



Basi di Dati e Conoscenza  
Progetto A.A. 2018/2019

5

## SISTEMA DI GESTIONE DI SALE CINEMATOGRAFICHE

0239276

10

Ezio Emanuele Ditella

### Indice

<b>1. Descrizione del Minimondo.....</b>	<b>3</b>
<b>2. Analisi dei Requisiti.....</b>	<b>4</b>
<b>15 3. Progettazione concettuale.....</b>	<b>5</b>
<b>4. Progettazione logica.....</b>	<b>6</b>
<b>5. Progettazione fisica.....</b>	<b>8</b>
<b>Appendice: Implementazione.....</b>	<b>9</b>

20

## 1. Descrizione del Minimondo

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

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

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

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

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

25 La verifica dei biglietti avviene da parte delle maschere mediante l'utilizzo del codice di  
26 prenotazione. Un biglietto utilizzato viene associato nel sistema al fatto che quel  
27 determinato posto, per quella determinata proiezione, è stato occupato dallo spettatore.

28 Non è possibile utilizzare più volte lo stesso codice di prenotazione per accedere al  
29 cinema.

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

## 2. Analisi dei Requisiti

### Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
11	Prenotarsi	Acquistare biglietti	Per collegare concettualmente la procedura di prenotazione con i biglietti
11	Visione	Proiezione	Evitare uso di sinonimi
13	Perfezionare	Completare	Perfezionata è ambigua, è invece definita completa se sono stati inseriti tutti i dati enunciati nelle specifiche disambigue
16	Utente	Cliente	Evitare uso si sinonimi, a riga 17 fa riferimento ad utente come cliente
18	Prenotazione	Acquisto del biglietto	Legare il concetto di prenotazione a quella del biglietto
21	Report	Segnalazione	Presente un ononimo a riga 30 nelle specifiche originali
22	Spettacolo	Proiezione	Evitare uso si sinonimi
26	Biglietto utilizzato	Biglietto acquistato mediante il sistema di prenotazione	Esplicita utilizzato, meno ambiguo
27	Spettatore	Cliente	Evitare uso si sinonimi

### 5 Specifica disambiguata

Si vuole realizzare il sistema informativo di una catena di cinema. Il sistema informativo si occupa della gestione delle prenotazioni dei film e della gestione dei dipendenti tramite i loro turni di lavoro. L'amministrazione della catena gestisce i cinema. Ciascun cinema è identificato da un codice, univoco per ogni cinema della catena. Ogni cinema ha un numero arbitrario di sale. Ogni sala è identificata, all'interno di un cinema, da un numero di sala. In ogni sala c'è un numero arbitrario di posti. Ciascun posto è identificato, all'interno di una sala, da una lettera per la fila ed un numero di posto.

In ogni sala vengono proiettati più film quotidianamente. Lo stesso film può essere proiettato, in ciascun cinema, più volte in una giornata. Lo stesso film può essere proiettato contemporaneamente in sale differenti. Ogni film ha una durata, un nome, è associato al cast degli attori protagonisti e ad una casa cinematografica. Lo stesso film può avere un costo del biglietto differente in relazione alla

sala e all'orario in cui esso viene proiettato. In particolare uno stesso film ha costo differente del biglietto se è proiettato in orari uguali ma in sale differenti oppure in orari diversi ma sale uguali oppure orari e sale differenti. Quindi il costo dipende dalla proiezione di un determinato film.

Il sistema di prenotazione è tale per cui i clienti possono acquistare biglietti per una proiezione. Dal momento dell'inizio della procedura di prenotazione, un cliente ha a disposizione 10 minuti per completarla la stessa. La prenotazione è completa quando un cliente ha inserito orario di inizio del film, numero della sala (se il film è proiettato in piu' sale nell'orario scelto), numero e fila del posto ed i dati relativi alla propria carta di credito (numero, intestatario, data di scadenza, codice CVV). Una volta inseriti questi dati, il sistema restituisce al cliente un codice di prenotazione, che identifica univocamente ogni biglietto. Fino a 30 minuti dall'inizio della proiezione, il cliente ha la possibilità di annullare l'acquisto del biglietto fornendo al sistema il codice di prenotazione. Un cliente può prenotarsi per una proiezione fino e non oltre l'inizio della stessa.

La catena di cinema gestisce anche i propri dipendenti, divisi in maschere e proiezionisti. Gli amministratori della catena definiscono i turni di lavoro. Ciascun turno è consecutivo, ed è di otto ore al massimo. I turni sono gestiti su base settimanale dagli amministratori. Ciascun turno è identificato dal giorno, dalla fascia oraria della giornata (16-20 e 20-24) e dal cinema, ed è associato ad uno o più dipendenti. I turni vengono fatti "incrementalmente", ovvero vengono assegnati dal gestore durante la settimana (mentre si stanno usando i turni per la settimana in corso), e allo scadere del calendario dei turni per la settimana corrente, viene pubblicato il nuovo calendario dei turni. Una segnalazione permette di sapere agli amministratori della catena se qualche proiezione è sprovvista 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. Una segnalazione serve agli amministratori della catena proprio ad aiutare chi crea i turni affinchè ci sia sempre proiezionista per ogni proiezione ed almeno due maschere per la verifica dei biglietti.

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

A fini statistici, gli amministratori possono generare dei report mensili. Un report è un'operazione che mostra 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	Locale attrezzato per spettacoli cinematografici.		Sala, Report, Dipendente
Sala	Ambiente molto ampio utilizzato da più persone in cui guardare film.		Cinema, Posto, Proiezione, Biglietto
Posto			Sala, Biglietto
Film	Spettacolo cinematografico		Proiezione
Proiezione	Film proiettato al cinema in un certo orario		Film, Sala, Dipendente
Biglietto	'Lasciapassare' per vedere una proiezione al cinema	Prenotazione	Proiezione, Posto, Sala, Report
Report	Operazione che consente di verificare le tipologie di biglietti associati ad ogni proiezione (come nelle specifiche disambigue)		Biglietto, Cinema
Turno			Dipendente
Dipendente	Funzionario del cinema che può essere una maschera o un proiezionista		Turno, Cinema, Proiezione
Segnalazione	Operazione di controllo dei turni dei dipendenti, atta a verificare la presenza dei dipendenti		Turno, Dipendente

## Raggruppamento dei requisiti in insiemi omogenei

### Frasi relative ai cinema

L'amministrazione della catena gestisce i cinema. Ciascun cinema è identificato da un codice, univoco per ogni cinema della catena. Ogni cinema ha un numero arbitrario di sale.

### **Frasi relative alle sale**

Ogni sala è identificata, all'interno di un cinema, da un numero di sala. In ogni sala c'è un numero arbitrario di posti. In ogni sala vengono proiettati più film quotidianamente.

### **Frasi relative ai posti**

Ciascun posto è identificato, all'interno di una sala, da una lettera per la fila ed un numero di posto.

### **Frasi relative ai film**

In ogni sala vengono proiettati più film quotidianamente. Lo stesso film può essere proiettato, in ciascun cinema, più volte in una giornata.. Lo stesso film può essere proiettato contemporaneamente in sale differenti. Ogni film ha una durata, un nome, è associato al cast degli attori protagonisti e ad una casa cinematografica.

### **Frasi relative alle prenotazioni (biglietti)**

Il sistema di prenotazione è tale per cui i clienti possono acquistare biglietti per una proiezione. Lo stesso film può avere un costo del biglietto differente in relazione alla sala e all'orario in cui esso viene proiettato. In particolare uno stesso film ha costo differente del biglietto se è proiettato in orari uguali ma in sale differenti oppure in orari diversi ma sale uguali oppure orari e sale differenti. Quindi il costo dipende dalla proiezione di un determinato film. Dal momento dell'inizio della procedura di prenotazione, un cliente ha a disposizione 10 minuti per completarla la stessa. La prenotazione è completa quando un cliente ha inserito orario di inizio del film, numero della sala (se il film è proiettato in più sale nell'orario scelto), numero e fila del posto ed i dati relativi alla propria carta di credito (numero, intestatario, data di scadenza, codice CVV). Una volta inseriti questi dati, il sistema restituisce al cliente un codice di prenotazione, che identifica univocamente ogni biglietto. Fino a 30 minuti dall'inizio della proiezione, il cliente ha la possibilità di annullare l'acquisto del biglietto fornendo al sistema il codice di prenotazione. Un cliente può prenotarsi per una proiezione fino e non oltre l'inizio della stessa. La verifica dei biglietti avviene da parte delle maschere mediante l'utilizzo del codice di prenotazione. Un biglietto acquistato mediante il sistema di prenotazione viene associato nel sistema al fatto che quel determinato posto, per quella determinata proiezione, è stato occupato dal cliente. Non è possibile utilizzare più volte lo stesso codice di prenotazione.

5

### **Frasi relative alle prenotazioni (biglietti) e report**

A fini statistici, gli amministratori possono generare dei report mensili. Un report è un'operazione che mostra 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 ai turni e dipendenti**

La catena di cinema gestisce anche i propri dipendenti, divisi in maschere e proiezionisti. Gli amministratori della catena definiscono i turni di lavoro. Ciascun turno è consecutivo, ed è di al massimo. I turni sono gestiti su base settimanale dagli amministratori. Ciascun turno è

identificato dal giorno, dalla fascia oraria della giornata e dal cinema, ed è associato ad uno o più dipendenti. I turni vengono fatti “incrementalmente”, ovvero vengono assegnati dal gestore durante la settimana (mentre si stanno usando i turni per la settimana in corso), e allo scadere del calendario dei turni per la settimana corrente, viene pubblicato il nuovo calendario dei turni.

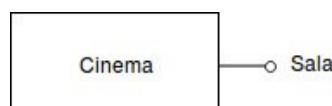
### **Frasi relative alle segnalazioni**

Una segnalazione permette di sapere agli amministratori della catena se qualche proiezione è sprovvista 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. Una segnalazione serve agli amministratori della catena proprio ad aiutare chi crea i turni affinchè ci sia sempre proiezionario per ogni proiezione ed almeno due maschere per la verifica dei biglietti.

### 3. Progettazione concettuale

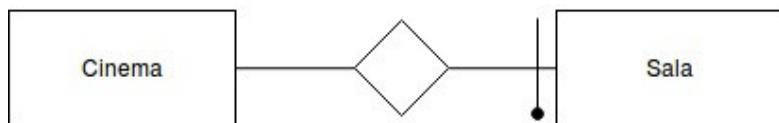
#### Costruzione dello schema E-R

Per la costruzione del modello E-R è stata utilizzata una tecnica mista. Per prima cosa sono stati analizzati I concetti principali enunciati dalle specifiche, per poi collegarli tra di loro e arricchendo tali concetti in base alle richieste delle specifiche. Il concetto principale è quello del Cinema, che avrà un determinato numero di sale.

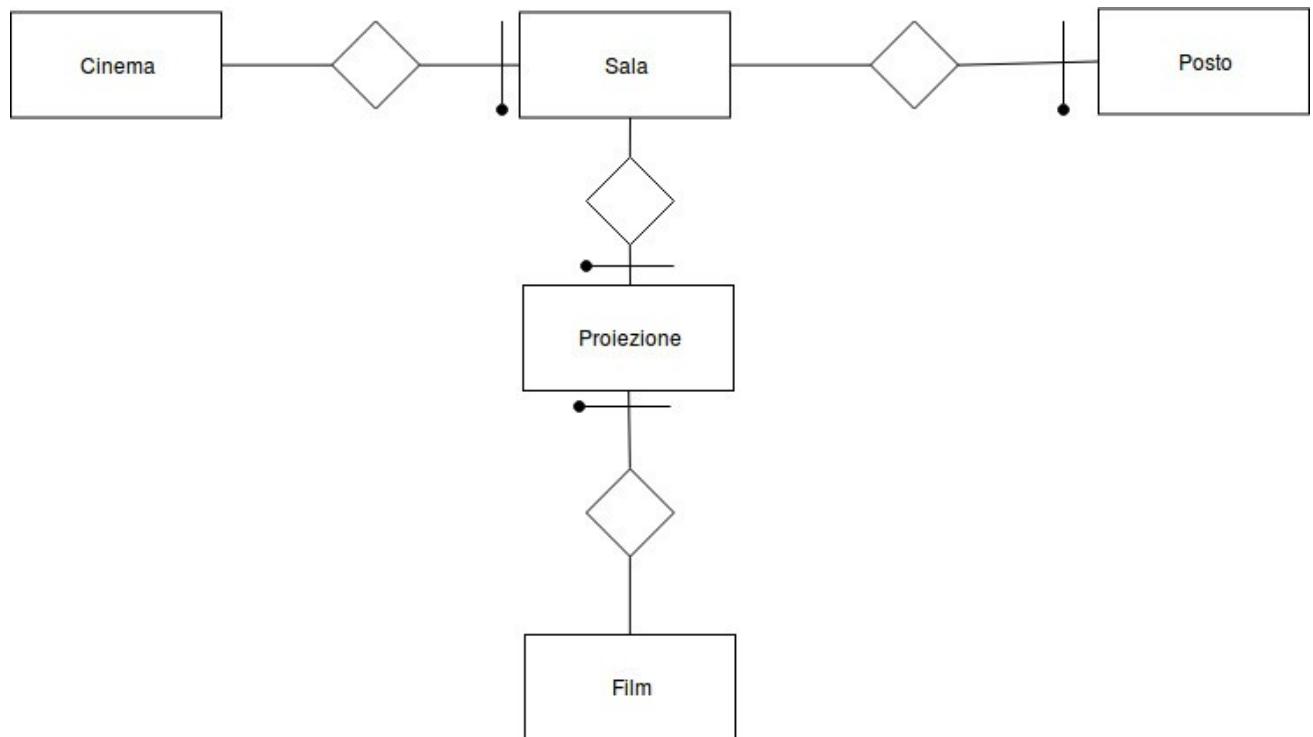


Dato che di ogni sala si vogliono mantenere delle informazioni, tramite il pattern di reificazione di attributo di entità, il costrutto diventa il seguente:

10

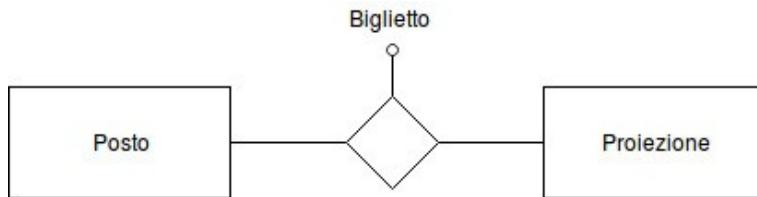


Un ragionamento analogo è stato seguito per legare il concetto di sala con quello di posto, sala e proiezione e proiezioni e film, producendo lo schema seguente:

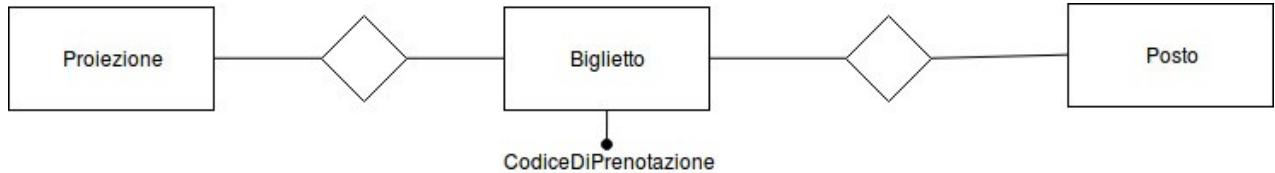


15

Un posto ed una proiezione sono legati tra di loro per mezzo di un biglietto:

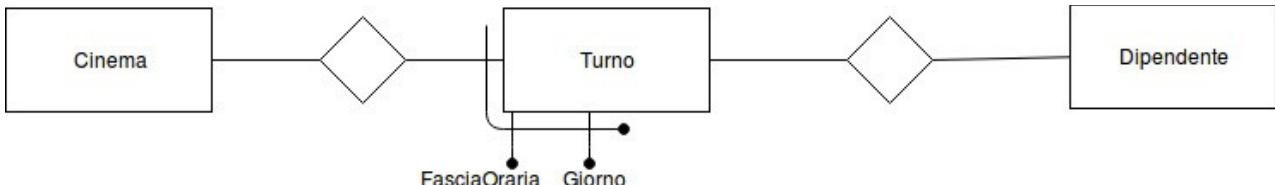


- 5 Dato che si vuole tenere traccia dei biglietti di ogni proiezione, si procede alla reificazione di attributo di relazione binaria, e dato che dalle specifiche è detto che un biglietto è univocamente identificato da un codice di prenotazione, per il momento, si decide di non esprimere l'entità Biglietto come entità debole:



- 10 All'interno dell'entità Biglietto si vuole anche memorizzare la carta di credito con la quale un cliente lo ha acquistato ed anche in questo caso si procede con la reificazione di attributo di entità poiché della carta di credito si vogliono memorizzare altre informazioni oltre il suo numero.

