaReportMensile`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `generaReportMensile` ()
BEGIN
     select  SALA_NUMERO, SALA_CINEMA_NOMEC, TIPO, conteggio from
ReportMensile;
END$$

DELIMITER ;
```

Questa procedura permette all'amministratore di visualizzare il report sprovvisto di proiezionista

```
-- procedure vediReportSprovvistoProiezionista
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`vediReportSprovvistoProiezionista`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `vediReportSprovvistoProiezionista` ()
BEGIN
     select FILM from ReportSprovvistoProiezionista;
END$$

DELIMITER ;
```

Questa produra permette all'amministratore di visualizzare il report sprovvisto di maschere in quel cinema.

```
-- procedure VediReportSprovvistoMaschera
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`VediReportSprovvistoMaschera`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `VediReportSprovvistoMaschera` ()
BEGIN
     SELECT CINEMA FROM ReportSprovvistoMaschere;
END$$

DELIMITER ;
```

Questa procedura permette all'utente di annullare una prenotazione.

```
-- procedure annullarePrenotazione
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`annullarePrenotazione`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `annullarePrenotazione` (IN codice INT)
```

```
BEGIN

    start transaction;
           if ( codice NOT IN (select CODICE from PRENOTAZIONE where
codice = PRENOTAZIONE.CODICE
                  and tipo <> 'ANNULLATA')) then
                signal sqlstate '45000'
                SET message_text = 'Codice Errato, riprova';
           else
                UPDATE  PRENOTAZIONE SET tipo = 'ANNULLATA' WHERE
PRENOTAZIONE.CODICE = codice;
           end if;
     commit;

END$$

DELIMITER ;
```

Questa procedura permette all'utente di scegliere quale film vedere, indicando il nome.

```
-- procedure PrenotareUnaVisione
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`PrenotareUnaVisione`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `PrenotareUnaVisione` (IN utente VARCHAR(45), IN tipo
VARCHAR(45), in film VARCHAR(45))
BEGIN
     insert INTO PRENOTAZIONE (UTENTE_USERNAME, TIPO, FILM_NOME) values
(utente,tipo,film);
END$$

DELIMITER ;
```

Questa procedure permette all'utente di scegliere il posto, indicando il numero del posto, la fila, la sala e il cinema.

```
-- procedure ScegliPosto
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ScegliPosto`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
```

```
CREATE PROCEDURE `ScegliPosto` (IN posto VARCHAR(45), IN fila
VARCHAR(45), IN sala varchar(45), IN cinema VARCHAR(45), IN codice
VARCHAR(45))
BEGIN
      insert into POSTO (NUMEROP, FILA, SALA_NUMERO, SALA_CINEMA_NOMEC,
PRENOTAZIONE_CODICE) values (posto,fila,sala,cinema, codice);
END$$

DELIMITER ;
```

Questa procedura permette all'utente di inserire i dati della carta di credito per prenotare una visione, indicando il numero della carta, il mese di scadenza, l'anno di scadenza e il cvv.

```
-- procedure InserireCarta
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`InserireCarta`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `InserireCarta` ( IN Utente VARCHAR(45), IN NumeroCC
BIGINT(16), IN DataScadenza VARCHAR(45), IN NumeroCVV INT,  IN codice INT
)
BEGIN
    insert into CARTADICREDITO(NUMEROCC,
DATASCADENZA,NUMEROCVV,UTENTE_USERNAME) values (NumeroCC, DataScadenza,
NumeroCVV, Utente);
      update PRENOTAZIONE SET tipo = 'CONFERMATA' where CODICE = codice;
END$$

DELIMITER ;
```

Con questa procedura controllo se l'utente è un dipendente. Se accedo con l'username di utente non dipendente mi da errore. Inoltre il dipendente può essere un "proiezionista" o una "maschera".
Per controllare ciò utilizzo la procedure "check biglietto" che viene utilizzata solo dalla maschera.
Quindi se accedo come proiezionista e controllo il codice del biglietto, mi da errore perché non è una maschera.

```
-- procedure checkDipendente
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS
`GESTIONESALECINEMATOGRAFICHE`.`checkDipendente`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `checkDipendente` (IN var_username VARCHAR(45), IN
var_pass VARCHAR(45), OUT var_dipendente INT)
```

```
BEGIN

           declare var_dipendente_role ENUM ('maschera',
'proiezionista');

     SELECT
    TIPOD
FROM
    UTENTE
WHERE
    USERNAME = var_username
        AND PASS_WORD = MD5(var_pass) INTO var_dipendente_role;


        if var_dipendente_role = 'proiezionista' then
                set var_dipendente = 1;
        elseif var_dipendente_role = 'maschera' then
                set var_dipendente = 2;
           else
                set var_dipendente_role = 3;
           end if;



END$$

DELIMITER ;
```

Questa procedura permette alla maschera di controllare il biglietto tramite l'inserimento del codice di prenotazione.

```
-- procedure checkBiglietto
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`checkBiglietto`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `checkBiglietto` (IN usernamed VARCHAR(45),IN codiced
VARCHAR (45))
BEGIN
     if (usernamed IN (select USERNAME from UTENTE where TIPOD =
'maschera')) then
           start transaction;
                if (codiced NOT IN  (select CODICE from PRENOTAZIONE
where TIPO = 'CONFERMATA' and TIPO <> 'GIA CONFERMATO')) then
                     signal sqlstate '45000'
                SET MESSAGE_TEXT = 'Codice di conferma errato';
                else
                     UPDATE PRENOTAZIONE set tipo = 'GIA CONFERMATO'
where PRENOTAZIONE.CODICE = codiced;
```

```
                end if;
            commit;
      else
            signal sqlstate '45000'
            SET MESSAGE_TEXT = 'Non puoi controllare il biglietto perchè
non sei una maschera';




      end if;
END$$

DELIMITER ;
```

Appendice: Implementazione

Codice SQL per instanziare il database

```
-- MySQL Workbench Forward Engineering


SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';


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

-- Schema GESTIONESALECINEMATOGRAFICHE

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

DROP SCHEMA IF EXISTS `GESTIONESALECINEMATOGRAFICHE` ;


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

-- Schema GESTIONESALECINEMATOGRAFICHE

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

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

USE `GESTIONESALECINEMATOGRAFICHE` ;