Si vuole inoltre legare in qualche modo il concetto di dipendente con quello di cinema, tramite il concetto di turno.

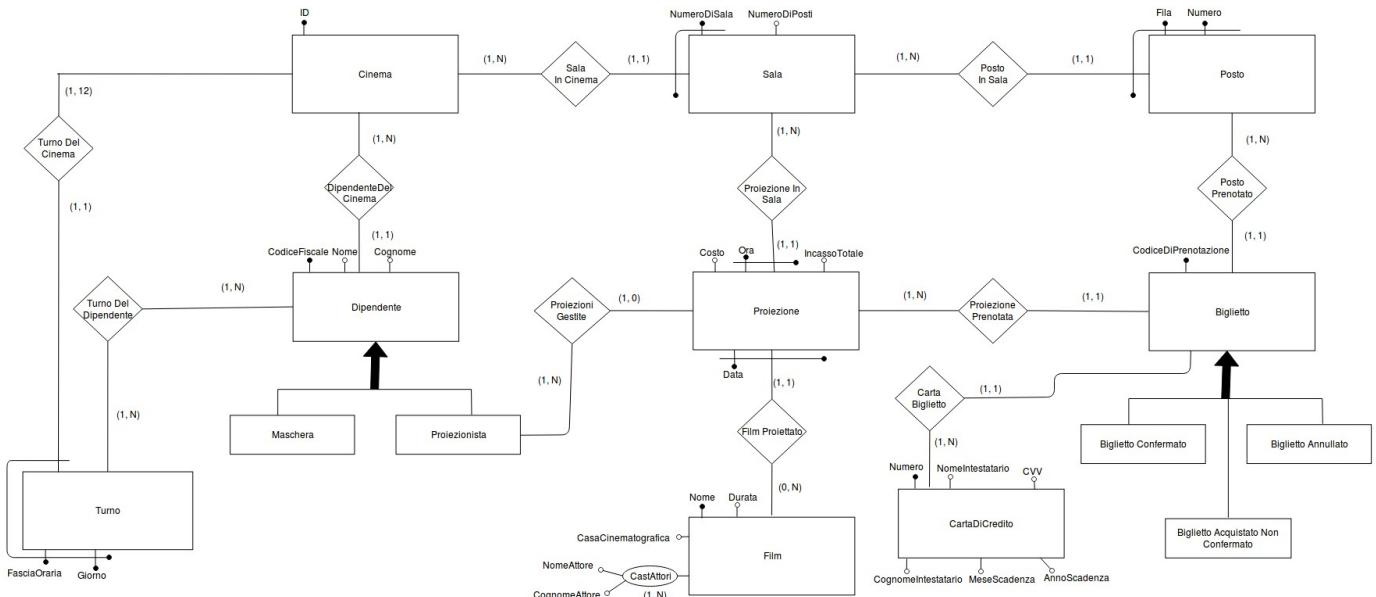


- 15 Questa soluzione permette di controllare se ad una certa fascia oraria di un determinato giorno sono presenti dei determinati dipendenti in un cinema, come richiesto dalle specifiche.

Un dipendente può essere una maschera o un proiezionista. Un biglietto può essere confermato, annullato, o acquistato non confermato. Per racchiudere questi concetti uso il costrutto di generalizzazione.

Infine, lego I concetti principali, aggiungo attributi alle entità e cardinalità alle relazioni.

## Integrazione finale



## Regole aziendali

Nella costruzione dello schema E-R si è supposto che ciascun turno può avere due fasce orarie da 4 ore ciascuna, in questo modo ciascun dipendente può avere al massimo due turni al giorno dalla durata complessiva di 8 ore. Con tale supposizione non è necessario l'inserimento di nessuna regola aziendale per catturare quello che è stato espresso dalle specifiche.

## Dizionario dei dati

Entità	Descrizione	Attributi	Identifieri
Cinema	Luogo in cui vedere film		-ID
Sala	Luogo in cui vengono proiettati I film	-NumeroDiPosti	-NumeroDiSala -Cinema
Posto	Luogo in cui un cliente del cinema si sofferma a vedere il film proiettato		-Fila -Numero -Sala -Cinema
Film	Spettacolo cinematografico	-Durata -CastAttori -Casa Cinematografica	-Nome

Proiezione	Film che è proiettato al cinema in una determinata sala ad una data ora	-IncassoTotale -Costo	-Film -Ora -Sala -Cinema -Data
Biglietto	Lasciapassare per poter assistere ad una proiezione		-CodiceDiPrenotazione
Biglietto Confermato	Biglietto acquistato ed usato per entrare al cinema		
Biglietto Annullato	Biglietto il cui acquisto è stato annullato		
Biglietto Acquistato non confermato	Biglietto acquistato ma non ancora usato per entrare al cinema		
Dipendente	Persona che lavora nel cinema	-Nome -Cognome	-CodiceFiscale
Maschera	Tipo di dipendente adibito al controllo di biglietti all'ingresso del cinema		
Proiezionista	Tipo di dipendente adibito alla gestione delle proiezioni nel cinema		
Turno	Intervallo di tempo di un giorno della settimana in cui lavorano uno o piu' dipendenti del cinema		-Giorno -FasciaOraria -Cinema
CartaDiCredito	Strumento per acquistare biglietti (in questo minimondo)	-Nome Intestatario -Cognome Intestatario -MeseScadenza -AnnoScadenza -CVV	-Numero

## 4. Progettazione logica

### Volume dei dati

N.B.: Di seguito con "settimana lavorativa" si intendono 6 giorni.

Concepto nello schema	Tipo <sup>1</sup>	Volume atteso
Cinema	E	10
Sala	E	80 (8 Sale in media per ogni cinema)
Posto	E	4000 (50 posti in media per sala)
Film	E	300
Proiezione	E	14400 (supponendo una media di 4 proiezioni al gg x 2 settimane lavorative per uno stesso film)
Dipendente	E	200 (20 Dipendenti per cinema)
Maschera	E	50
Proiezionista	E	150
Biglietto	E	288000 (supponendo che ogni proiezione venga proiettata in sale con 50 posti di media, ed una occupazione media della sala del 40%)
Turno	E	120 (10cinema*6gg*2fascieOrarie (pom/sera) e tutti I cinema della catena con stessa fascia oraria supponendo inoltre che I turni della settimana passata vengano cancellati)
Biglietto Confermato	E	244800 (Assumento 85% dei biglietti totali)
Biglietto Annullato	E	28800 (Assumento 10% dei biglietti totali)
Biglietto Acquistato Non Confermato	E	14400 (Assumento 5% dei biglietti totali)
Carta Di Credito	E	48000 (Assumendo che ogni carta venga usata per acquistare 6 biglietti in media)
Sala In Cinema	R	80
Posto In Sala	R	4000
Posto Prenotato	R	288000
Proiezione In Sala	R	14400
Proiezione Prenotata	R	288000
Film Proiettato	R	14400
Proiezioni Gestite	R	14400
Turno Del Dipendente	R	1600 (Ogni dipendente puo avere al piu' 12 turni, pomeriggio e sera tutta la settimana lavorativa, ma in media si

<sup>1</sup>Indicare con E le entità, con R le relazioni

		suppone 8 turni a settimana. Supponendo di voler memorizzare I turni dei dipendenti per una settimana lavorativa, saranno presenti in db 200x8 occorrenze di tale relazione)
Turno Del Cinema	R	120
Dipendente Del Cinema	R	200
Carta Biglietto	R	288000

## Tavola delle operazioni

N.B.: Ogni volta che si fa riferimento a frequenza settimanali, si intende settimana lavorativa (6gg).

Cod.	Descrizione	Frequenza attesa
<b>Operazioni utili ai gestori della catena</b>		
0	Contare il numero di biglietti confermati di un cinema, nell'ultimo mese	1 al mese
1	Contare il numero di biglietti annullati di un cinema, nell'ultimo mese	1 al mese
2	Contare il numero di biglietti acquistati ma non confermati di un cinema, nell'ultimo mese	1 al mese
3	Visualizzare proiezionisti di un cinema che gestiscono una proiezione	15 al giorno (assumendo una media di 15 proiezioni giornaliere)
4	Visualizzare le maschere che per una certa fascia oraria controllano I biglietti all'ingresso del cinema	2 al giorno (un controllo per turno, asusnti due turni al giorno)
5	Contare il numero di biglietti confermati per una certa proiezione in un cinema (gradimento proiezione)	15 al giorno (assumedo 15 proiezioni giornaliere)
6	Contare il numero di biglietti annullati per una certa proiezione in un cinema (ripensamento sulla proiezione)	15 al giorno
7	Incasso di un cinema date da tutte le proiezioni di un film	1 mese
8	Incasso della catena dei cinema dato da tutte le proiezioni di un film	1 mese
9	Incasso totale di un cinema	1 all'anno
10	Incasso totale della catena di cinema	1 all'anno
19	Aggiornamento (modifica/inserimento) del proiezionista per una certa proiezione	5 al giorno (Assumento il 30% delle proiezioni giornaliere, alle quali sarà modificato/aggiunto Il proiezionista)
20	Aggiunta di una maschera in un turno di un giorno per un cinema	2 al giorno
21	Rimozione proiezionista per una proiezione	2 al giorno (assumento circa il 10% delle proiezioni giornaliere)
22	Rimozione maschera in una certa fascia	3 a settimana ( assumo che la frequenza di

	oraria	aggiornamento delle maschere siamo molto inferiore rispetto quella dei proiezionisti)
23	Visualizza proiezionisti (di un cinema) disponibili in un ora del giorno per un determinato periodo di tempo (durata della proiezione la quale gli si vorrebbe eventualmente far gestire)	15 al giorno (assumendo 15 proiezioni giornaliere)
24	Mostra sale disponibili (per un certo intervallo di tempo) in un cinema in un certo orario	15 al giorno
25	Aggiunta di una proiezione	13 al giorno
26	Aggiunta di una proiezione senza fissare un proiezinoista	2 al giorno (assumendo che delle 15 proiezioni, 13 vengano inserite assieme ad un proiezionista e 2 senza, dando la possibilità al gestore di poter aggiungere tale informazione in un secondo momento)
27	Visualizzare il calendario dei turni di un determinato cinema (ovvero I turni assegnati ad ogni dipendente di un determinato cinema, dal giorno in cui si richiede il calendario dei turni, fino alla domenica piu' vicina)	1 a settimana
28	Mostra I turni dei dipendenti di un cinema di un giorno specifico	1 al giorno
29	Inserisci film	8 al mese (assumendo una media di 8 nuovi film usciti ogni mese)
30	Inserisci nuovo attore	100 all'anno (assumendo che ogni anno vengano registrati una media di 100 nuovi attori, ovvero che non sono già presenti nel data base)
31	Mostra tutte le maschere di un cinema	5 al giorno (operazione che puo' essere usata dai gestori come supporto, per facilitare l'assegnazione dei turni)
32	Mostra tutti I proiezionisti di un cinema	15 al giorno (operazione che puo' essere usata dai gestori come supporto, per facilitare l'assegnazione dei turni)

**Operazioni utili ai clienti e visitatori**

11	Visualizzare le proiezioni (film, sala e orario) possibili per un certo film di un certo cinema	1000 al giorno
12	Visualizzare I dettagli di un film (Durata, casa cinematografica)	2000 al giorno
17	Mostra il cast degli attori protagonisti di un film	500 al giorno
18	Visualizza tutti I film del cinema	500 al giorno
13	Visualizzare I posti disponibili per una certa proiezione	600 al giorno

**Operazioni utili ai clienti**

14	Acquisto di un biglietto (per un cinema)	300 al giorno (Assunto un occupazione
----	--	---------------------------------------

		media delle sale del 40%, 20 posti per sala occupati, e 15 proiezioni giornaliere)
15	Annulloamento dell'acquisto di un biglietto (per un cinema)	30 al giorno (10% dei biglietti totali)
<b>Operazioni utili alle maschere</b>		
16	Conferma del biglietto (per un cinema)	255 (assumendo che il 5% dei biglietti (15 al giorno) sia di tipo acquistato non confermato, e tenendo presente anche dei biglietti annullati)

## Costo delle operazioni

<b>Operazione 0, 1, 2</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	360	L
ProiezionePrenotata	R	360	L
Biglietto	E	7200	L

Per eseguire tale operazione bisogna individuare il tipo dei biglietti memorizzati nel data base nell'ultimo mese. Assumo 360 proiezioni mensili (15proiezioni\*6gg\*4) e 7200 accessi per biglietti stimando un

occupazione media delle sale di ogni proiezione del 40% (e sale da 50 posti) 15 proiezioni giornaliere in 4 settimane lavorative ( $0.4 \times 50 \times 15 \times 4 \times 6 = 7200$ ). Selezionati I biglietti per l'Operazione 0 conto I confermati, per l'operazione 1 conto gli annullati e per l'Operazione 2 conto gli acquistati ma non confermati. Costo Totale: 7920 accessi al mese.

<b>Operazione 3</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1	L
ProiezioneGestita	R	1	L
Proiezionista	E	1	L

Nota la proiezione mostro il dipendente che la gestisce. Costo Totale:  $3 \times 15 = 45$  accessi al giorno.

<b>Operazione 4</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Cinema	E	1	L
TurnoDelCinema	R	1	L
Turno	E	2	L
TurnoDelDipendente	R	15	L
Dipendente	E	15	L

Noti I turni di un cinema, con giorno e fascia oraria, cerco I dipendenti che lavorano in quel turno (nel caso peggiore posso averne al più 15, ovvero tutti I dipendenti del cinema) e da questi seleziono le maschere. Costo Totale:  $34 \times 2 = 68$  accessi al giorno.

<b>Operazione 5, 6</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1	L
ProiezionePrenotata	R	20	L
Biglietto	E	20	L

Metodo di ricerca analogo alle operazioni 0,1,2 ma riferita ad una sola proiezione. Costo Totale:  $41 \times 15 = 615$

5 accessi al giorno.

<b>Operazione 7</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	48	L

Ipotizzo 4 proiezioni giornaliere per ogni film per 2 settimane lavorative. Costo Totale: 48 accessi al mese.

<b>Operazione 8</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	480	L

Come operazione 7 ma cercando le proiezioni associate ad ogni cinema. Costo Totale: 480 accessi al mese.

<b>Operazione 9</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1440	L

Asusmendo 14400 proiezioni totali nella catena dei cinema, e 10 cinema in totale registrati nel database. Costo Totale: 14400 accessi all'anno.

<b>Operazione 10</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	14400	L

10 Come operazione 9. Costo Totale: 14400 accessi all'anno.

<b>Operazione 11</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	48	L

Costo Totale:  $48 \times 1000 = 48000$  accessi al giorno.

<b>Operazione 12</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Film	E	1	L

Costo Totale:  $1 \times 2000 = 2000$  accessi al giorno.

<b>Operazione 13</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1	L
ProiezionePrenotata	R	50	L
Biglietto	E	50	L
PostoPrenotato	R	50	L
Posto	E	50	L

Individuata la proiezione, vengono distinti i posti disponibili da quelli non disponibili vedendo se non esistono codici di prenotazione nell'entità Biglietto per la prenotazione selezionata o tali per cui il tipo è annullato, 5 vedendo il posto a cui tali biglietti sono associati (nel caso peggiore per ogni prenotazione saranno associati 50 biglietti in media). Costo Totale:  $201 \times 600 = 120600$  accessi al giorno.

<b>Operazione 14</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Biglietto	E	1	S
PostoPrenotato	R	1	S
ProiezionePrenotata	R	1	S
Proiezione	E	1	L
Proiezione	E	1	S

Dopo aver memorizzato il biglietto, viene aggiornato l'incasso totale all'proiezione per la quale si è comprato il biglietto. Costo Totale:  $9 \times 300 = 2700$  accessi al giorno.

<b>Operazione 15</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Biglietto	E	1	S
ProiezionePrenotata	R	1	L
Proiezione	E	1	L
Proiezione	E	1	S

Dato il codice di prenotazione del biglietto, viene modificato il tipo del biglietto in annullato, poi viene aggiornato il valore dell'incasso totale della poiezione associata a quel biglietto Costo Totale:  $6 \times 30 = 180$  accessi al giorno.

<b>Operazione 16</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Biglietto	E	1	L

Dato il codice di prenotazione del biglietto viene verificato che il tipo non sia annullato o (già) confermato e  
5 in caso affermativo viene modificato in confermato. Costo Totale:  $1 \times 255 = 255$  accessi al giorno.

<b>Operazione 17</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Film	R	8	L

Dato il film mostro il nome e cognome degli attori ad esso associato (8 attori protagonisti in media per ogni film). Costo Totale:  $8 \times 500 = 4000$  accessi al giorno.

<b>Operazione 18</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Film	E	300	L

Costo Totale:  $300 \times 500 = 150000$  accessi al giorno.

<b>Operazione 19</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
ProiezioneGestita	R	1	S

Noti proiezione e il nuovo proiezionista a cui voler far gestire la proiezione, viene modificata l'associazione  
10 tra proiezionista e proiezione. Costo Totale:  $2 \times 5 = 10$  accessi al giorno.

<b>Operazione 20</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
TurnoDelDipendente	R	1	S

Costo Totale:  $2 \times 2 = 4$  accessi al giorno.

<b>Operazione 21</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1	S

Costo Totale:  $2 \times 2 = 4$  accessi al giorno.

<b>Operazione 22</b>			
----------------------	--	--	--

Concetto	Costrutto	Accessi	Tipo
TurnoDelDipendente	R	1	S

Costo Totale:  $2 \times 3 = 6$  accessi a settimana.

Operazione 23			
Concetto	Costrutto	Accessi	Tipo
Cinema	E	1	L
DipendenteDelCinema	R	1	L
Dipendente	E	15	L
ProiezioniGestite	R	96	L
Proiezione	E	96	L
FilmProiettato	R	96	L
Film	E	96	L

Da tutti I dipendenti di un cinema, elimino quelli che sono associati ad una proiezione la cui ora di inizio più durata si intersecano con l'ora di inizio e durata del film alla quale lo si vorrebbe assegnare. (si assume una media di 96 proiezioni associate ad ogni dipendente di tipo proiezionista, dato dalle proiezioni di un cinema diviso il numero di proiezionisti di un cinema a 1440/15) Costo Totale:  $401 \times 15 = 6015$  accessi al giorno.

Operazione 24			
Concetto	Costrutto	Accessi	Tipo
Sala	E	8	L
ProiezioneInSala	R	16	L
Proiezione	E	16	L
FilmProiettato	R	16	L
Film	E	16	L

Da tutte le sale di un cinema, elimino quelle che sono associate ad una proiezione (di un certo giorno) la cui ora di inizio più durata si intersecano con l'ora di inizio e durata del film alla quale lo si vorrebbe assegnare. Assumendo al più 2 proiezioni giornaliere associate ad ogni sala ho  $8 \times 2 = 16$  accesso all'entità proiezione. (Per ogni proiezione devo ricavare la durata del film dall'entità proiezione) Costo Totale:  $72 \times 15 = 1080$  accessi al giorno.

Operazione 25			
Concetto	Costrutto	Accessi	Tipo
Proiezione	E	1	S

Costo Totale:  $2 \times 15 = 30$  accessi al giorno.

#### Operazione 26

<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1	S

Costo Totale:  $2 \times 2 = 4$  accessi al giorno.

<b>Operazione 27</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Cinema	E	1	L
DipendenteDelCinema	R	20	L
Dipendente	E	20	L
TurnoDelDipendente	R	160	L
Turno	E	120	L

Mostra I turni dei dipendenti di un cinema per una settimana lavorativa (6 gg). Assumo che ogni dipendente del cinema abbia in media 8 turni settimanali, quindi  $8 \text{turni} \times 20 \text{dipendenti} = 160$  accessi a turno del dipendente. Costo Totale:  $321 \times 1 = 321$  accessi a settimana.

<b>Operazione 28</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Turno	E	1	L
TurnoDelDipendente	R	15	L
Dipendente	E	15	L

- 5 Si assume una media di 15 dipendenti al giorno che lavorano in un cinema. Costo Totale:  $31 \times 1 = 31$  accessi al giorno.

<b>Operazione 29</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Film	E	1	S

Costo Totale:  $2 \times 8 = 16$  accessi al mese.

<b>Operazione 30</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Film	E	1	S

Costo Totale:  $2 \times 100 = 200$  accessi all'anno.

<b>Operazione 31</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Cinema	E	1	L
DipendenteDelCinema	R	20	L

Dipendente	E	20	L
------------	---	----	---

Costo Totale:  $41 \times 5 = 1025$  accessi al giorno.

<b>Operazione 32</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Cinema	E	1	L
DipendenteDelCinema	R	20	L
Dipendente	E	20	L

Costo Totale:  $41 \times 15 = 615$  accessi al giorno.

## Ristrutturazione dello schema E-R

Scelgo di trasformare l'entità 'Biglietto' come identità debole, identificandolo dalle entità 'Posto' e 'Proiezione' (anche se è sufficiente che Biglietto sia identificato dal solo attributo 'Codice Di Prenotazione'). In questo modo le Operazioni 0, 1, 2 cambierebbero come di seguito:

<b>Operazione 0, 1, 2</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Biglietto	E	7200	L

Dato che biglietto è identificato dalla proiezione, conterrà l'attributo 'data'. Verrà quindi usata la data per cercare i biglietti di un cinema dell'ultimo mese. Costo Totale: 7200 accessi al mese. L'aggiunta degli attributi comporta l'aggiunta in memoria di  $(4+4+4+1+20+8)$ Byte\*288000 Biglietti, circa 11 Mbyte ed un risparmio di 720 accessi al mese.

Cambierebbero anche le Operazione 5, 6:

<b>Operazione 5, 6</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Biglietto	E	20	L

Assumo un occupazione della sala del 40%. Dai biglietti associati alla prenotazione, cerco quelli confermati per l'Operazione 5 e quelli annullati per l'operazione 6. Costo Totale:  $20 \times 15 = 300$  accessi al giorno, con un risparmio di 315 accessi giornalieri.

15

L'attributo 'Incasso Totale' nell'entità 'Proiezione', potrebbe essere ricavato andando a contare il costo dei biglietti associati a ciascuna proiezione. In caso di eliminazione di tale attributo le Operazioni 7, 8, 9, 10, 14, 15 cambierebbero come segue:

<b>Operazione 7</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	48	L

Biglietto	E	960	L
-----------	---	-----	---

Dopo la ristrutturazione precedente, è possibile usare in ‘Biglietto’ l’attributo ‘NomeFilm’ per effettuare delle query senza passare per la relazione ProiezionePrenotata. Ipotizzo 4 proiezioni giornaliere per ogni film per 2 settimane lavorative (48 proiezioni per film, ognuna può avere un prezzo diverso), e per ogni proiezione assumo che l’occupazione dei posti sia del 40%. Costo Totale: 1008 accessi al mese.

<b>Operazione 8</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	480	
Biglietto	E	9600	L

5 Costo Totale: 10080 accessi al mese.

<b>Operazione 9</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	1440	L
Biglietto	E	24480	L

Dopo la ristrutturazione precedente, è possibile usare in ‘Biglietto’ l’attributo ‘Cinema’ per effettuare delle query. Il numero degli accessi è dato dai biglietti confermati presenti (nel database stimo un numero di biglietti di 244800), diviso il numero di cinema. Costo Totale: 25920 accessi all’anno.

<b>Operazione 10</b>			
<b>Concetto</b>	<b>Costrutto</b>	<b>Accessi</b>	<b>Tipo</b>
Proiezione	E	14400	L
Biglietto	E	244800	L

Costo Totale: 259200 accessi all’anno.

10 In conclusione se si decidesse di eliminare l’attributo ‘Incasso Totale’ si guadagnerebbero 4Byte\*14400 proiezioni, circa 57 Kbyte, aumentando il numero degli accessi di 10032 al mese e 253440 accessi all’anno.

Tuttavia l’eliminazione farebbe risparmiare degli accessi alle Operazioni 14 e 15. Infatti, in tali operazioni, non verrebbe effettuato l’aggiornamento dell’incasso totale risparmiando 1 accesso in  
15 lettura ed uno in scrittura in ‘Proiezione’. In particolare:

Accessi mensili con ridondanza:  $48 + 480 = 528$

Accessi mensili senza ridondanza:  $1008 + 10080 = 11088$

Accesso annuali con ridondanza:  $1440 + 14400 = 15840$

20 Accessi annuali senza ridondanza:  $25920 + 259200 = 285120$

Accessi giornalieri con ridondanza:  $3300 + 180 = 3480$

Accessi giornalieri senza ridondanza:  $2400 + 90 = 2490$

Riportando tutto in mesi, accessi con ridondanza = 85368 accessi mensili.

Accessi senza ridondanza: 94608 accessi mensili.

In conclusione conviene mantenere la ridondanza.

5

Per quanto riguarda le generalizzazioni, sia per l'Entità 'Dipendente' e 'Biglietto', inglobo gli attributi delle entità figlie, in quelle del padre. Nell'entità 'Dipendente' verrà aggiunto un attributo 'Tipo' il quale potrà essere 'Proiezionista' o 'Maschera'. Nell'entità 'Biglietto' verrà aggiunto un attributo 'Tipo' il quale potrà essere 'Confermato' o 'AcquistatoNonConfemrato' o 'Annullato'.

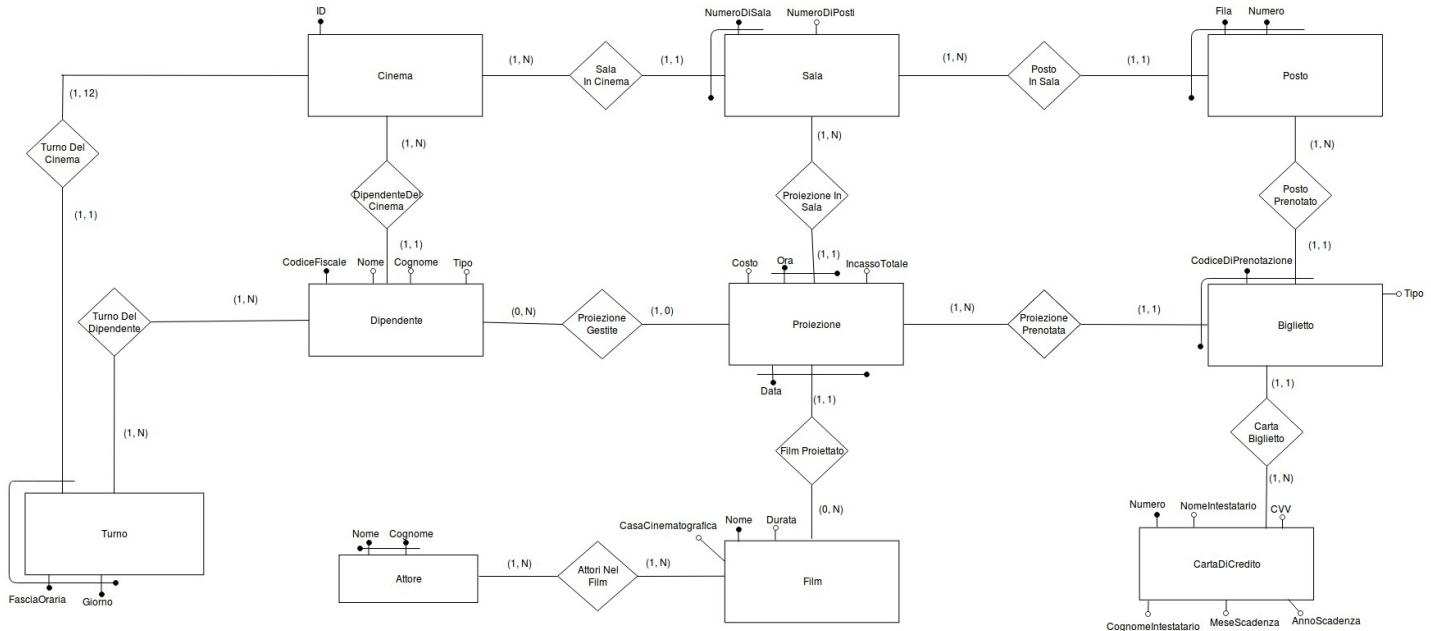
10 Inoltre l'associazione tra l'entità 'Proiezionista' e 'Proiezione' che nel modello E-R aveva cardinalità (1, N), ora dovrà avere cardinalità (0, N) in quanto I dipendenti registrati con attributo 'Tipo' uguale a 'Maschera' non potranno parteciparvi.

Procedo inoltre, con l'eliminazione dell'attribbuto composto e multivalore 'CastAttori' nell'entità 'Film' inserendo la nuova associazione 'AttoriNelFilm' tra l'entità 'Film' e la nuova entità 'Attori'.

15 Per quanto riguarda la scelta degli identificatori principali, ogni entità ha un unico identificatore.

Potrei decidere di scegliere di separare alcuni attributi nell'identificatore dell'entità 'Proiezione' e 'Biglietto' in modo da alleggerire, in fase implementativa, gli indici che verranno usati per effettuare le query. Tuttavia tutti gli attributi degli identificatori appena citati sono usati in almeno un operazione. Decido quindi di non separare alcune chiavi dagli identificatori principali, a scapito della dimensioone degli indici, per motivi prestazionali.