-- -----------------------------------------------------
```

```
-- Table `GESTIONESALECINEMATOGRAFICHE`.`CINEMA`

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

DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` ;


CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (

  `NOMEC` VARCHAR(45) NOT NULL,

  `ORARIODIAPERTURA` TIME NOT NULL,

  PRIMARY KEY (`NOMEC`))

ENGINE = InnoDB;




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

-- Table `GESTIONESALECINEMATOGRAFICHE`.`SALA`

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

DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`SALA` ;


CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`SALA` (

  `NUMERO` INT NOT NULL,

  `CINEMA_NOMEC` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`NUMERO`, `CINEMA_NOMEC`),

  INDEX `fk_SALA_CINEMA1_idx` (`CINEMA_NOMEC` ASC),

  CONSTRAINT `fk_SALA_CINEMA1`

    FOREIGN KEY (`CINEMA_NOMEC`)
```

```
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`UTENTE`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` ;

CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (
  `USERNAME` VARCHAR(45) NOT NULL,
  `NOME` VARCHAR(45) NOT NULL,
  `COGNOME` VARCHAR(45) NOT NULL,
  `PASS_WORD` VARCHAR(45) NOT NULL,
  `DIPENDENTE` VARCHAR(45) NOT NULL,
  `TIPOD` VARCHAR(45) NOT NULL,
  `CINEMA_NOMEC` VARCHAR(45) NULL,
  PRIMARY KEY (`USERNAME`),
  INDEX `fk_UTENTE_CINEMA1_idx` (`CINEMA_NOMEC` ASC),
  CONSTRAINT `fk_UTENTE_CINEMA1`
    FOREIGN KEY (`CINEMA_NOMEC`)
```

```
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`FILM`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`FILM` ;

CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`FILM` (
  `NOME` VARCHAR(45) NOT NULL,
  `DURATA` VARCHAR(45) NOT NULL,
  `CAST` INT NOT NULL,
  `CASACINEMATOGRAFICA` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`NOME`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` ;


CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` (
  `CODICE` INT NOT NULL AUTO_INCREMENT,
  `UTENTE_USERNAME` VARCHAR(45) NOT NULL,
  `TIPO` VARCHAR(45) NOT NULL,
  `FILM_NOME` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CODICE`),
  INDEX `fk_PRENOTAZIONE_UTENTE1_idx` (`UTENTE_USERNAME` ASC),
  INDEX `fk_PRENOTAZIONE_FILM1_idx` (`FILM_NOME` ASC),
  CONSTRAINT `fk_PRENOTAZIONE_UTENTE1`
    FOREIGN KEY (`UTENTE_USERNAME`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PRENOTAZIONE_FILM1`
    FOREIGN KEY (`FILM_NOME`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`POSTO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`POSTO` ;

CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`POSTO` (
  `NUMEROP` VARCHAR(45) NOT NULL,
  `FILA` VARCHAR(45) NOT NULL,
  `PRENOTAZIONE_CODICE` INT NOT NULL,
  `SALA_NUMERO` INT NOT NULL,
  `SALA_CINEMA_NOMEC` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`NUMEROP`, `FILA`, `SALA_NUMERO`, `SALA_CINEMA_NOMEC`),
  INDEX `fk_POSTO_PRENOTAZIONE1_idx` (`PRENOTAZIONE_CODICE` ASC),
  INDEX `fk_POSTO_SALA1_idx` (`SALA_NUMERO` ASC, `SALA_CINEMA_NOMEC` ASC),
  CONSTRAINT `fk_POSTO_PRENOTAZIONE1`
    FOREIGN KEY (`PRENOTAZIONE_CODICE`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` (`CODICE`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_POSTO_SALA1`
    FOREIGN KEY (`SALA_NUMERO`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
COMMENT = '                    ';



-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` ;


CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` (
  `NUMEROCC` BIGINT(16) NOT NULL,
  `DATASCADENZA` VARCHAR(45) NOT NULL,
  `NUMEROCVV` INT UNSIGNED NOT NULL,
  `UTENTE_USERNAME` VARCHAR(45) NOT NULL,
  INDEX `fk_CARTADICREDITO_UTENTE1_idx` (`UTENTE_USERNAME` ASC),
  PRIMARY KEY (`NUMEROCC`),
  CONSTRAINT `fk_CARTADICREDITO_UTENTE1`
    FOREIGN KEY (`UTENTE_USERNAME`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)

ENGINE = InnoDB;



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

-- Table `GESTIONESALECINEMATOGRAFICHE`.`PROIEZIONE`

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

DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PROIEZIONE` ;


CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PROIEZIONE` (

  `ORARIO` TIME NOT NULL,

  `GIORNO` DATE NOT NULL,

  `COSTOBIGLIETTO` INT NOT NULL,

  `SALA_NUMERO` INT NOT NULL,

  `SALA_CINEMA_NOMEC` VARCHAR(45) NOT NULL,

  `FILM_NOME` VARCHAR(45) NULL,

  `UTENTE_PROIEZIONISTA` VARCHAR(45) NULL,

  PRIMARY KEY (`ORARIO`, `SALA_NUMERO`, `SALA_CINEMA_NOMEC`),

  INDEX `fk_PROIEZIONE_SALA1_idx` (`SALA_NUMERO` ASC, `SALA_CINEMA_NOMEC` ASC),

  INDEX `fk_PROIEZIONE_FILM1_idx` (`FILM_NOME` ASC),

  INDEX `fk_PROIEZIONE_UTENTE1_idx` (`UTENTE_PROIEZIONISTA` ASC),

  CONSTRAINT `fk_PROIEZIONE_SALA1`
```

```
  FOREIGN KEY (`SALA_NUMERO` , `SALA_CINEMA_NOMEC`)
  REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO` , `CINEMA_NOMEC`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_PROIEZIONE_FILM1`
  FOREIGN KEY (`FILM_NOME`)
  REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_PROIEZIONE_UTENTE1`
  FOREIGN KEY (`UTENTE_PROIEZIONISTA`)
  REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`TURNO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`TURNO` ;
```