Lo schema E-R ristruttorato è il seguente:



## Trasformazione di attributi e identificatori

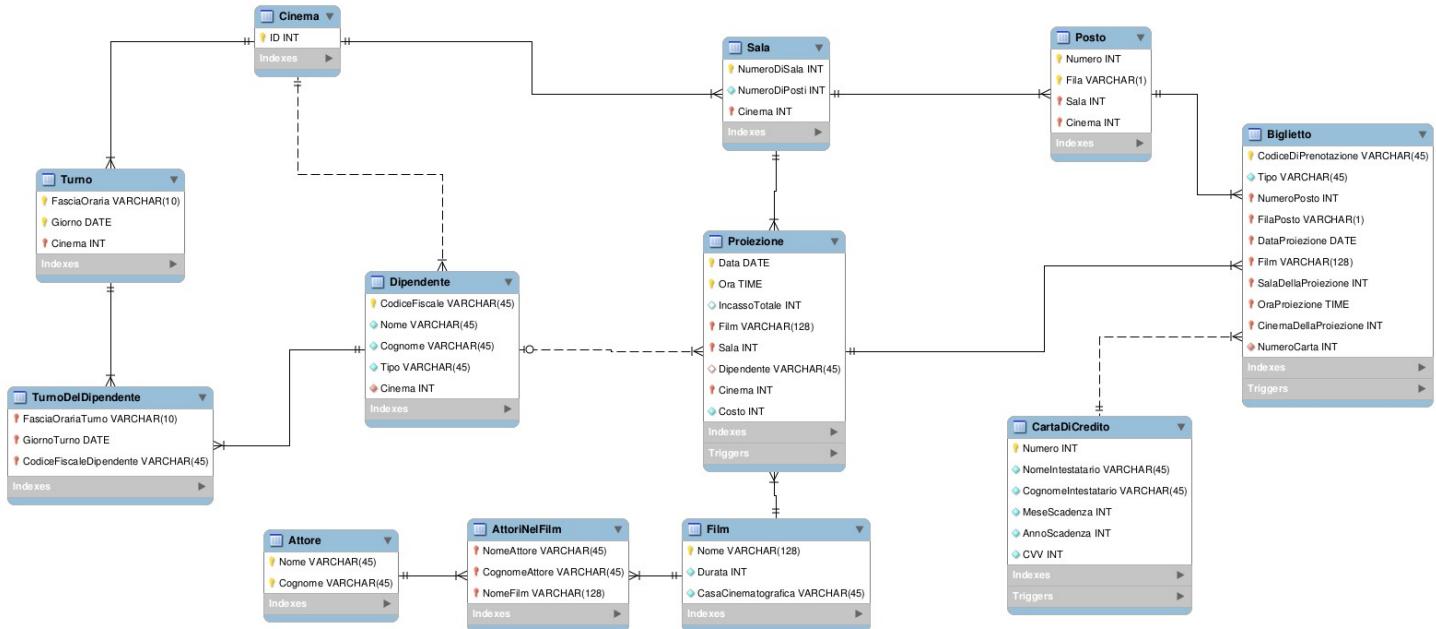
All'interno dell'entità 'Biglietto' vengono ripetuti due volte gli attributi 'Sala' e 'Cinema', in quanto

"ereditati" dalle entità 'Posto' e 'Proiezione'. Per alleggerire l'indice primario che verrà usato

5 sull'entità 'Biglietto', tali attributi ereditari da 'Posto' non li inserisco nell'identificatore principale

di 'Biglietto'. Dunque la relazione tra 'Biglietto' e 'Posto' non è più identificante.

## Traduzione di entità e associazioni



## Normalizzazione del modello relazionale

Il modello relazionale è già in forma normale.

## 5. Progettazione fisica

### Utenti e privilegi

Gli utenti previsti all'interno del sistema sono I seguenti.

Gestore: possiede tutti I privilegi su tutte le tabelle del database.

- 5 Maschera: possiede tutti i privilegi sulla tabella “Biglietto”, in quanto ha la possibilità, e il dovere, di segnare I biglietti all'ingresso del cinema come “Confermati”.

Visitatore: ha privilegi di lettura sulle tabelle “Cinema”, “Sala”, “Posto”, “Proiezione”, “Film”, “Attore”, “AttoriNelFilm”, in modo da poter visualizzare le informazioni necessarie per far sì che possa acquistare un biglietto per una certa proiezione.

- 10 Cliente: ha privilegi di lettura sulle tabelle “Cinema”, “Sala”, “Posto”, “Proiezione”, “Film”, “Attore”, “AttoriNelFilm”, in modo da poter visualizzare le informazioni necessarie per aiutarlo nella scelta d'acquisto di un biglietto per una proiezione. Ha privilegi in scrittura sulle tabelle che vengono utilizzate nelle procedure “acquistaBiglietto” e “annullaAcquisto” (ovvero le tabelle “Biglietto”, “Proiezione” e “CartaDiCredito”), per consentire la memorizzazione nel database delle 15 informazioni sul biglietto acquistato, o modificarlo in annullato.

### Strutture di memorizzazione

<b>Tabella &lt;Attore&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi<sup>2</sup></b>
Nome	VARCHAR(45)	PK, NN
Cognome	VARCHAR(45)	PK, NN

<b>Tabella &lt;Film&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
Nome	VARCHAR(128)	PK, NN
Durata	INT	NN

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

CasaCinematografica	VARCHAR(45)	NN
---------------------	-------------	----

<b>Tabella &lt;AttoriNelFilm&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
NomeAttore	VARCHAR(45)	PK, NN
CognomeAttore	VARCHAR(45)	PK, NN
NomeFilm	VARCHAR(45)	PK, NN

<b>Tabella &lt;Cinema&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
ID	INT	PK, NN

<b>Tabella &lt;Sala&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
NumerodiSala	INT	PK, NN
NumerodiPosti	INT	NN
Cinema	INT	PK, NN

<b>Tabella &lt;Posto&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
Numero	INT	PK, NN
Fila	VARCHAR(1)	PK, NN
Sala	INT	PK, NN
Cinema	INT	PK, NN

5

<b>Tabella &lt;Turno&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
FasciaOraria	VARCHAR(10)	PK, NN
Giorno	DATE	PK, NN
Cinema	INT	PK, NN

<b>Tabella &lt;Dipendente&gt;</b>		
<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
Nome	VARCHAR(45)	NN
Cognome	VARCHAR(45)	NN
Tipo	VARCHAR(45)	NN
Cinema	INT	NN
CodiceFiscale	VARCHAR(45)	PK, NN

**Tabella <TurnoDelDipendente>**

<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
FasciaOrariaTurno	VARCHAR(10)	PK, NN
GiornoTurno	DATE	PK, NN
CodiceFiscaleDipendente	VARCHAR(45)	PK, NN

**Tabella <Proiezione>**

<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
Data	DATE	PK, NN
Ora	TIME	PK, NN
IncassoTotale	INT	
Sala	INT	PK, NN
Dipendente	VARCHAR(45)	
Cinema	INT	PK, NN
Costo	INT	NN

**Tabella <CartaDiCredito>**

<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi<sup>3</sup></b>
Numero	VARCHAR(16)	PK, NN, UN
CognomeIntestatario	VARCHAR(45)	NN
NomeIntestatario	VARCHAR(45)	NN
MeseScadenza	INT	NN, UN
AnnoScadenza	INT	NN, UN
CVV	INT	NN, UN

**Tabella <Biglietto>**

<b>Attributo</b>	<b>Tipo di dato</b>	<b>Attributi</b>
CodiceDiPrenotazione	VARCHAR(45)	PK, NN
Tipo	VARCHAR(45)	NN
NumeroPosto	INT	PK, NN
FilaPosto	VARCHAR(1)	PK, NN
DataProiezione	DATE	PK, NN
Film	VARCHAR(128)	PK, NN
SalaDellaProiezione	INT	PK, NN
OraProiezione	TIME	PK, NN
CinemaDellaProiezione	INT	PK, NN
NumeroCarta	VARCHAR(16)	NN

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

## Indici

I seguenti indici sono stati automaticamente generati da MySQL Workbench. Non si è ritenuto necessario l'aggiunta di ulteriori indici.

5

<b>Tabella &lt;Attore&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo<sup>4</sup>:</b>
Nome, Cognome	PR

<b>Tabella &lt;Film&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
Nome	PR

<b>Tabella &lt;AttoriNelFilm&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
NomeAttore, CognomeAttore, NomeFilm	PR

<b>Tabella &lt;AttoriNelFilm&gt;</b>	
<b>Indice &lt;fk_Attore_has_Film_Film1_idx&gt;</b>	<b>Tipo:</b>
NomeFilm	IDX

<b>Tabella &lt;AttoriNelFilm&gt;</b>	
<b>Indice &lt;fk_Attore_has_Film_Attore_idx&gt;</b>	<b>Tipo:</b>
NomeAttore, CognomeAttore	IDX

10

<b>Tabella &lt;Cinema&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
ID	PR

<b>Tabella &lt;Sala&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
NumeroSala, Cinema	PR

4IDX = index, UQ = unique, FT = full text, PR = primary.

<b>Tabella &lt;Sala&gt;</b>	
<b>Indice &lt;fk_Sala_Cinema1_idx&gt;</b>	<b>Tipo:</b>
Cinema	IDX

<b>Tabella &lt;Posto&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
Cinema, Sala, Fila, Numero	PR

<b>Tabella &lt;Posto&gt;</b>	
<b>Indice &lt;fk_Posto_Sala1_idx&gt;</b>	<b>Tipo:</b>
Cinema, Sala	IDX

<b>Tabella &lt;Turno&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
FasciaOraria, Giorno, Cinema	PR

<b>Tabella &lt;Turno&gt;</b>	
<b>Indice &lt;fk_Turno_Cinema1_idx&gt;</b>	<b>Tipo:</b>
Cinema	IDX

5

<b>Tabella &lt;Dipendente&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
CodiceFiscale	PR

<b>Tabella &lt;Dipendente&gt;</b>	
<b>Indice &lt;fk_Dipendente_Cinema1_idx&gt;</b>	<b>Tipo:</b>
Cinema	IDX

<b>Tabella &lt;TurnoDelDipendente&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
FasciaOrariaTurno, GiornoTurno, CodiceFiscaleDipendente	PR

<b>Tabella &lt;TurnoDelDipendente&gt;</b>	
<b>Indice &lt;fk_Turno_has_Dipendente_Dipendente1_idx&gt;</b>	<b>Tipo:</b>
CodiceFiscaleDipendente	IDX

<b>Tabella &lt;Proiezione&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
Data, Ora, Film, Sala, Cinema	PR

10

<b>Tabella &lt;Proiezione&gt;</b>	
<b>Indice &lt;fk_Proiezione_Film1_idx&gt;</b>	<b>Tipo:</b>
Film	IDX

<b>Tabella &lt;Proiezione&gt;</b>	
<b>Indice &lt;fk_Proiezione_Sala1_idx&gt;</b>	<b>Tipo:</b>
Sala, Cinema	IDX

<b>Tabella &lt;Proiezione&gt;</b>	
<b>Indice &lt;fk_Proiezione_Dipendente1_idx&gt;</b>	<b>Tipo:</b>
Dipendente	IDX

<b>Tabella &lt;CartaDiCredito&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
Numero	PR

<b>Tabella &lt;Biglietto&gt;</b>	
<b>Indice &lt;PRIMARY&gt;</b>	<b>Tipo:</b>
CodiceDiPrenotazione, NumeroPosto, FilaPosto, DataPrenotazione, Film, SalaDellaProiezione, OraProiezione, CinemaDellaProiezione	PR

5

<b>Tabella &lt;Biglietto&gt;</b>	
<b>Indice &lt;fk_Biglietto_Posto1_idx&gt;</b>	<b>Tipo:</b>
NumeroPosto, FilaPosto	IDX

<b>Tabella &lt;Biglietto&gt;</b>	
<b>Indice &lt;fk_Biglietto_CartaDiCredito1_idx&gt;</b>	<b>Tipo:</b>
NumeroCarta	IDX

<b>Tabella &lt;Biglietto&gt;</b>	
<b>Indice &lt;fk_Biglietto_Proiezione1_idx&gt;</b>	<b>Tipo:</b>
DataProiezione, Film, SalaDellaProiezione, OraProiezione, CinemaDellaProiezione	IDX

## Trigger

Trigger di tipo BEFORE INSERT della tabella ‘Proiezione’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Proiezione_BEFORE_INSERT` BEFORE INSERT ON `Proiezione` FOR EACH ROW
2 BEGIN
3
4     declare pom varchar(10) default '16-20';
5     declare ser varchar(10) default '20-24';
6     declare entrambi int default 0;
7     declare maschereDisponibili int;
8     declare maschereDisponibiliSeriali int;
9     declare maschereDisponibiliPomeridiane int;
10    declare mascheraCasuale varchar(45);
11    declare durata_p int;
12    declare fasciaOraria varchar(10);
13
14    #se un proiezionista viene associato ad una proiezione, si deve verificare che
15    #quel giorno a quell'ora sia presente nella tabella 'TurnoDelDipendente', altrimenti vi deve essere aggiunto
16    #e che quel giorno (data della proiezione) con una certa fascia oraria (da determinare in base
17    #all'ora della proiezione) deve essere presente nella tabella 'Turno'
18
19    select F.Durata
20    from Film F
21    where F.Nome = NEW.Film
22    into durata_p;
23
24    #identifico la faccia oraria in base all'ora della proiezione
25    if ('16:00:00' <= NEW.Ora and NEW.Ora <= '20:00:00') then
26        set fasciaOraria = pom;
27    end if;
28    if ('20:00:00' <= NEW.Ora and NEW.Ora <= '24:00:00') then
29        set fasciaOraria = ser;
30    end if;
31    #se ho una proiezione che si sovrappone tra i 2 turni
32    if (NEW.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(NEW.Ora, SEC_TO_TIME(durata_p*60))) then
33        set entrambi = 1;
34    end if;
35    if ('24:00:00' < NEW.Ora or NEW.Ora < '16:00:00') then
36        signal sqlstate '45009'
37        set message_text = 'Ora della proiezione non valida, deve essere compresa tra le 16 e le 24';
38    end if;
39
40    if NEW.Dipendente is not null then
41        if not entrambi then
42            #prima verifico che sia presente nella tabella Turno la tupla:
43            if ((select FasciaOraria
44                from Turno T
45                where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = fasciaOraria)
46
47                is null) then
48                    #se non è presente, la creo
49                    insert into Turno values (fasciaOraria, NEW.Data, NEW.Cinema);
50                end if;
51                #se è presente devo verificare che il dipendente (se è stato selezionato per la proiezione
52                #allora sicuramente 'libero' per come funzionano le procedure sviluppate) sia associato a quel turno
53                #e se non lo fosse lo associo
54                if ((select CodiceFiscaleDipendente
55                    from TurnoDelDipendente TD
56                    where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and TD.FasciaOrariaTurno = fasciaOraria)
57                    is null) then
58                    insert into TurnoDelDipendente
59                    values (fasciaOraria, NEW.Data, NEW.Dipendente);
60                end if;
61
62                #faccio in modo che ci siano almeno 2 maschere differenti per la fascia oraria in cui ho la proiezione
63                select count(distinct TD.CodiceFiscaleDipendente)
64                from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
65                where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = fasciaOraria and D.Tipo = 'Maschera'
66                into maschereDisponibili;
67                #per tante volte quante sono le maschere mancanti per arrivare a 2
68                #aggiungo al turno una maschera a caso disponibile.
69                #definisco una maschera disponibile per un cinema se:
70                #1)lavora per quel cinema
71                #2)non ha un turno associato alla fascia oraria della proiezione
72                while maschereDisponibili < 2 do
73
74                    select CodiceFiscale
75                    from Dipendente
76                    where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
77                        select D.CodiceFiscale
78                        from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale = TD.CodiceFiscaleDipendente
79                            and D.Cinema = NEW.Cinema
80                            where TD.FasciaOrariaTurno = fasciaOraria and TD.GiornoTurno = NEW.Data and D.Tipo = 'Maschera')
81                    order by RAND() limit 1 into mascheraCasuale;
```

```

82     insert into TurnoDelDipendente
83         values (fasciaOraria, NEW.Data, mascheraCasuale);
84
85         set maschereDisponibili = maschereDisponibili +1;
86
87     end while;
88 #se il proiezionista deve gestire una proiezione tale per cui gli occupa entrambi i turni della giornata:
89 else
90     #devo verificare che sia presente sia il turno notturno sia quello serale
91
92
93     if ((select FasciaOraria
94         from Turno T
95         where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = pom)
96         is null) then
97         insert into Turno values (pom, NEW.Data, NEW.Cinema);
98     end if;
99     if ((select FasciaOraria
100        from Turno T
101        where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = ser)
102         is null) then
103         insert into Turno values (ser, NEW.Data, NEW.Cinema);
104     end if;
105    #resta da verificare che il dipendente scelto per la gestione della proiezione
106    #sia associato ai turni sia serale che pomeridiano
107    if ((select CodiceFiscaleDipendente
108        from TurnoDelDipendente TD
109        where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and TD.FasciaOrariaTurno = pom)
110        is null )then
111        insert into TurnoDelDipendente
112            values (pom, NEW.Data, NEW.Dipendente);
113    end if;
114    if ((select CodiceFiscaleDipendente
115        from TurnoDelDipendente TD
116        where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and TD.FasciaOrariaTurno = ser)
117        is null )then
118        insert into TurnoDelDipendente
119            values (ser, NEW.Data, NEW.Dipendente);
120    end if;
121    #stessa cosa per le maschere, almeno 2 per il turno di sera e quello pomeridiano
122    select count(distinct TD.CodiceFiscaleDipendente)
123    from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
124    where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = ser and D.Tipo = 'Maschera'
125    into maschereDisponibiliSerale;
126
127    select count(distinct TD.CodiceFiscaleDipendente)
128    from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
129    where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = pom and D.Tipo = 'Maschera'
130    into maschereDisponibiliPomeridiane;
131
132    while maschereDisponibiliSerale < 2 do
133
134        select CodiceFiscale
135        from Dipendente
136        where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
137
138            from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale = TD.CodiceFiscaleDipendente
139                and D.Cinema = NEW.Cinema
140                where TD.FasciaOrariaTurno = ser and TD.GiornoTurno = NEW.Data and D.Tipo = 'Maschera')
141                order by RAND() limit 1 into mascheraCasuale;
142
143        insert into TurnoDelDipendente
144            values (ser, NEW.Data, mascheraCasuale);
145
146        set maschereDisponibiliSerale = maschereDisponibiliSerale +1;
147
148    end while;
149    while maschereDisponibiliPomeridiane < 2 do
150
151        select CodiceFiscale
152        from Dipendente
153        where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
154            select D.CodiceFiscale
155            from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale = TD.CodiceFiscaleDipendente
156                and D.Cinema = NEW.Cinema
157                where TD.FasciaOrariaTurno = pom and TD.GiornoTurno = NEW.Data and D.Tipo = 'Maschera')
158                order by RAND() limit 1 into mascheraCasuale;
159
160        insert into TurnoDelDipendente
161            values (pom, NEW.Data, mascheraCasuale);
162
163        set maschereDisponibiliPomeridiane = maschereDisponibiliPomeridiane +1;
164
165    end while;
166    end if;
167
168    end if;
169 END

```

Trigger di tipo AFTER UPDATE della tabella ‘Proiezione’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Proiezione_AFTER_UPDATE` AFTER UPDATE ON `Proiezione` FOR EACH ROW
2 BEGIN
3
4     declare pom varchar(10) default '16-20';
5     declare ser varchar(10) default '20-24';
6     declare fasciaOraria varchar(10);
7     declare entrambi int default 0;
8     declare durata_p int;
9
10    declare maschereDisponibili int;
11    declare maschereDisponibiliSerali int;
12    declare maschereDisponibiliPomeridiane int;
13    declare mascheraCasuale varchar(45);
14
15    if (OLD.Dipendente is not null) then
16
17        select F.Durata
18        from Film F
19        where F.Nome = OLD.Film
20        into durata_p;
21
22        if ('16:00:00' <= OLD.Ora and OLD.Ora <= '20:00:00') then
23            set fasciaOraria = pom;
24        end if;
25        if ('20:00:00' <= OLD.Ora and OLD.Ora <= '24:00:00') then
26            set fasciaOraria = ser;
27        end if;
28        #se ho una proiezione che si sovrappone tra i 2 turni
29        if (OLD.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(OLD.Ora, SEC_TO_TIME(durata_p*60))) then
30            set entrambi = 1;
31        end if;
32
33        # se il proiezionista non ha associate ALTRE proiezioni (ora, sala etc.. differenti) per quel giorno a quella fascia oraria
34        # gli cancello il turno, o entrambi i turni, qualora la proiezione che verra occupata gli occupi
35        # entrambi i turni
36        if not entrambi then
37
38            if ((select count(*)
39                from Proiezione P join Film F on P.Film = F.Nome
40                where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
41                # calcolo la faccia oraria per quel film
42                fasciaOraria = (case
43                    # se in quel giorno il proiezionista ha un altro film che gli occupa
44                    # entrambi i turni, non voglio cancellarglieli
45
46
47                when '16:00:00' <= P.Ora and P.Ora <= '20:00:00' then pom
48                when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
49            end)) = 0) then
50
51            #se e 0 elimino il turno del dipendente a quella fascia oraria
52            delete
53            from TurnoDelDipendente
54            where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = fasciaOraria and GiornoTurno = OLD.Data;
55
56            #verifico che ci siano ALTRE proiezioni per quella fascia oraria, se no, elimino le maschere di quella fascia oraria
57            if ((select count(*)
58                from Proiezione P join Film F on P.Film = F.Nome
59                where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente)
60                and fasciaOraria = (case
61                    when P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) then fasciaOrari
62                    when '16:00:00' <= P.Ora and P.Ora <= '20:00:00' then pom
63                    when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
64                end)) = 0) then
65
66            #se non ho proiezioni per quella fasciaOraria, non mi serovno maschere
67            delete
68            from TurnoDelDipendente
69            where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
70                (select D.CodiceFiscale
71                 from Dipendente D
72                 where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
73
74        end if;
75
76    else
77        #se e maggiore di 0, non lo elimino, il dipendente ha un turno associato ad un altra proiezione
78
79
80        #verifico presenza di ALTRE proiezioni del turno di sera gestite dal proiezionista che si sta aggiornando
81        if ((select count(*)
82            from Proiezione P join Film F on P.Film = F.Nome
83            where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
84            ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
85            P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
86
87            delete
88            from TurnoDelDipendente
89            where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = ser and GiornoTurno = OLD.Data;
90

```

```

91      #analogaente a sopra
92      #elimino le 2 maschere serali se oltre la proiezione da eliminare non ne ho altre nella stessa fascia oraria
93      if ((select count(*)
94          from Proiezione P join Film F on P.Film = F.Nome
95          where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente) and
96              ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
97                  P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) ) = 0) then
98          delete
99          from TurnoDelDipendente
100         where GiornoTurno = OLD.Data and FasciaOrariaTurno = ser and CodiceFiscaleDipendente in
101             (select D.CodiceFiscale
102                 from Dipendente D
103                 where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
104     end if;
105
106    end if;
107
108    #verifico presenza di ALTRE proiezioni del turno pomeridiano gestite dal proiezionista che si sta aggiornando
109    if ((select count(*)
110        from Proiezione P join Film F on P.Film = F.Nome
111        where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
112            ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
113                P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) ) = 0) then
114        delete
115        from TurnoDelDipendente
116        where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = pom and GiornoTurno = OLD.Data;
117
118    #analogaente a sopra
119    #elimino le 2 maschere pomeridiane se oltre la proiezione da eliminare non ne ho altre nella stessa fascia oraria
120    if ((select count(*)
121        from Proiezione P join Film F on P.Film = F.Nome
122        where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente) and
123            ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
124                P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) ) = 0) then
125        delete
126        from TurnoDelDipendente
127        where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
128            (select D.CodiceFiscale
129                 from Dipendente D
130                 where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
131    end if;
132
133    end if;
134
135  end if;

136
137  end if;
138
139  #ora effettuo gli stessi controlli effettuati nel trigger di before insert:
140
141  select F.Durata
142  from Film F
143  where F.Nome = NEW.Film
144  into durata_p;
145
146  #identifico la faccia oraria in base all'ora della proiezione
147  if ('16:00:00' <= NEW.Ora and NEW.Ora <= '20:00:00') then
148      set fasciaOraria = pom;
149  end if;
150  if ('20:00:00' <= NEW.Ora and NEW.Ora <= '24:00:00') then
151      set fasciaOraria = ser;
152  end if;
153  #se ho una proiezione che si sovrappone tra i 2 turni
154  if (NEW.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(NEW.Ora, SEC_TO_TIME(durata_p*60))) then
155      set entrambi = 1;
156  end if;
157  if ('24:00:00' < NEW.Ora or NEW.Ora < '16:00:00') then
158      signal sqlstate '45009';
159      set message_text = 'Ora della proiezione non valida, deve essere compresa tra le 16 e le 24';
160  end if;
161
162  if NEW.Dipendente is not null then
163
164      if not entrambi then
165          #prima verifico che sia presente nella tabella Turno la tupla:
166          if ((select FasciaOraria
167              from Turno T
168              where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = fasciaOraria)
169                  is null) then
170                      #se non è presente, la creo
171                      insert into Turno values (fasciaOraria, NEW.Data, NEW.Cinema);
172      end if;
173      #se è presente devo verificare che il dipendente (se è stato selezionato per la proiezione
174      #allora è sicuramente libero per come funzionano le procedure sviluppate) sia associato a quel turno
175      #e se non lo fosse lo associo
176      if ((select CodiceFiscaleDipendente
177          from TurnoDelDipendente TD
178          where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and TD.FasciaOrariaTurno = fasciaOraria)
179          is null)then
180          insert into TurnoDelDipendente

```

```

181     values (fasciaOraria, NEW.Data, NEW.Dipendente);
182 end if;
183
184 #faccio in modo che ci siano almeno 2 maschere differenti per la fascia oraria in cui ho la proiezione
185 select count(distinct TD.CodiceFiscaleDipendente)
186 from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
187 where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = fasciaOraria and D.Tipo = 'Maschera'
188 into maschereDisponibili;
189 #per tante volte quante sono le maschere mancanti per arrivare a 2
190 #aggiungo al turno una maschera a caso disponibile.
191 #definisco una maschera disponibile per un cinema se:
192 #1)lavora per quel cinema
193 #2)non ha un turno associato alla fascia oraria della proiezione
194 while maschereDisponibili < 2 do
195
196     select CodiceFiscale
197     from Dipendente
198     where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
199         select D.CodiceFiscale
200         from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale = TD.CodiceFiscaleDipendente
201             and D.Cinema = NEW.Cinema
202             where TD.FasciaOrariaTurno = fasciaOraria and TD.GiornoTurno = NEW.Data and D.Tipo = 'Maschera')
203     order by RAND() limit 1 into mascheraCasuale;
204
205     insert into TurnoDelDipendente
206     values (fasciaOraria, NEW.Data, mascheraCasuale);
207
208     set maschereDisponibili = maschereDisponibili +1;
209
210 end while;
211 #se il proiezionista deve gestire una proiezione tale per cui gli occupa entrambi i turni della giornata:
212 else
213     #devo verificare che sia presente sia il turno pomeridiano che quello serale
214     if ((select FasciaOraria
215         from Turno T
216         where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = pom)
217         is null) then
218         insert into Turno values (pom, NEW.Data, NEW.Cinema);
219     end if;
220     if ((select FasciaOraria
221         from Turno T
222         where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = ser)
223         is null) then
224         insert into Turno values (ser, NEW.Data, NEW.Cinema);
225     end if;
226
227     end if;
228     #resta da verificare che il dipendente scelto per la gestione della proiezione
229     #sia associato ai turni sia serale che pomeridiano
230     if ((select CodiceFiscaleDipendente
231         from TurnoDelDipendente TD
232         where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and TD.FasciaOrariaTurno = pom)
233         is null )then
234         insert into TurnoDelDipendente
235             values (pom, NEW.Data, NEW.Dipendente);
236     end if;
237     if ((select CodiceFiscaleDipendente
238         from TurnoDelDipendente TD
239         where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and TD.FasciaOrariaTurno = ser)
240         is null )then
241         insert into TurnoDelDipendente
242             values (ser, NEW.Data, NEW.Dipendente);
243     end if;
244     #stessa cosa per le maschere, almeno 2 per il turno di sera e quello pomeridiano
245     select count(distinct TD.CodiceFiscaleDipendente)
246     from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
247     where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = ser and D.Tipo = 'Maschera'
248     into maschereDisponibiliSerali;
249
250     select count(distinct TD.CodiceFiscaleDipendente)
251     from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
252     where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = pom and D.Tipo = 'Maschera'
253     into maschereDisponibiliPomeridiane;
254
255     while maschereDisponibiliSerali < 2 do
256
257         select CodiceFiscale
258         from Dipendente
259         where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
260             select D.CodiceFiscale
261             from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale = TD.CodiceFiscaleDipendente
262                 and D.Cinema = NEW.Cinema
263                 where TD.FasciaOrariaTurno = ser and TD.GiornoTurno = NEW.Data and D.Tipo = 'Maschera')
264         order by RAND() limit 1 into mascheraCasuale;
265
266         insert into TurnoDelDipendente
267         values (ser, NEW.Data, mascheraCasuale);
268
269         set maschereDisponibiliSerali = maschereDisponibiliSerali +1;
270
271     end while;

```

```

270      while maschereDisponibiliPomeridiane < 2 do
271      select CodiceFiscale
272      from Dipendente
273      where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
274          select D.CodiceFiscale
275          from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale = TD.CodiceFiscaleDipendente
276          and D.Cinema = NEW.Cinema
277          where TD.FasciaOrariaTurno = pom and TD.GiornoTurno = NEW.Data and D.Tipo = 'Maschera')
278      order by RAND() limit 1 into mascheraCasuale;
279
280      insert into TurnoDelDipendente
281      values (pom, NEW.Data, mascheraCasuale);
282
283      set maschereDisponibiliPomeridiane = maschereDisponibiliPomeridiane +1;
284
285      end while;
286  end if;
287 end if;
288 #se e stata inserita una proiezione senza specificare il dipendente
289 #non faccio nulla
290
291
292 END