```
CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`TURNO` (
  `ORARIO` INT NOT NULL,
  `FILM_NOME` VARCHAR(45) NOT NULL,
  `UTENTE_DIPENDENTE` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ORARIO`, `UTENTE_DIPENDENTE`),
  INDEX `fk_TURNO_FILM1_idx` (`FILM_NOME` ASC),
  INDEX `fk_TURNO_UTENTE1_idx` (`UTENTE_DIPENDENTE` ASC),
  CONSTRAINT `fk_TURNO_FILM1`
    FOREIGN KEY (`FILM_NOME`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TURNO_UTENTE1`
    FOREIGN KEY (`UTENTE_DIPENDENTE`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `GESTIONESALECINEMATOGRAFICHE`.`PROIETTAFC`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PROIETTAFC` ;


CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PROIETTAFC` (
  `CINEMA_NOMEC` VARCHAR(45) NOT NULL,
  `FILM_NOME` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CINEMA_NOMEC`, `FILM_NOME`),
  INDEX `fk_CINEMA_has_FILM_FILM1_idx` (`FILM_NOME` ASC),
  INDEX `fk_CINEMA_has_FILM_CINEMA1_idx` (`CINEMA_NOMEC` ASC),
  CONSTRAINT `fk_CINEMA_has_FILM_CINEMA1`
    FOREIGN KEY (`CINEMA_NOMEC`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_CINEMA_has_FILM_FILM1`
    FOREIGN KEY (`FILM_NOME`)
    REFERENCES `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


USE `GESTIONESALECINEMATOGRAFICHE` ;


-- -----------------------------------------------------
```

```
-- Placeholder table for view `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile`

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

CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile` (`sala_cinema_nomec` INT, `sala_numero` INT, `tipo` INT, `conteggio` INT);


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

-- Placeholder table for view `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista`

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

CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista` (`FILM` INT);


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

-- Placeholder table for view `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere`

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

CREATE TABLE IF NOT EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere` (`CINEMA` INT);


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

-- procedure checkDipendente

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


USE `GESTIONESALECINEMATOGRAFICHE`;
```

```
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`checkDipendente`;


DELIMITER $$

USE `GESTIONESALECINEMATOGRAFICHE`$$

CREATE PROCEDURE `checkDipendente` (IN var_username VARCHAR(45), IN var_pass VARCHAR(45), OUT var_dipendente INT)

BEGIN


		declare var_dipendente_role ENUM ('maschera', 'proiezionista');


	SELECT

  TIPOD

FROM

  UTENTE

WHERE

  USERNAME = var_username

    AND PASS_WORD = MD5(var_pass) INTO var_dipendente_role;



    if var_dipendente_role = 'proiezionista' then

			set var_dipendente = 1;

    elseif var_dipendente_role = 'maschera' then

			set var_dipendente = 2;
```

```
        else
            set var_dipendente_role = 3;
        end if;




END$$


DELIMITER ;


-- -----------------------------------------------------
-- procedure generaReportMensile
-- -----------------------------------------------------


USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`generaReportMensile`;


DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `generaReportMensile` ()
BEGIN
    select SALA_NUMERO, SALA_CINEMA_NOMEC, TIPO, conteggio from
ReportMensile;
```

```
END$$


DELIMITER ;


-- -----------------------------------------------------
-- procedure annullarePrenotazione
-- -----------------------------------------------------


USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`annullarePrenotazione`;


DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `annullarePrenotazione` (IN codice INT)
BEGIN


   start transaction;
                if ( codice NOT IN (select CODICE from PRENOTAZIONE where codice = PRENOTAZIONE.CODICE
                    and tipo <> 'ANNULLATA')) then
                   signal sqlstate '45000'
                   SET message_text = 'Codice Errato, riprova';
            else
```

```
                UPDATE PRENOTAZIONE SET tipo = 'ANNULLATA' WHERE PRENOTAZIONE.CODICE = codice;
            end if;
        commit;

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure PrenotareUnaVisione
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`PrenotareUnaVisione`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `PrenotareUnaVisione` (IN utente VARCHAR(45), IN tipo VARCHAR(45), in film VARCHAR(45))
BEGIN
    insert INTO PRENOTAZIONE (UTENTE_USERNAME, TIPO, FILM_NOME) values (utente,tipo,film);
END$$
```

```
DELIMITER ;

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

-- procedure ScegliPosto

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

USE `GESTIONESALECINEMATOGRAFICHE`;

DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ScegliPosto`;

DELIMITER $$

USE `GESTIONESALECINEMATOGRAFICHE`$$

CREATE PROCEDURE `ScegliPosto` (IN posto VARCHAR(45), IN fila VARCHAR(45), IN sala varchar(45), IN cinema VARCHAR(45), IN codice VARCHAR(45))

BEGIN

	insert into POSTO (NUMEROP, FILA, SALA_NUMERO, SALA_CINEMA_NOMEC, PRENOTAZIONE_CODICE) values (posto,fila,sala,cinema, codice);

END$$

DELIMITER ;

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

-- procedure InserireCarta
```

```
-- -----------------------------------------------------


USE `GESTIONESALECINEMATOGRAFICHE`;

DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`InserireCarta`;


DELIMITER $$

USE `GESTIONESALECINEMATOGRAFICHE`$$

CREATE PROCEDURE `InserireCarta` ( IN Utente VARCHAR(45), IN NumeroCC BIGINT(16), IN DataScadenza VARCHAR(45), IN NumeroCVV INT,  IN codice INT )

BEGIN

   insert into CARTADICREDITO(NUMEROCC, DATASCADENZA,NUMEROCVV,UTENTE_USERNAME) values (NumeroCC, DataScadenza, NumeroCVV, Utente);

        update PRENOTAZIONE SET tipo = 'CONFERMATA' where CODICE = codice;

END$$


DELIMITER ;


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

-- procedure checkBiglietto

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


USE `GESTIONESALECINEMATOGRAFICHE`;

DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`checkBiglietto`;
```

```
DELIMITER $$

USE `GESTIONESALECINEMATOGRAFICHE`$$

CREATE PROCEDURE `checkBiglietto` (IN usernamed VARCHAR(45),IN codiced VARCHAR (45))

BEGIN

	if (usernamed IN (select USERNAME from UTENTE where TIPOD = 'maschera')) then

		start transaction;

			if (codiced NOT IN  (select CODICE from PRENOTAZIONE where TIPO = 'CONFERMATA' and TIPO <> 'GIA CONFERMATO')) then

				signal sqlstate '45000'

	SET MESSAGE_TEXT = 'Codice di conferma errato';

			else

				UPDATE PRENOTAZIONE set tipo = 'GIA CONFERMATO' where PRENOTAZIONE.CODICE = codiced;



			end if;

		commit;

	else

		signal sqlstate '45000'

		SET MESSAGE_TEXT = 'Non puoi controllare il biglietto perchè non sei una maschera';
```

```
        end if;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure checkUser
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`checkUser`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `checkUser` (IN user VARCHAR(45), IN pass VARCHAR (45))
BEGIN
        if (user not in (select USERNAME from UTENTE)) then
                signal sqlstate '45000'
      SET message_text = 'Non registrato';
        end if;
  if (MD5(pass) <> (select PASS_WORD from UTENTE where user = USERNAME)) then
```

```
                signal sqlstate '45000'
        SET message_text = 'Password errata';
            end if;

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure vediReportSprovvistoProiezionista
-- -----------------------------------------------------

USE `GESTIONESALECINEMATOGRAFICHE`;
DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`vediReportSprovvistoProiezionista`;

DELIMITER $$
USE `GESTIONESALECINEMATOGRAFICHE`$$
CREATE PROCEDURE `vediReportSprovvistoProiezionista` ()
BEGIN
        SELECT FILM FROM ReportSprovvistoProiezionista;
END$$
```

```
DELIMITER ;


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

-- procedure VediReportSprovvistoMaschera

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


USE `GESTIONESALECINEMATOGRAFICHE`;

DROP procedure IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`VediReportSprovvistoMaschera`;


DELIMITER $$

USE `GESTIONESALECINEMATOGRAFICHE`$$

CREATE PROCEDURE `VediReportSprovvistoMaschera` ()

BEGIN

	SELECT CINEMA FROM ReportSprovvistoMaschere;

END$$


DELIMITER ;


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

-- View `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile`

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

DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile`;
```

```
DROP VIEW IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile` ;

USE `GESTIONESALECINEMATOGRAFICHE`;

CREATE  OR REPLACE VIEW `ReportMensile` AS
   SELECT
      sala_cinema_nomec,
      sala_numero,
      tipo,
      COUNT(tipo) AS conteggio
   FROM
      prenotazione
         JOIN
      posto
   WHERE
      codice = prenotazione_codice
   GROUP BY sala_cinema_nomec , sala_numero , tipo;


-- -----------------------------------------------------
-- View `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista`;
DROP VIEW IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista` ;
```

```
USE `GESTIONESALECINEMATOGRAFICHE`;

CREATE  OR REPLACE VIEW `ReportSprovvistoProiezionista` AS
   SELECT
      FILM_NOME AS FILM
   FROM
      PROIEZIONE
   WHERE
      FILM_NOME NOT IN (SELECT
            FILM_NOME AS FILM
         FROM
            UTENTE
               JOIN
            PROIEZIONE ON USERNAME = UTENTE_PROIEZIONISTA);


-- ----------------------------------------------------
-- View `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere`;
DROP VIEW IF EXISTS `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere` ;
USE `GESTIONESALECINEMATOGRAFICHE`;

CREATE  OR REPLACE VIEW `ReportSprovvistoMaschere` AS
```

```
    SELECT
        NOMEC AS CINEMA
    FROM
        CINEMA
    WHERE
        NOMEC NOT IN (SELECT
            NOMEC AS CINEMA
        FROM
            UTENTE
                JOIN
            CINEMA
        WHERE
            NOMEC = CINEMA_NOMEC
        GROUP BY NOMEC
        HAVING COUNT(NOMEC) >= 2);
SET SQL_MODE = '';
GRANT USAGE ON *.* TO amministratore;
 DROP USER amministratore;
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'amministratore' IDENTIFIED BY 'amministratore';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`generaReportMensile` TO 'amministratore';
```

```
GRANT ALL ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` TO 'amministratore';

GRANT SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`ReportMensile` TO 'amministratore';

GRANT ALL ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` TO 'amministratore';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`vediReportSprovvistoProiezionista` TO 'amministratore';

GRANT SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoProiezionista` TO 'amministratore';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`VediReportSprovvistoMaschera` TO 'amministratore';

GRANT SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`ReportSprovvistoMaschere` TO 'amministratore';

SET SQL_MODE = '';

GRANT USAGE ON *.* TO login;

 DROP USER login;

SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE USER 'login' IDENTIFIED BY 'login';


GRANT ALL ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` TO 'login';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`checkDipendente` TO 'login';
```

```
GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`checkUser` TO 'login';

SET SQL_MODE = '';

GRANT USAGE ON *.* TO utente;

 DROP USER utente;

SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE USER 'utente' IDENTIFIED BY 'utente';


GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`ScegliPosto` TO 'utente';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`PrenotareUnaVisione` TO 'utente';

GRANT DELETE, INSERT, GRANT OPTION, SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` TO 'utente';

GRANT INSERT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` TO 'utente';

GRANT INSERT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`POSTO` TO 'utente';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`InserireCarta` TO 'utente';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`annullarePrenotazione` TO 'utente';

GRANT EXECUTE ON procedure `GESTIONESALECINEMATOGRAFICHE`.`checkBiglietto` TO 'utente';

GRANT SELECT ON TABLE `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` TO 'utente';
```

```
SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;


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

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`CINEMA`

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

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('cinemanuovo', '9:00');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('cineplex', '18:00');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('av', '19:00');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('cd', '17:00');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('ab', '20:00');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('de', '8:00');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CINEMA` (`NOMEC`, `ORARIODIAPERTURA`) VALUES ('fh', '18:00');


COMMIT;
```

```
-- -----------------------------------------------------

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`SALA`

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

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (01, 'cinemanuovo');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (02, 'cineplex');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (03, 'cinemanuovo');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (04, 'av');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (01, 'cd');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (05, 'cineplex');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (05, 'ab');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (08, 'de');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (03, 'fh');
```

```
INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`SALA` (`NUMERO`, `CINEMA_NOMEC`) VALUES (07, 'cinemanuovo');


COMMIT;



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

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`UTENTE`

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

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('nico96@libero.it', 'nico', 'rossi', '0c88028bf3aa6a6a143ed846f2be1ea4', '0', '0', NULL);