```

Trigger di tipo BEFORE DELETE della tabella ‘Proiezione’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Proiezione_BEFORE_DELETE` BEFORE DELETE ON `Proiezione` FOR EACH ROW
2 BEGIN
3 #prima di cancellare una proiezione devo eliminare l'associazione tra il proiezionista
4 #(qualora esistesse) e la proiezione stessa, e controllare se nella data e fascia oraria
5 #non ci sono altre proiezioni devo eliminare anche le maschere che dovevano timbrare i biglietti all ingresso
6
7 declare pom varchar(10) default '16-20';
8 declare ser varchar(10) default '20-24';
9 declare fasciaOraria varchar(10);
10 declare entrambi int default 0;
11 declare durata_p int;
12
13 if (OLD.Dipendente is not null) then
14
15     select F.Durata
16     from Film F
17     where F.Nome = OLD.Film
18     into durata_p;
19
20     if ('16:00:00' <= OLD.Ora and OLD.Ora <= '20:00:00') then
21         set fasciaOraria = pom;
22     end if;
23     if ('20:00:00' <= OLD.Ora and OLD.Ora <= '24:00:00') then
24         set fasciaOraria = ser;
25     end if;
26     #se ho una proiezione che si sovrappone tra i 2 turni
27     if (OLD.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(OLD.Ora, SEC_TO_TIME(durata_p*60))) then
28         set entrambi = 1;
29     end if;
30
31     # se il proiezionista non ha associate altre proiezioni per quel giorno a quella fascia oraria
32     # gli cancello il turno, o entrambi i turni, qualora la proiezione che verra occupata gli occupi
33     # entrambi i turni
34     if not entrambi then
35
36         if ((select count(*)
37             from Proiezione P join Film F on P.Film = F.Nome
38             where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
39                 # calcolo la faccia oraria per quel film
40                 fasciaOraria = (case
41                     # se in quel giorno il proiezionista ha un altro film che gli occupa
42                     # entrambi i turni, non voglio cancellarglieli
43                     when P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) then fasciaOrar
44                     when '16:00:00' <= P.Ora and P.Ora <= '20:00:00' then pom
45                     when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser

```

```

46      end)) = 0) then
47      #se è 0 elimino il turno del dipendente a quella fascia oraria
48      delete
49      from TurnoDelDipendente
50      where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = fasciaOraria and GiornoTurno = OLD.Data;
51
52      #verifico che ci siano ALTRE proiezioni per quella fascia oraria, se no, elimino le maschere di quella fascia oraria
53      if ((select count(*)
54          from Proiezione P join Film F on P.Film = F.Nome
55          where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente)
56          and fasciaOraria = (case
57              when P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) then fasciaOraria
58              when '16:00:00' <= P.Ora and P.Ora <= '20:00:00' then pom
59              when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
60          end)) = 0) then
61          #se non ho proiezioni per quella fasciaOraria, non mi servono maschere
62          delete
63          from TurnoDelDipendente
64          where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
65              (select D.CodiceFiscale
66                  from Dipendente D
67                  where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
68
69      end if;
70
71  end if;
72  #se è maggiore di 0, non lo elimino, il dipendente ha un turno associato ad un'altra proiezione
73
74 else
75  #se la proiezione che si vuole cancellare occupava al dipendente entrambi i turni,
76  #verifico sia per il turno di sera che di pomeriggio che il dipendente abbia schedule
77  #altre proiezioni, in caso contrario, gli cancello il turno (uno dei due o entrambi)
78  if ((select count(*)
79      from Proiezione P join Film F on P.Film = F.Nome
80      where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
81          ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
82          P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
83      delete
84      from TurnoDelDipendente
85      where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = ser and GiornoTurno = OLD.Data;
86
87  #analogalemente a sopra
88  #elimino le 2 maschere serali se oltre la proiezione da eliminare non ne ho altre nella stessa fascia oraria
89  if ((select count(*)
90      from Proiezione P join Film F on P.Film = F.Nome
91      where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente) and
92          ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
93          P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
94      delete
95      from TurnoDelDipendente
96      where GiornoTurno = OLD.Data and FasciaOrariaTurno = ser and CodiceFiscaleDipendente in
97          (select D.CodiceFiscale
98              from Dipendente D
99              where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
100
101 end if;
102
103 if ((select count(*)
104      from Proiezione P join Film F on P.Film = F.Nome
105      where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
106          ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
107          P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
108      delete
109      from TurnoDelDipendente
110      where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = pom and GiornoTurno = OLD.Data;
111
112 if ((select count(*)
113      from Proiezione P join Film F on P.Film = F.Nome
114      where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente) and
115          ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
116          P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
117      delete
118      from TurnoDelDipendente
119      where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
120          (select D.CodiceFiscale
121              from Dipendente D
122              where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
123
124 end if;
125
126 end if;
127
128 end if;
129
130 END
131 END

```

Trigger di tipo AFTER INSERT della tabella ‘Biglietto’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Biglietto_AFTER_INSERT` AFTER INSERT ON `Biglietto` FOR EACH ROW
2 BEGIN
3     #dopo che è stato inserito un biglietto
4     #aggiorno l'incasso totale della relativa proiezione
5
6     declare costo int;
7     declare incasso int;
8
9     select P.Costo, P.IncassoTotale
10    from Proiezione P
11   where P.Cinema = NEW.CinemaDellaProiezione and P.Sala = NEW.SalaDellaProiezione and P.Ora = NEW.OraProiezione
12      and P.Film = NEW.Film and P.Data = NEW.DataProiezione
13   into costo, incasso;
14
15    if incasso is null then
16        set incasso = 0;
17    end if;
18
19    set incasso = incasso + costo;
20
21    update Proiezione
22    set IncassoTotale = incasso
23   where Cinema = NEW.CinemaDellaProiezione and Sala = NEW.SalaDellaProiezione and Ora = NEW.OraProiezione
24      and Film = NEW.Film and Data = NEW.DataProiezione;
25
26 END

```

Trigger di tipo AFTER UPDATE della tabella ‘Biglietto’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Biglietto_AFTER_UPDATE` AFTER UPDATE ON `Biglietto` FOR EACH ROW
2 BEGIN
3     #se e' stato aggiornato il suo tipo in 'Annullato' devo decrementare il costo dall'incasso totale
4     declare costo int;
5     declare incasso int;
6
7     if(NEW.Tipo = 'Annullato') then
8         select P.Costo, P.IncassoTotale
9         from Proiezione P
10        where P.Cinema = NEW.CinemaDellaProiezione and P.Sala = NEW.SalaDellaProiezione and P.Ora = NEW.OraProiezione
11          and P.Film = NEW.Film and P.Data = NEW.DataProiezione
12        into costo, incasso;
13
14        set incasso = incasso - costo;
15
16        update Proiezione
17        set IncassoTotale = incasso
18       where Cinema = NEW.CinemaDellaProiezione and Sala = NEW.SalaDellaProiezione and Ora = NEW.OraProiezione
19          and Film = NEW.Film and Data = NEW.DataProiezione;
20    end if;
21
22 END

```

Trigger di tipo BEFORE UPDATE della tabella ‘CartaDiCredito’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`CartaDiCredito_BEFORE_UPDATE` BEFORE UPDATE ON `CartaDiCredito` FOR EACH ROW
2 BEGIN
3     #verifico che la data sia un numero compreso tra 1 e 12,
4     #che l'anno sia di 4 cifre
5     #il CVV di 3 cifre
6     #e il numero della carta di 16 cifre
7
8     if (LENGTH(NEW.Numero) != 16) then
9         signal sqlstate '45021'
10        set message_text = 'Numero carta non valido';
11    end if;
12
13    if ( 13 <= NEW.MeseScadenza or NEW.MeseScadenza <= 0) then
14        signal sqlstate '45019'
15        set message_text = 'Mese di scadenza non valido';
16    end if;
17
18    if ( 0 >= NEW.AnnoScadenza or NEW.AnnoScadenza >= 10000) then
19        signal sqlstate '45022'
20        set message_text = 'Anno di scadenza non valido';
21    end if;
22
23    if ( 0 >= NEW.CVV or NEW.CVV >= 1000) then
24        signal sqlstate '45020'
25        set message_text = 'CVV non valido';
26    end if;
27
28 END

```

Trigger di tipo BEFORE INSERT della tabella ‘CartaDiCredito’.

```

1 • CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`CartaDiCredito_BEFORE_INSERT` BEFORE INSERT ON `CartaDiCredito` FOR EACH ROW
2   BEGIN
3
4     #verifico che la data sia un numero compreso tra 1 e 12,
5     #che l'anno sia di 4 cifre
6     #il CVV di 3 cifre
7     #e il numero della carta di 16 cifre
8
9     if (LENGTH(NEW.Numero) != 16) then
10       signal sqlstate '45021'
11       set message_text = 'Numero carta non valido';
12     end if;
13
14     if ( 13 <= NEW.MeseScadenza or NEW.MeseScadenza <= 0) then
15       signal sqlstate '45019'
16       set message_text = 'Mese di scadenza non valido';
17     end if;
18
19     if ( 0 >= NEW.AnnoScadenza or NEW.AnnoScadenza >= 100) then
20       signal sqlstate '45022'
21       set message_text = 'Anno di scadenza non valido';
22     end if;
23
24     if ( 0 >= NEW.CVV or NEW.CVV >= 1000) then
25       signal sqlstate '45020'
26       set message_text = 'CVV non valido';
27     end if;
28
29   END

```

## Eventi

Non sono stati implementati eventi.

## 5 Viste

Non sono state implementate visite.

## Stored Procedures e transazioni

N.B.: alcune delle stored procedures non contengono (erroneamente) l'operazione 'start transaction' e 'commit' dopo che è stato modificato il livello di isolamento delle transazioni. Nel codice sql vero e proprio tali operazioni sono presenti.

### 5 Procedura 'acquistaBiglietto'.

```

1 • CREATE PROCEDURE `acquistaBiglietto`(in cinema int, in sala int, in data_p DATE, in ora_p TIME, in film varchar(128), in numero_posto int,
2   in fila_posto varchar(), in numero varchar(16), in nomeIntest varchar(45), in cognomeIntest varchar(45), in meseScad int, in annoScad
3   int, in cvv int, out codice_di_p varchar(45))
4   BEGIN
5     declare codice varchar(45);
6     declare exit handler for sqlexception
7     begin
8       rollback;
9       resignal;
10    end;
11
12    declare exit handler for 1062
13    begin
14      rollback;
15      signal sqlstate '45014';
16      set message_text = 'Sei stato battuto sul tempo. Qualcuno ha appena occupato il tuo stesso posto.';
17    end;
18
19    set transaction isolation level serializable;
20
21    start transaction;
22
23    #prime di procedere controllo che la carta di credito sia presente nel db,
24    #se così non fosse devo inserirla prima di continuare la procedura di acquisto del biglietto
25
26    if (not cartaEsistente('numero')) then
27      insert into CartaDiCredito values(numero, nomeIntest, cognomeIntest, meseScad, annoScad, cvv);
28    end if;
29
30    set codice = concat(data_p);
31    set codice = concat(codice, cinema);
32    set codice = concat(codice, ora_p);
33    set codice = concat(codice, sala);
34    set codice = concat(codice, CURTIME());
35    set codice = concat(codice, fila_posto);
36    set codice = concat(codice, numero_posto);
37
38    #verifico che la proiezione inserita esista
39    if( select P.Cinema
40      from Proiezione P
41      where P.Data = data_p and P.Ora = ora_p and P.Film = film and P.Cinema = cinema and P.Sala = sala
42
43      and
44      #e che i posti inseriti siano effettivamente disponibili
45      (fila_posto, numero_posto) in (select P.Fila, P.Numero
46        from Posto P
47        where P.Cinema = cinema and P.Sala = sala and ((select B.CodiceDiPrenotazione
48          from Biglietto B
49          where B.CinemaDellaProiezione = cinema and B.SalaDellaProiezione = sala and
50            B.OraProiezione = ora_p and B.DataProiezione = data_p and B.Film = film and
51            B.NumeroPosto = P.Numero and B.FilaPosto = P.Fila) is null
52
53      or
54      (select B.CodiceDiPrenotazione
55        from Biglietto B
56        where B.CinemaDellaProiezione = cinema and B.SalaDellaProiezione = sala and
57          B.OraProiezione = ora_p and B.DataProiezione = data_p and B.Film =
58          film and
59          'Annullato') is not null)) is not null then
60
61      insert into Biglietto values (codice, 'AcquistatoNonConfermato', numero_posto, fila_posto, data_p, film, sala, ora_p, cinema,
62      numero);
63      select codice into codice_di_p;
64    end if;
65
66    commit;
67
68 END

```

Procedura ‘aggiornaProiezionistaDellaProiezione’. Usata quando un amministratore vuole cambiare il proiezionista di una certa proiezione.

```

1 • CREATE PROCEDURE `aggiornaProiezionistaDellaProiezione`(in proiezionista varchar(45), in data_p DATE, in ora_p TIME, in film varchar(128),
2   BEGIN
3     #prima dell'aggiunta del proiezionista alla proiezione,
4     #assumo che il gestore abbia consultato i proiezionisti disponibili
5     #con la procedura 'proiezionistiDisponibili'
6     declare durata_p int;
7     declare exit handler for sqlexception
8       begin
9         rollback;
10        resignal;
11      end;
12
13      set transaction isolation level serializable;
14      start transaction;
15      select Durata from Film where Nome = film into durata_p;
16      #verifico che il dipendente scelto sia un proiezionista
17      if (select proiezionista where proiezionista in (
18        select CodiceFiscale
19        from Dipendente
20        where Tipo = 'Proiezionista' and Cinema = cinema) is null) then
21          signal sqlstate '45006'
22          set message_text = 'Il dipendente scelto non e nel cinema dato in input, o non e un proiezionista';
23        end if;
24
25      #verifico che il proiezionista selezionato sia effettivamente disponibile
26      if (select proiezionista where proiezionista in (
27        select distinct D.CodiceFiscale
28        from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.Film = F.Nome
29        where P.Data = data_p and D.Tipo = 'Proiezionista'
30        and (
31          (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(F.Ora)
32          or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
33          or (P.Ora < ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
34          or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))) ) is not null ) then
35          signal sqlstate '45005'
36          set message_text = 'Proiezionista occupato, usa proiezionistiDisponibili per vedere quale proiezionista scegliere';
37        end if;
38      #aggiungo il proiezionista alla proiezione
39      update Proiezione P
40      set Dipendente = proiezionista
41      where P.Cinema = cinema and P.Data = data_p and P.Ora = ora and P.Film = film and P.Sala = sala;
42      commit;
43    END

```

Procedura ‘aggiungiAttoriAlFilm’.

```

1 • CREATE PROCEDURE `aggiungiAttoriAlFilm`(in nome_attore varchar(30), in cognome_attore varchar(20), in nome_film varchar(128))
2   BEGIN
3
4     set transaction isolation level serializable;
5
6     insert into AttoriNelFilm values (nome_attore, cognome_attore, nome_film);
7
8   END

```

### Procedura 'aggiungiProiezione'.

```

1 • CREATE PROCEDURE `aggiungiProiezione`(in data_p DATE, in ora_p TIME, in film varchar(128), in sala_p int, in cinema int, in proiezionista
2   varchar(45), in costo_p int)
3   BEGIN
4
5     declare durata_p int;
6
7     declare exit handler for sqlexception
8     begin
9       rollback;
10      resignal;
11    end;
12
13    #non voglio permettere nessun tipo di anomalia perchè l'aggiunta della proiezione
14    #coinvolge la ricerca di sala e proiezionista "liberi" e innesca dei trigger per
15    #la costruzione dei turni dei dipendenti, quindi una anomalia potrebbe
16    #provocare il comportamento inaspettato di tale procedura
17
18    set transaction isolation level serializable;
19
20    start transaction;
21
22    #estratto la durata del film che si vuole proiettare
23    select Durata from Film where Nome = film into durata_p;
24    #verifico che la sala in input si effettivamente disponibile
25    if (select sala_p where sala_p not in (
26      select distinct S.NumeroDiSala
27      from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
28      where S.Cinema = cinema and P.Data = data_p
29      and (
30        (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(
31          F.Durata*60)))
32        or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
33        or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
34        or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))) ))is null) then
35      signal sqlstate '45004';
36      set message_text = 'Sala occupata, usa saleDisponibili per vedere quale sala scegliere';
37    end if;
38
39    #verifico che il dipendente scelto sia un proiezionista
40    if (select proiezionista where proiezionista in (
41      select CodiceFiscale
42      from Dipendente
43      where Tipo = 'Proiezionista' and Cinema = cinema) is null) then
44
45      signal sqlstate '45006';
46      set message_text = 'Il dipendente scelto non è nel cinema dato in input, o non è un proiezionista';
47    end if;
48
49    #verifico che il proiezionista selezionato sia effettivamente disponibile
50    if (select proiezionista where proiezionista in (
51      select distinct CodiceFiscale
52      from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.Film = F.Nome
53      where P.Data = data_p and D.Tipo = 'Proiezionista'
54      and (
55        (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(
56          F.Durata*60)))
57        or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
58        or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
59        or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))) ))is not null)
60    then
61      signal sqlstate '45005';
62      set message_text = 'Proiezionista occupato, usa proiezionistiDisponibili per vedere quale proiezionista scegliere';
63    end if;
64
65    insert into Proiezione (Data, Ora, Film, Sala, Dipendente, Cinema, Costo)
66    values (data_p, ora_p, film, sala_p, proiezionista, cinema, costo_p);
67
68    commit;
69  END

```

Procedura ‘aggiornaProiezioneSenzaProiezionista’. Usata se si volesse aggiungere una proiezione senza ancora aver deciso il proiezionista a doverla gestire.

```

1 • CREATE PROCEDURE `aggiungiProiezioneSenzaProiezionista`(in data_p DATE, in ora_p TIME, in film varchar(128), in cinema int, in sala int,
2   in costo_p int)
3   BEGIN
4
5     declare durata_p int;
6     declare exit handler for sqlexception
7     begin
8       rollback;
9       resignal;
10    end;
11
12    #non voglio permettere nessun tipo di anomalia perchè l'aggiunta della proiezione
13    #coinvolge la ricerca di sale "libere"
14    #quindi una anomalia potrebbe
15    #provocare il comportamento inaspettato di tale procedura
16
17    set transaction isolation level serializable;
18
19    start transaction;
20    select Durata from Film where Nome = film into durata_p;
21    #verifico che la sala in input si effettivamente disponibile
22    if (select sala where sala not in (
23      select distinct S.NumeroDiSala
24      from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
25      where S.Cinema = cinema and P.Data = data_p
26      and (
27        (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(
28          F.Durata*60)))
29        or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
30        or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
31        or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))) ))is null) then
32      signal sqlstate '45004';
33      set message_text = 'Sala occupata, usa saleDisponibili per vedere quale sala scegliere';
34    end if;
35
36    insert into Proiezione(Data, Ora, Film, Sala, Cinema, Costo)
37      values (data_p, ora_p, film, sala, cinema, costo_p);
38    commit;
39  END

```

Procedura ‘annullaBiglietto’.

```

1 • CREATE PROCEDURE `annullaBiglietto`(in codice varchar(45))
2   BEGIN
3     declare ora_p TIME;
4     declare data_p DATE;
5
6     declare exit handler for sqlexception
7     begin
8       rollback;
9       resignal;
10    end;
11
12    #voglio evitare tutte le anomalie, potrei incorrere in lettura inconsistente ed aggiornamento fantasma
13    # (ad esempio se procedura chiamata contemporaneamente ad un report, per contare la tipologia dei biglietti in un mese)
14    #se non mettessi il massimo livello di isolamento
15    set transaction isolation level serializable;
16    start transaction;
17
18    #verifico l'esistenza del biglietto da annullare
19    if (select CodiceDiPrenotazione
20      from Biglietto
21      where CodiceDiPrenotazione = codice) is null then
22      signal sqlstate '45011';
23      set message_text = 'CodiceDiPrenotazione NON trovato';
24    end if;
25
26    #verifico che il biglietto non sia di tipo 'Confermato', sono questi posso annullare
27    if (select CodiceDiPrenotazione
28      from Biglietto
29      where CodiceDiPrenotazione = codice and Tipo = 'Confermato') is not null then
30      signal sqlstate '45012';
31      set message_text = 'Puoi annullare solo biglietto di tipo Confermato';
32    end if;
33    select OraProiezione, DataProiezione from Biglietto where CodiceDiPrenotazione = codice into ora_p, data_p;
34    #verifico che non mi trovo nella mezz'ora prima di inizio del film
35    if ((CURDATE() = data_p and ADDTIME(ora_p, -CURTIME()) < '00:30:00') or
36      CURDATE() > data_p) then
37      signal sqlstate '45013';
38      set message_text = 'NON puoi annullare il biglietto prima di mezz'ora dall inizio del film o dopo la scadenza dello stesso';
39    end if;
40    update Biglietto set Tipo = 'Annullato' where CodiceDiPrenotazione = codice;
41    commit;
42  END

```

Procedura ‘calendarioTurni’. Usata per stampare (a schermo) il calendario dei turni dei dipendenti di un cinema. Verranno spampati I turni dal giorno in cui si invoca funzione fino alla domenica più vicina. Si assume quindi che il calendario dei turni venga richiesto ogni domenica in modo da avere I turni dei dipendenti per tutta la settimana lavorativa.

```

1 • CREATE PROCEDURE `calendarioTurni`(in cinema int)
2   BEGIN
3     declare data_corrente DATE default CURDATE();
4
5     #voglio evitare le letture sporche e inconsistenti che potrebbero derivare
6     #dalla chiamata di tale procedura contemporaneamente alla modifica di un turno di un dipendente
7
8     set transaction isolation level repeatable read;
9
10    #stampa i turni a partire dalla data corrente fino la domenica piu' vicina
11    week_loop: loop
12
13      select TD.FasciaOrariaTurno as FasciaOraria, TD.GiornoTurno as Giorno, TD.CodiceFiscaleDipendente as Dipendente
14      from TurnoDelDipendente TD join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
15      where D.Cinema = cinema and TD.GiornoTurno = data_corrente;
16
17      #se sono arrivato a domenica, mi fermo
18      if (DAYNAME(data_corrente) like 'Sun%') then
19        leave week_loop;
20      end if;
21
22      #altrimenti incremento il giorno
23      set data_corrente = DATE_ADD(data_corrente, interval 1 day);
24
25    end loop week_loop;
26
27  END

```

## 5 Procedura ‘confermaBiglietto’.

```

1 • CREATE PROCEDURE `confermaBiglietto`(in codice varchar(45))
2   BEGIN
3
4     declare exit handler for sqlexception
5       begin
6         rollback;
7         resignal;
8       end;
9
10    #scelta del livello di isolamento come per la procedura 'annullaBiglietto'
11
12    set transaction isolation level serializable;
13
14    start transaction;
15
16    #verifico l'esistenza del biglietto da annullare
17    if (select CodiceDiPrenotazione
18        from Biglietto
19        where CodiceDiPrenotazione = codice) is null then
20      signal sqlstate '45011';
21      set message_text = 'CodiceDiPrenotazione NON trovato';
22    end if;
23
24    #verifico che il biglietto non sia di tipo 'Annullato'
25    if (select CodiceDiPrenotazione
26        from Biglietto
27        where CodiceDiPrenotazione = codice and Tipo = 'Annullato') is not null then
28      signal sqlstate '45014';
29      set message_text = 'Questo biglietto e stato annullato';
30    end if;
31    #verifico che il biglietto non sia di tipo 'Confermato', non posso confermarlo 2 volte
32    if (select CodiceDiPrenotazione
33        from Biglietto
34        where CodiceDiPrenotazione = codice and Tipo = 'Confermato') is not null then
35      signal sqlstate '45015';
36      set message_text = 'Biglietto gia confermato';
37    end if;
38
39    update Biglietto set Tipo = 'Confermato' where CodiceDiPrenotazione = codice;
40    commit;
41
42  END

```

Procedura 'popolaAttoriNelFilm'. Procedura di popolamento usata per testing.

```

1 • CREATE PROCEDURE `popolaAttoriNelFilm`()
2 BEGIN
3
4     declare i int default 0;
5     declare j int;
6     declare nomeFilm varchar(128);
7     declare nomeAtt varchar(30);
8     declare cognomeAtt varchar(20);
9     declare num_attori int;
10    declare cur cursor for select Nome from Film;
11
12    declare exit handler for sqlexception
13    begin
14        rollback;
15        resignal;
16    end;
17
18    #procedura usata per popolare il db, usata solo nel test, ed una sola volta,
19    #non mi preoccupo del livello di isolamento
20
21    #se la tabella non è vuota, è preferibile usare la procedura 'AggiungiAttoriAlFilm'
22    if ((select NomeAttore from AttoriNelFilm limit 1) is not null) then
23        signal sqlstate '45001'
24        set message_text = 'Dei record in AttoriNelFilm sono già presenti in db';
25    end if;
26
27    #se non sono presenti dei record in Attore o Film, non posso far nulla
28    if ((select Nome from Attore limit 1) is null) then
29        signal sqlstate '45002'
30        set message_text = 'Devi prima inserire degli attori';
31    end if;
32    if ((select Nome from Film limit 1) is null) then
33        signal sqlstate '45003'
34        set message_text = 'Devi prima inserire dei film';
35    end if;
36    #assegno a tutti i film in db un 'cast' di attori protagonisti
37    #che varia da 6 a 10 attori protagonisti
38
39    open cur;
40    while (i < (select count(*) from Film)) do
41
42        fetch cur into nomeFilm;
43        select floor((10-6)*RAND()+6) into num attori;
44
45        set j = 0;
46        while_att: while ( j < num_attori ) do
47
48            select Nome, Cognome from Attore order by RAND() limit 1 into nomeAtt, cognomeAtt;
49            #se è stato scelto lo stesso attore per lo stesso film ripeto
50            if ((select A.Nome
51                  from Attore A join AttoriNelFilm ANF on A.Nome = ANF.NomeAttore and A.Cognome = ANF.CognomeAttore
52                  where Nome = nomeAtt and Cognome = cognomeAtt and ANF.NomeFilm = nomeFilm) is not null) then
53                iterate while_att;
54            end if;
55
56            insert into AttoriNelFilm values (nomeAtt, cognomeAtt, nomeFilm);
57            set j = j + 1;
58
59        end while while_att;
60
61        set i = i + 1;
62
63    end while;
64    close cur;
65
66 END

```

Procedura ‘popolaBiglietti’. Procedura di popolamento usata per testing.

```

1 • CREATE PROCEDURE `popolaBiglietti`(in cinema int, in sistema_reale int, in num_proiezioni int)
2   BEGIN
3     #procedura che aggiunge dei biglietti casuali alle proiezioni di un cinema
4     #se sono presenti dei biglietti nel db, meglio chiamare acquistaBiglietto
5
6     declare i int;
7     declare j int;
8     declare data_p DATE;
9     declare ora_p TIME;
10    declare num_bigl int;
11    declare capienza_sala int;
12    declare numero_posto int;
13    declare fila_posto varchar(1);
14    declare nomeFilm varchar(128);
15    declare sala int;
16
17    declare exit handler for sqlexception
18    begin
19      rollback;
20      resignal;
21    end;
22
23    #procedura usata per popolare il db, usata solo nel test, ed una sola volta,
24    #non mi preoccupo del livello di isolamento
25
26    if (select count(*) 
27        from Biglietto where CinemaDellaProiezione = cinema) > 0 then
28      signal sqlstate '45016'
29      set message_text = 'Dei biglietti sono già presenti in db. Popola con acquistaBiglietto';
30    end if;
31
32    if sistema_reale = 1 then
33      set cinema = 1;
34    end if;
35
36    if num_proiezioni > 12 then
37      signal sqlstate '45016'
38      set message_text = 'NON puo essere num_proiezioni > 12';
39    end if;
40
41    while cinema <= (select count(*) from Cinema) do
42
43      set i = 0;

```

```

43      set i = 0;
44      #1)sceglio num_proiezioni a cui far associare i biglietti
45      while proi: while i < num_proiezioni do
46        #2)estraggo i dati della proiezione casuale nelle variabili dichiarate,
47        select P.Data, P.Ora, P.Film, P.Sala
48        from Proiezione P
49        where P.Data >= CURDATE() and P.Ora >= CURTIME() and P.Film = film and P.Cinema = cinema
50        order by RAND() limit 1 into data_p, ora_p, nomeFilm, sala;
51
52        #se ha già dei biglietti associati ritento,
53        if(select CodiceDiPrenotazione
54            from Biglietto
55            where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
56                  OraProiezione = ora_p and DataProiezione = data_p and Film = film limit 1) is not null then
57          iterate while_proi;
58        end if;
59
60        #3)associo tale proiezione ad un numero casuale di biglietti,
61        # che va dal 5% al 90% della capienza della sala
62        select S.NumeroDiPosti from Sala S where S.NumeroDiSala = sala and S.Cinema = cinema into capienza_sala;
63        select FLOOR(((RAND()*(90-5))+5)/100)*capienza_sala into num_bigl;
64
65        set j = 0;
66        while j < num_bigl do
67
68          #4)estraggo un numero e fila di posto casuale, tra quelli disponibili
69          select P.Fila, P.Numero
70          from Posto P
71          where P.Cinema = cinema and P.Sala = sala and ((select CodiceDiPrenotazione
72              from Biglietto
73              where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
74                  OraProiezione = ora_p and DataProiezione = data_p and Film = nomeFilm and
75                  NumeroPosto = P.Numero and FilaPosto = P.Fila) is null
76
77          or
78              (select CodiceDiPrenotazione
79                  from Biglietto
80                  where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
81                      OraProiezione = ora_p and DataProiezione = data_p and Film = nomeFilm
82                      NumeroPosto = P.Numero and FilaPosto = P.Fila and Tipo =
83                      'Annullato') is not null)
84          order by RAND() limit 1
85          into fila_posto, numero_posto;

```

```

84          #per completare la procedura di acquisto biglietto un cliente dovrebbe inserire i dati relativi alla
85          #sua carta di credito. Inseriti gli stessi dati relativi alla CC per tutti i biglietto inseriti in fase di test
86
87          call acquistaBiglietto(cinema, sala, data_p, ora_p, nomeFilm, numero_posto, fila_posto, '5255294813425786', 'Test',
88          'Tester', 2, 22, 884, @cdp);
89          set j = j + 1;
90
91      end while;
92
93      set i = i + 1;
94
95  end while while_proi;
96
97  if sistema_reale = 1 then
98      set cinema = cinema + 1;
99  else
100     set cinema = 100;
101  end if;
102
103 end while;
104
105 END