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('pcm@gmail.com', 'paolo', 'marino', '3ce98305181b1bac59d024a49b0ffd73', '1', 'maschera', 'cinemanuovo');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('stbr97@outlook.it', 'stefano', 'reale', '21232f297a57a5a743894a0e4a801fc3', '1', 'proiezionista', NULL);

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('parl98@libero.it', 'gianluca', 'rossi', '63a9f0ea7bb98050796b649e85481845', '1', 'maschera', 'cineplex');
```

```
INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('amministratore', 'admin', 'admin', 'e792cd9665119b1244e8afcf36fb5f48', '0', '0', NULL);

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('adr96@libero.it', 'andrea', 'reale', '2e92d188dbdbd88e7c15831f4abfe1fc', '1', 'maschera', 'cinemanuovo');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`UTENTE` (`USERNAME`, `NOME`, `COGNOME`, `PASS_WORD`, `DIPENDENTE`, `TIPOD`, `CINEMA_NOMEC`) VALUES ('glr97@libero.it', 'paolo', 'rossi', '81bd8fe38f89db0696e623c09b6bb820', '1', 'maschera', 'cineplex');


COMMIT;



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

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`FILM`

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

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`, `DURATA`, `CAST`, `CASACINEMATOGRAFICA`) VALUES ('ladridibiciclette', '3 ore', 5, 'universal');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`, `DURATA`, `CAST`, `CASACINEMATOGRAFICA`) VALUES ('ilpadrino', '4 ore', 10, 'paramount');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`, `DURATA`, `CAST`, `CASACINEMATOGRAFICA`) VALUES ('ilcavaliereoscuro', '3 ore', 7, 'warnerbros');
```

```
INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`, `DURATA`, `CAST`, `CASACINEMATOGRAFICA`) VALUES ('quartopotere', '5 ore', 9, 'waltDisney');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`FILM` (`NOME`, `DURATA`, `CAST`, `CASACINEMATOGRAFICA`) VALUES ('ilsignoredeglianelli', '6 ore', 11, 'columbia');


COMMIT;



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

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE`

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

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` (`CODICE`, `UTENTE_USERNAME`, `TIPO`, `FILM_NOME`) VALUES (DEFAULT, 'nico96@libero.it', 'CONFERMATA', 'ilpadrino');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`PRENOTAZIONE` (`CODICE`, `UTENTE_USERNAME`, `TIPO`, `FILM_NOME`) VALUES (DEFAULT, 'nico96@libero.it', 'CONFERMATA', 'ladridibiciclette');


COMMIT;



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

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO`
```

```
-- -----------------------------------------------------

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` (`NUMEROCC`, `DATASCADENZA`, `NUMEROCVV`, `UTENTE_USERNAME`) VALUES (1234895674123569, '2025-05-02', 4562, 'nico96@libero.it');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` (`NUMEROCC`, `DATASCADENZA`, `NUMEROCVV`, `UTENTE_USERNAME`) VALUES (1536985412563303, '2022-08-06', 6589, 'pcm@gmail.com');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` (`NUMEROCC`, `DATASCADENZA`, `NUMEROCVV`, `UTENTE_USERNAME`) VALUES (8935712045691246, '2023-09-11', 9876, 'stbr97@outlook.it');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`CARTADICREDITO` (`NUMEROCC`, `DATASCADENZA`, `NUMEROCVV`, `UTENTE_USERNAME`) VALUES (9856423712368016, '2026-03-23', 8546, 'parl98@libero.it');


COMMIT;



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

-- Data for table `GESTIONESALECINEMATOGRAFICHE`.`PROIEZIONE`

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

START TRANSACTION;

USE `GESTIONESALECINEMATOGRAFICHE`;

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`PROIEZIONE` (`ORARIO`, `GIORNO`, `COSTOBIGLIETTO`, `SALA_NUMERO`, `SALA_CINEMA_NOMEC`,
```

`FILM_NOME`, `UTENTE_PROIEZIONISTA`) VALUES ('11:00', '2022/11/22', 10, 01, 'cinemanuovo', 'ilpadrino', 'stbr97@outlook.it');

INSERT INTO `GESTIONESALECINEMATOGRAFICHE`.`PROIEZIONE` (`ORARIO`, `GIORNO`, `COSTOBIGLIETTO`, `SALA_NUMERO`, `SALA_CINEMA_NOMEC`, `FILM_NOME`, `UTENTE_PROIEZIONISTA`) VALUES ('14:00', '2022/11/23', 20, 03, 'cinemanuovo', 'ladridibiciclette', NULL);

COMMIT;

Codice del Front-End

```
#include <stdio.h>
#include <stdlib.h>
#include <mysql.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>

#define CURRENT_YEAR 2021


char username[46];
char pwd[46];
MYSQL* conn;
int codice;
char comando5[1000];
```

```
void parseCreditCard (long long numberOfCC, char dateOfCC[46]){
    char *p;
    p = malloc(46*sizeof(char));
    strcpy(p, dateOfCC);
    long long t = numberOfCC;
    int j;
    int count=0;
    while (t){
        t /= 10;
        count++;
    }
    if (count != 16){
        printf("\nNumero carta credito non valido\n");
        exit(-1);
    }
    //printf("\n%i\n", count);
    for (int i=0; i<2; i++){
        p=strtok(p, "-");
        sscanf(p,"%d",&j);
        printf("%d\n",j);
        p=strtok(NULL, "-");
        //printf("%i",i);
        switch(i){
            case 0:
            if (j>13){
                printf("\nMese di scadenza non valido\n");
                exit(-1);
```

```
            }
            break;
                case 1:
                    if (j<CURRENT_YEAR){
                    printf("\nAnno scadenza non valido\n\n\n");
                    exit(-1);
            }
            break;
        }

    }
    return;

}



typedef enum{
    PROJECTIONIST = 1,
    MASK,
    FAILED_LOGIN
}role_t;




static role_t checkDipendente(){
    MYSQL_STMT *stmt;


    stmt = mysql_stmt_init(conn);
    MYSQL_BIND param[3];
```

```
    int var_dipendente = 0;

    if(mysql_stmt_prepare(stmt, "call checkDipendente(?, ?, ?)",
strlen("call checkDipendente(?, ?, ?)"))) {

        printf("\nErrore preparazione stamenent\n");

        exit(-1);

    }

        // 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 = pwd;

    param[1].buffer_length = strlen(pwd);


    param[2].buffer_type = MYSQL_TYPE_LONG; //OUT

    param[2].buffer = &var_dipendente;

    param[2].buffer_length = sizeof(var_dipendente);




    if (mysql_stmt_bind_param(stmt, param)){

        printf("\nErrore binding...\n");

        return FALSE;

    }


    if(mysql_stmt_execute(stmt)){

        printf("\nErrore esecuzione operazione\n");
```

```
        fprintf(stderr, "%s\n", mysql_error(conn));
        return FALSE;
    }
    memset(param, 0, sizeof(param));
    param[0].buffer_type = MYSQL_TYPE_LONG; //OUT
    param[0].buffer = &var_dipendente;
    param[0].buffer_length = sizeof(var_dipendente);

    if (mysql_stmt_bind_result(stmt, param)){
        printf("\nErrore binding...\n");
        return FALSE;
    }

    if (mysql_stmt_fetch(stmt)){
        printf("\nErrore fetch\n");
        fprintf(stderr, "%s\n", mysql_error(conn));
        return FALSE;
    }

    else{
        mysql_stmt_close(stmt);
        mysql_next_result(conn);
        return TRUE;}
    return var_dipendente;

}

bool checkUser(){
    MYSQL_STMT *stmt;
```

```
    stmt = mysql_stmt_init(conn);
    MYSQL_BIND param[2];
    if (mysql_stmt_prepare(stmt, "call checkUser(?,?)", strlen("call
checkUser(?,?)"))){
        printf("\nErrore preparazione statement\n ");
        exit(-1);
    }

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

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

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

    if (mysql_stmt_bind_param(stmt, param)){
        printf ("\nErrore binding...\n");
        return FALSE;
    }

    if (mysql_stmt_execute(stmt)){
        printf ("\nErrore esecuzione operazione\n");
        fprintf(stderr, "%s\n", mysql_error(conn));
        return FALSE;

    }
    else {
```

```
        return TRUE;}
}



void Prenotazione(){
    MYSQL_STMT *stmt;

    char tipo[46];
    int numerocvv;
    char film[46];
    char posto[46];
    char fila[46];
    char salan[46];
    int sala = 0;
    char cinema[46];
    long long numeroCarta = 0;
    char parameter[46];
    char temp[46];
    stmt = mysql_stmt_init(conn);
    char op[1];
    int t=0;
    int p = 0;
    int codice_prenotazione = 0;



    //MYSQL *conn = &connection; //il puntatore alla connessione a mysql
    MYSQL_RES *res; //il puntatore alla risorsa mysql
```

```
    MYSQL_ROW row;  // il vettore in cui verranno messi i risultati delle
query


seleziona:
    while(1){
        printf("\nCosa vuoi fare?:\n");
        printf("\n1.Prenotare Una Visione");
        printf("\n2.Annullare Prenotazione\n\n");
        scanf("%s",op);
        fflush(stdin);



         if(strcmp(op,"1") == 0){
                printf("\nCompila i seguenti campi.");


                        printf("\nScrivi nome del film:\n");
                        fflush(stdin);
                        fgets(film, 45, stdin);
                        strtok(film, "\n");
                        fflush(stdin);


                        stmt = mysql_stmt_init(conn);
                        MYSQL_BIND param_one[3];
                        if(mysql_stmt_prepare(stmt, "call
PrenotareUnaVisione(?,?,?)", strlen("call PrenotareUnaVisione(?,?,?)")))
{
                            printf("\nErrorepreparazionestamenent\n");
                            exit(-1);
                        }


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

```
param_one[0].buffer_type = MYSQL_TYPE_VARCHAR;
param_one[0].buffer = username;
param_one[0].buffer_length = strlen(username);

strcpy(tipo, "NON_UTILIZZATA");
param_one[1].buffer_type = MYSQL_TYPE_VARCHAR;
param_one[1].buffer = tipo;
param_one[1].buffer_length = strlen(tipo);

param_one[2].buffer_type = MYSQL_TYPE_VARCHAR;
param_one[2].buffer = film;
param_one[2].buffer_length = strlen(film);



if (mysql_stmt_bind_param(stmt, param_one)){
    printf("\nErrore binding...\n");
    exit(-1);
    }

if(mysql_stmt_execute(stmt)){
    printf("\nErrore esecuzione operazione\n");
    fprintf(stderr,"%s\n",mysql_error(conn));
    exit(-1);
}


else{
    printf("\nContinua!!!\n");
}
```

```
                        fflush(stdin);
                        mysql_stmt_close(stmt);



                        stmt = mysql_stmt_init(conn);



                        strcpy(comando5,"SELECT CODICE FROM PRENOTAZIONE
WHERE PRENOTAZIONE.UTENTE_USERNAME = '");
                        strcat(comando5, username);
                        strcat(comando5, "' AND PRENOTAZIONE.FILM_NOME
='");
                        strcat(comando5, film);
                        strcat(comando5, "';");

                        //strcpy(comando5, "call riceviCodice(?,?)");
                        if(mysql_query(conn, comando5)) {
                            fprintf(stderr, "%s\n", mysql_error(conn));
                            exit(-1);
                        }

                        else{
                            //*Prendo i risultati della query*/
                            res = mysql_use_result(conn);

                            while ((row = mysql_fetch_row(res)) != NULL){
                            //nel vettore row ci sono irisultati adesso
                                codice_prenotazione = atoi(row[0]);
                                }

                    }
```

```
fflush(stdin);
mysql_stmt_close(stmt);




printf("\nScegli posto:\n");

fflush(stdin);
printf("\n");
printf("Numero posto:\n");
fgets(posto, 45, stdin);
strtok(posto, "\n");
fflush(stdin);
printf("\n");
printf("Fila:\n");
fgets(fila, 45, stdin);
strtok(fila, "\n");
fflush(stdin);
printf("\n");
printf("Numero sala:\n");
fgets(salan, 45, stdin);

sala = atoi(salan);
fflush(stdin);
printf("Nome cinema:\n");
fgets(cinema, 45, stdin);
strtok(cinema, "\n");
fflush(stdin);
```

```
                        stmt = mysql_stmt_init(conn);
                        MYSQL_BIND param_two[5];
                        if(mysql_stmt_prepare(stmt, "call
ScegliPosto(?,?,?,?,?)", strlen("call ScegliPosto(?,?,?,?,?)"))) {
                            printf("\nErrorepreparazionestamenent\n");
                            return;
                        }



                        //Prepare parameters
                        memset(param_two, 0, sizeof(param_two));
                        param_two[0].buffer_type = MYSQL_TYPE_VARCHAR;
                        param_two[0].buffer = posto;
                        param_two[0].buffer_length = strlen(posto);

                        param_two[1].buffer_type = MYSQL_TYPE_VARCHAR;
                        param_two[1].buffer = fila;
                        param_two[1].buffer_length = strlen(fila);

                        param_two[2].buffer_type = MYSQL_TYPE_LONG;
                        param_two[2].buffer = &sala;
                        param_two[2].buffer_length = sizeof(sala);

                        param_two[3].buffer_type = MYSQL_TYPE_VARCHAR;
                        param_two[3].buffer = cinema;
                        param_two[3].buffer_length = strlen(cinema);
```

```
                        param_two[4].buffer_type = MYSQL_TYPE_LONG;
                        param_two[4].buffer = &codice_prenotazione;
                        param_two[4].buffer_length =
sizeof(codice_prenotazione);



                        if (mysql_stmt_bind_param(stmt, param_two)){
                            printf("\nErrore binding...\n");
                            return;
                            }

                        if(mysql_stmt_execute(stmt)){
                            printf("\nErrore esecuzione operazione\n");
                            fprintf(stderr,"%s\n",mysql_error(conn));
                            return;
                        }
                        else{
                            printf("\nPosto Inserito!!!\n");}



                        fflush(stdin);
                        mysql_stmt_close(stmt);

                    printf("\nInserisci dati carta\n");
                    fflush(stdin);
                    scanf("%lld",&numeroCarta);
                    fflush(stdin);
                    printf("\n");
                    printf("Inserisci mese scadenza: mm\n");
                    fgets(parameter, 45, stdin);
```

```
                    strtok(parameter, "\n");
                    fflush(stdin);
                    strcat(parameter, "-");
                    printf("\nInserisci anno di scadenza: yyyy\n");
                    fgets(temp, 45, stdin);
                    strtok(temp, "\n");
                    fflush(stdin);
                    strcat(parameter, temp);

                    printf("\nInserisci CVV:\n");
                    scanf("%i", &numerocvv);
                    fflush(stdin);



                    parseCreditCard(numeroCarta, parameter);

                    char comando6[1000];

                    stmt = mysql_stmt_init(conn);
                    MYSQL_BIND param_three[5];
                    memset(comando6, 0, 1000);




                    if(mysql_stmt_prepare(stmt, "call
InserireCarta(?,?,?,?,?)", strlen("call InserireCarta(?,?,?,?,?)"))) {
                        fprintf(stderr,"%s\n",mysql_error(conn));
                        printf("\nErrorepreparazionestamenent\n");
                    return;}

                    memset(param_three, 0, sizeof(param_three));
```

```
                    printf("%s\n", parameter);
                    param_three[0].buffer_type = MYSQL_TYPE_VARCHAR;
                    param_three[0].buffer = username;
                    param_three[0].buffer_length = strlen(username);

                    param_three[1].buffer_type = MYSQL_TYPE_LONGLONG;
                    param_three[1].buffer = &numeroCarta;
                    param_three[1].buffer_length = sizeof(numeroCarta);

                    param_three[2].buffer_type = MYSQL_TYPE_VARCHAR;
                    param_three[2].buffer = parameter;
                    param_three[2].buffer_length = strlen(parameter);

                    param_three[3].buffer_type = MYSQL_TYPE_LONG;
                    param_three[3].buffer = &numerocvv;
                    param_three[3].buffer_length = sizeof(numerocvv);

                    param_three[4].buffer_type = MYSQL_TYPE_LONG;
                    param_three[4].buffer = &codice_prenotazione;
                    param_three[4].buffer_length =
sizeof(codice_prenotazione);


                    if (mysql_stmt_bind_param(stmt, param_three)){
                        printf("\nErrore binding...\n");
                        return;}

                    if(mysql_stmt_execute(stmt)){
                        fprintf(stderr,"%s\n",mysql_error(conn));
                        printf("\nErroreesecuzioneoperazione\n");
```

```
                    return;}


                else{
                    printf("\nDati inseriti!!!\n");}
                fflush(stdin);
                mysql_stmt_close(stmt);


                printf("PRENOTAZIONE PER IL FILM %s AVVENUTA CON
SUCCESSO CON CODICE DI PRENOTAZIONE %i\n", film, codice_prenotazione);
                return;


            }



        else if(strcmp(op,"2")== 0){
            printf("\nScrivi il codice di prenotazione da rimuovere:\n");
            scanf("%i", &codice);


            stmt = mysql_stmt_init(conn);
            MYSQL_BIND param2[1];
            if(mysql_stmt_prepare(stmt, "call annullarePrenotazione(?)",
strlen("call annullarePrenotazione(?)"))) {
                printf("\nErrorepreparazionestamenent\n");
                exit(-1);
            }
            //Prepare parameters
            memset(param2, 0, sizeof(param2));
            param2[0].buffer_type = MYSQL_TYPE_LONG;
            param2[0].buffer = &codice;
            param2[0].buffer_length = sizeof(codice);
```

```
            if (mysql_stmt_bind_param(stmt, param2)){
                printf("\nErrore binding...\n");
                exit(-1);
                }

            if(mysql_stmt_execute(stmt)){
                printf("\nErrore esecuzione operazione\n");
                fprintf(stderr,"%s\n",mysql_error(conn));
                exit(-1);
                }

                else{
                    printf("\nRimosso!!!\n");}
                fflush(stdin);
                mysql_stmt_close(stmt);
                goto seleziona;









        }
}
}

bool ControlloBiglietto(){
    MYSQL_STMT *stmt;
    char query[1000];

    printf("\n");
```

```
    printf("Per controllare il biglietto inserisci il codice di
prenotazione:");
    printf("\n");
    scanf("%i", &codice);



    stmt = mysql_stmt_init(conn);
    MYSQL_BIND param[2];
    if(mysql_stmt_prepare(stmt,"call checkBiglietto(?,?)" , strlen("call
checkBiglietto(?,?)"))) {
        printf("\nErrorepreparazionestamenent\n");
        fprintf(stderr, "%s\n", mysql_error(conn));
        return;
    }


        //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_LONG; //IN
    param[1].buffer = &codice;
    param[1].buffer_length = sizeof(codice);



    if (mysql_stmt_bind_param(stmt, param)){
        printf("\nErrore binding...\n");
        return FALSE;
    }
```

```
    if(mysql_stmt_execute(stmt)){
        printf("\nErrore esecuzione operazione\n");
        fprintf(stderr, "%s\n", mysql_error(conn));
        return FALSE;
    }
    printf("\nCodice valido\n");

    mysql_stmt_close(stmt);
    return TRUE;









}


void amministratore(){
    char op[1];
    char richiesta [1000];
    int check = 0;
    MYSQL_STMT *stmt;
    MYSQL_RES *res; //ilpuntatore alla risorsa mysql
    MYSQL_ROW row;  // il vettore in cui verranno messi i risultati delle
query


    while(1){
            strcpy(username, "amministratore");
```

```
            printf("Inserire password:\n");
            fflush(stdin);
            fgets(pwd, 45, stdin);
            fflush(stdin);
            strtok(pwd, "\n");

            if (checkUser() == FALSE){
                printf("Non Autorizzato\n\n");
                return;
            }
            if
(mysql_change_user(conn,username,pwd,"GESTIONESALECINEMATOGRAFICHE")){
                fprintf(stderr, "Failed to change user.  Error: %s\n",
mysql_error(conn));
                printf("Accesso come amministratore fallito\n");
                exit(-1);
            }
            else{
                printf ("\nConnessione avvenuta con successo al
server\n\n");

                while(1){
                    printf("Cosa vuoi fare?\n\n");
                    printf("1. GeneraReport\n2. Vedi Report Sprovvisto di
Proiezionista\n3. Vedi Report Sprovvisto di Maschere\n\n ");
                    scanf("%s",op);

                    if(strcmp(op,"1") == 0){
                        if (mysql_query(conn, "call
generaReportMensile()")) {
                            fprintf(stderr, "%s\n", mysql_error(conn));
                            exit(-1);
                        }
```

```
                        else{
                            //*Prendo i risultati della query*/
                            res = mysql_use_result(conn);

                            /* Stampo a video i risultati della query */
                            printf("\nReport\n");
                            printf("N.
Sala\t\tCinema\t\tTipo\t\t\tCOUNT\n");
                            while ((row = mysql_fetch_row(res)) != NULL){
                            //nel vettore row ci sono irisultati adesso
                                printf("%10s\t%10s\t%10s\t%10s\n",
row[0],row[1], row[2], row[3]);
                                }
                    mysql_next_result(conn);

                    }}
                    else if (strcmp(op, "2") == 0){
                        if (mysql_query(conn, "call
vediReportSprovvistoProiezionista()")) {
                            fprintf(stderr, "%s\n", mysql_error(conn));
                            exit(-1);
                        }
                        else{
                            //*Prendo i risultati della query*/
                            res = mysql_use_result(conn);

                            /* Stampo a video i risultati della query */
                            printf("\nReport\n");
                            printf(" \nFilm\n");
                            while ((row = mysql_fetch_row(res)) != NULL){
                            //nel vettore row ci sono irisultati adesso
                                printf("%s\n", row[0]);
```

```
                    }
                mysql_next_result(conn);


                }}
                else if (strcmp(op, "3") == 0){
                    if (mysql_query(conn, "call
vediReportSprovvistoMaschera()")) {
                        fprintf(stderr, "%s\n", mysql_error(conn));
                        exit(-1);
                    }
                    else{
                        //*Prendo i risultati della query*/
                        res = mysql_use_result(conn);

                        /* Stampo a video i risultati della query */
                        printf("\nReport\n");
                        printf(" \nCinema\n");
                        while ((row = mysql_fetch_row(res)) != NULL){
                        //nel vettore row ci sono irisultati adesso
                            printf("%s\n", row[0]);
                            }
                mysql_next_result(conn);
                printf("\nBye!\n");
                mysql_close (conn);
                return 0;
                }}
```

```
    }


}}}




int main(void){
    role_t var_dipendente;

    char op[1];
    /*dichiaro le variabili di connessione*/
    char *server = "localhost";
    char *user = /*"root"; */"root";
    char *password = /*"admin";*/"admin";
    char *database = "GESTIONESALECINEMATOGRAFICHE";

    /*inizializzo la connessione*/
    conn = mysql_init(NULL);
    //MYSQL_STMT *stmt;

    /*se possibile mi connetto al database altrimenti esco dal programma
e scrivo un messaggio di errore*/
```

```
if(!mysql_real_connect(conn,server,user,password,database,0,NULL,CLIENT_M
ULTI_STATEMENTS)){
        fprintf(stderr, "%s\n", mysql_error(conn));
        printf("Connessione al server non riuscita");
        exit(-1);
    }
    else{
        printf("\nSistema sale cinematografiche.\nBenvenuto\nCosa vuoi
fare?\n");

        while(1) {
            printf("1) Accedi come utente\n");
            printf("2) Accedi come dipendente\n");
            printf("3) Amministratore\n\n");

            scanf("%s",op);
            if(strcmp(op,"1")==0){
                printf("\nInserisci username: \n");
                fflush(stdin);
                fgets(username,45,stdin);
                strtok(username,"\n");
                printf("\nInserisci password: \n");
                fflush(stdin);
                fgets(pwd,45,stdin);
                strtok(pwd,"\n");
                fflush(stdin);
                    if (checkUser() == TRUE){
                        Prenotazione();
                    }}
            else if(strcmp(op,"2")==0){
                printf("\nInserisci username: \n");
```

```
            fflush(stdin);
            fgets(username,45,stdin);
            strtok(username,"\n");
            printf("\nInserisci password: \n");
            fflush(stdin);
            fgets(pwd,45,stdin);
            strtok(pwd,"\n");
            fflush(stdin);
                if (checkDipendente() == TRUE){
                        ControlloBiglietto();

                }}
        else if (strcmp(op,"3")== 0){
            amministratore();
            //break;
        }
        else
            printf("\nValore non valido\n");
        }
    }




}
```