```

Procedura ‘popolaProiezione’. Procedura di popolamento usata per testing.

```

1 • CREATE PROCEDURE `popolaProiezione`(in cinema int, in sistema_reale int, in num_film int)
2 BEGIN
3
4     declare i int;
5     declare j int;
6     declare k int;
7     #assumo che inizialmente ci siano 4 nuovi film di cui si
8     #vogliono fissare le proiezioni
9     declare costo_p int;
10    declare numProiezGiorn int;
11    declare giornoCorrente varchar(10);
12    declare sala_casuale int;
13    declare ora_casuale TIME;
14    declare proiezionista_casuale varchar(45);
15    declare dataCorrente DATE;
16    declare durata_p int;
17    declare finito int default false;
18    declare nome_film varchar(128);
19    declare cur_film cursor for select Nome, Durata from Film order by RAND() limit num_film;
20
21    declare exit handler for sqlexception
22    begin
23        rollback;
24        resignal;
25    end;
26
27    #procedura usata per popolare il db, usata solo nel test, ed una sola volta,
28    #non mi preoccupo del livello di isolamento
29
30    #se sono già presenti delle proiezioni, mi fermo.
31    #in questo caso aggiungerne di nuove con la procedura aggiungiProiezione
32    if ((select Ora from Proiezione where Cinema = cinema limit 1) is not null) then
33        signal sqlstate '45006';
34        set message_text = 'La tabella Proiezione e già stata popolata per il cinema inserito, usare aggiungiProiezione';
35    end if;
36
37    if sistema_reale then
38        set cinema = 1;
39    end if;
40
41    while (cinema <= (select count(*) from Cinema)) do
42        select cinema;
43        set i = 0;
44
45        open cur_film;
46        while_tag: while (i < num_film) do
47
48            #prendo un film a caso
49            fetch cur_film into nome_film, durata_p;
50
51            select CURDATE() into dataCorrente;
52            #e ne aggiungo un numero a caso di proiezioni giornaliere che va da 2 a 4
53            #per 2 settimane lavorative, a partire dalla data corrente + un giorno
54            set j = 0;
55            while_loop: while (j < 12) do
56
57                select DATE_ADD(dataCorrente, interval 1 day) into dataCorrente;
58                select DAYNAME(dataCorrente) into giornoCorrente;
59                #se sono a lunedì vado avanti, senza incrementare il contatore, il lunedì assumo cinema chiuso
60                if ((select giornoCorrente where giornoCorrente like 'Mon%') is not null) then
61                    iterate while_loop;
62                end if;
63                set j = j + 1;

```

```

62      set j = j + 1;
63
64      #per il giorno in cui mi trovo aggiungo o 2 o 4 proiezioni
65      select floor((4-2)*RAND()+2) into numProiezGiorn;
66      set k = 0;
67      set sala_casuale = 1;
68      scegli_ora_sala: while k < numProiezGiorn do
69          #scelgo un costo casuale che va dai 7 ai 10 euro
70          select (FLOOR((11-7)*RAND()+7)) into costo_p;
71
72          #scelgo un ora (di inizio film) casuale che vada dalle 16 alle 22 (film può avere durata massima di 2 ore e 10),
73          intervallate di 100 minuti
74          select (
75              case FLOOR((RAND()*100)%5)
76                  when 0 then '16:00:00'
77                  when 1 then '17:40:00'
78                  when 2 then '19:20:00'
79                  when 3 then '21:00:00'
80                  when 4 then '22:40:00'
81              else 0 end)
82          into ora_casuale;
83
84          #e una sala disponibile casuale
85          select SS.NumeroDiSala
86          from Sala SS
87
88
89      where SS.Cinema = cinema and SS.NumeroDiSala not in (
90          select distinct S.NumeroDiSala
91          from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
92          where S.Cinema = cinema and P.Data = dataCorrente
93          and (
94              (ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= P.Ora and ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60))
95              <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
96              or (ora_casuale >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(
97                  durata_p*60)))
98              or (P.Ora <= ora_casuale and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_casuale, SEC_TO_TIME(
99                  durata_p*60)))
100             or (ora_casuale <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(
101                 durata_p*60)))
102         )
103         order by RAND() limit 1
104         into sala_casuale;
105
106     if (select P.Data from Proiezione P
107         where P.Data = dataCorrente and P.Film = nome_film and P.Ora = ora_casuale and P.Sala = sala_casuale and P.Cinema
108         = cinema)
109         is not null) then
110         iterate scegli_ora_sala;
111     end if;
112
113     if sala_casuale is null then
114         select 'Non ci sono sale disponibili';
115         leave while_tag;
116     end if;
117     #devo selezionare un proiezionista casuale disponibile,
118     select DD.CodiceFiscale
119     from Dipendente DD
120     where DD.Tipo = 'Proiezionista' and DD.Cinema = cinema and DD.CodiceFiscale not in(
121         select distinct D.CodiceFiscale
122         from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.
123         Film = F.Nome
124         where P.Data = dataCorrente and D.Tipo = 'Proiezionista'
125         and (
126             (ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60))
127             <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
128             or (ora_casuale >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(
129                 durata_p*60)))
130             or (P.Ora <= ora_casuale and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_casuale, SEC_TO_TIME(
131                 durata_p*60)))
132             or (ora_casuale <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(
133                 durata_p*60)))
134             or (ora_casuale <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(
135                 durata_p*60)))
136         )
137         order by RAND() limit 1
138         into proiezionista_casuale;
139     if proiezionista_casuale is null then
140         select 'Non ci sono proiezionisti disponibili';
141         leave while_tag;
142     end if;
143     select cinema, j, k;
144     insert into Proiezione(Data, Ora, Film, Sala, Dipendente, Cinema, Costo)
145     values (dataCorrente, ora_casuale, nome_film, sala_casuale, proiezionista_casuale, cinema, costo_p);
146
147     set k = k + 1;
148     end while scegli_ora_sala;
149
150     end while while_loop;
151
152     set i = i + 1;
153     end while while_tag;

```

```

139      end while while_tag;
140
141      if sistema_reale then
142          set cinema = cinema + 1;
143      else
144          set cinema = 100;
145      end if;
146      close cur_film;
147
148  end while;
149
150
151 END

```

Procedura ‘popolaSalePosti’. Procedura di popolamento usata per testing.

```

1 • CREATE PROCEDURE `popolaSalePosti`(in sistema_reale int)
2 BEGIN
3
4     declare i int default 0;
5     declare j int;
6     declare k int;
7     declare larghezzaSala int;
8     declare fila varchar(1);
9     declare cinema int;
10    declare numSale int;
11    declare numPosti int;
12    declare cur cursor for select ID from Cinema;
13
14    declare exit handler for sqlexception
15    begin
16        rollback;
17        resignal;
18    end;
19
20    #procedura usata per popolare il db, usata solo nel test, ed una sola volta,
21    #non mi preoccupo del livello di isolamento
22
23    # se ci sono già delle sale mi fermo
24    if ((select NumeroDiSala from Sala where Cinema = '1' limit 1) is not null) then
25        signal sqlstate '45001';
26        set message_text = 'Delle sale sono già presenti in db';
27    end if;
28
29    open cur;
30    while i < (select count(*) from Cinema) do
31
32        fetch cur into cinema;
33        #ora a tale cinema inserisco un numero casuale di sale, da 6 a 10.
34        select floor((10-6)*RAND()+6) into numSale;
35        set j = 1;
36        while j <= numSale do
37
38            #ogni sala ha un numero di posti casuali, da 40 a 60
39            select floor((60-40)*RAND()+40) into numPosti;
40            insert into Sala values (j, numPosti, cinema);
41
42            #procedo con la popolazione dei posti
43
44            set fila = 'A';
45            #larghezza delle sale (data dal numero di posti per fila, assumo vari da 7 a 10)
46            select floor((10-7)*RAND()+7) into larghezzaSala;
47            set k = 0;
48            while k < numPosti do
49
50                insert into Posto values (k%larghezzaSala+1, fila, j, cinema);
51                set k = k +1;
52                #se ho raggiunto la fine della fila, 'scalo dietro'. Es. dalla fila A passo alla B
53                if (not k%larghezzaSala) then
54                    set fila = CHAR(ASCII(fila) + 1);
55                end if;
56
57            end while;
58            set j = j + 1;
59
60        end while;

```

```

61      #per evitare lungo tempo di attesa, possibilita di scegliere se simulare sistema reale oppure no
62      # set i = i + 1;
63      if sistema_reale then
64          set i = i +1;
65      else
66          set i = 100;
67      end if;
68
69      end while;
70      close cur;
71
72  END

```

### Procedura ‘postiDisponibiliDellaProiezione’.

```

1 • CREATE PROCEDURE `postiDisponibiliDellaProiezione`(in data_p DATE, in ora_p TIME, in film varchar(128), in sala int, in cinema int)
2 BEGIN
3
4     #voglio evitare le letture sporche e le letture inconsistenti che potrebbero
5     #non far funzionare come dovuto tale procedura
6
7     set transaction isolation level repeatable read;
8
9     #solo se la proiezione esiste veramente
10    if ((select count(*)
11        from Proiezione P
12        where P.Data = data_p and P.Ora = ora_p and P.Film = film and P.Sala = sala and P.Cinema = cinema) > 0) then
13
14        select P.Fila, P.Numero, (case
15            when ((select CodiceDiPrenotazione
16                from Biglietto
17                where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
18                    OraProiezione = ora_p and DataProiezione = data_p and Film = film and
19                        NumeroPosto = P.Numero and FilaPosto = P.Fila) is null
20
21            or
22                (select CodiceDiPrenotazione
23                    from Biglietto
24                    where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
25                        OraProiezione = ora_p and DataProiezione = data_p and Film = film and
26                            NumeroPosto = P.Numero and FilaPosto = P.Fila and Tipo = 'Annullato') is not null
27
28            else '1'
29        end
30        ) as Occupato
31
32        from Posto P
33        where P.Cinema = cinema and P.Sala = sala
34        order by P.Fila asc;
35    end if;
36
37  END

```

### Procedura ‘proiezioniDisponibili’.

```

1 • CREATE PROCEDURE `proiezioniDisponibili`(in film varchar(128), in cinema int)
2 BEGIN
3
4     # mostro tutte le proiezioni a partire dalla data corrente in poi,
5     # con ora di inizio maggiore o uguale all ora attuale,
6
7     # voglio evitare letture sporche ed inconsistenti
8
9     set transaction isolation level repeatable read;
10
11    select P.Data, P.Ora, P.Sala
12    from Proiezione P
13    where P.Data >= CURDATE() and P.Ora >= CURTIME() and P.Film = film and P.Cinema = cinema;
14
15
16  END

```

Procedura ‘proiezionistiDisponibili’. Procedura usata per verificare la disponibilità di un proiezionista prima di fargli gestire una proiezione.

```

1 • CREATE PROCEDURE `proiezionistiDisponibili`(`in cinema int, in ora_p TIME,in data_p DATE, in durata_p int)
2 ┌─┐ BEGIN
3   ┌─┐
4   ┌─┐ set transaction isolation level repeatable read;
5   ┌─┐
6   ┌─┐ #1)da tutti i proiezionisti di un determinato cinema
7   ┌─┐ select CodiceFiscale as DipendenteDisponibile
8   ┌─┐ from Dipendente
9   ┌─┐ where Tipo = 'Proiezionista' and Cinema = cinema and CodiceFiscale not in(
10  ┌─┐   #2)tolgo quelli occupati nell'ora e data in input, piu' la durata
11  ┌─┐   select distinct CodiceFiscale
12  ┌─┐   from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.Film = F.Nome
13  ┌─┐   where P.Data = data_p and D.Tipo = 'Proiezionista'
14  ┌─┐   and (
15  ┌─┐     (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(
16  ┌─┐       F.Durata*60)))
17  ┌─┐     or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
18  ┌─┐     or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
19  ┌─┐     or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))) );
20 ┌─┐ END

```

Procedura ‘saleDisponibili’.

```

1 • CREATE PROCEDURE `saleDisponibili`(`in cinema int, in ora_p TIME, in data_p DATE, in durata_p int)
2 ┌─┐ BEGIN
3   ┌─┐
4   ┌─┐ set transaction isolation level repeatable read;
5   ┌─┐
6   ┌─┐ #1)da tutte le sale del cinema in input
7   ┌─┐ select NumeroDiSala as SaleDisponibili
8   ┌─┐ from Sala
9   ┌─┐ where Cinema = cinema and NumeroDiSala not in(
10  ┌─┐   #2)tolgo le sale in cui ho una proiezione in quella data a quell'ora
11  ┌─┐   select distinct S.NumeroDiSala
12  ┌─┐   from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
13  ┌─┐   where S.Cinema = cinema and P.Data = data_p
14  ┌─┐   and (
15  ┌─┐     (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(
16  ┌─┐       F.Durata*60)))
17  ┌─┐     or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
18  ┌─┐     or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))
19  ┌─┐     or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)))) );
20 ┌─┐ END

```

5

10

15

Procedura ‘simulaPostProiezione’. Procedura di che simula la conferma e l’annullamento dei biglietti per una certa proiezione.

```

1  CREATE PROCEDURE `simulaPostProiezione`(in cinema int, in sala int, in film varchar(128), in data_p DATE, in ora_p TIME)
2  BEGIN
3      #procedura usata per test,
4      #per simulare l'acquisto e conferma e annullamento di biglietti di una proiezione
5      declare rand int;
6      declare codice varchar(45);
7      declare num_bigl int;
8      declare i int default 0;
9      declare cur cursor for select CodiceDiPrenotazione
10     from Biglietto
11    where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and Film = film
12        and DataProiezione = data_p and OraProiezione = ora_p;
13    declare exit handler for sqlexception
14    begin
15        rollback;
16        resignal;
17    end;
18    select count(CodiceDiPrenotazione)
19    from Biglietto
20   where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and Film = film
21       and DataProiezione = data_p and OraProiezione = ora_p into num_bigl;
22   #verifico che ci siano dei biglietti associati alla proiezione
23   if num_bigl = 0 then
24       signal sqlstate '45017';
25       set message_text = 'Nessun biglietto presente per la prenotazione in input';
26   end if;
27
28   open cur;
29   while (i < num_bigl) do
30       fetch cur into codice;
31       set rand = FLOOR(RAND()*10];
32       if( 0 <= rand and rand <= 5 ) then
33           call annullaBiglietto(codice);
34       end if;
35       if(6 <= rand and rand <= 90) then
36           call confermaBiglietto(codice);
37       end if;
38       set i = i + 1;
39
40   end while;
41   close cur;
42
43 END

```

## Appendice: Implementazione

### Codice SQL per instanziare il database

```
-- MySQL Workbench Forward Engineering

5   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';

10  -----
-- Schema CatenaCinema
-----
DROP SCHEMA IF EXISTS `CatenaCinema` ;

15  -----
-- Schema CatenaCinema
-----
CREATE SCHEMA IF NOT EXISTS `CatenaCinema` DEFAULT CHARACTER SET utf8 ;
USE `CatenaCinema` ;

20  -----
-- Table `CatenaCinema`.`Cinema`
-----
DROP TABLE IF EXISTS `CatenaCinema`.`Cinema` ;

25  CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Cinema` (
  `ID` INT NOT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;

30  -----
-- Table `CatenaCinema`.`Sala`
-----
DROP TABLE IF EXISTS `CatenaCinema`.`Sala` ;

35  CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Sala` (
  `NumeroDiSala` INT NOT NULL,
  `NumeroDiPosti` INT NOT NULL,
  `Cinema` INT NOT NULL,
  PRIMARY KEY (`NumeroDiSala`, `Cinema`),
  CONSTRAINT `fk_Sala_Cinema1`
    FOREIGN KEY (`Cinema`)
    REFERENCES `CatenaCinema`.`Cinema` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

40  ON UPDATE NO ACTION)
ENGINE = InnoDB;

45  CREATE INDEX `fk_Sala_Cinema1_idx` ON `CatenaCinema`.`Sala` (`Cinema` ASC);

50  -----
-- Table `CatenaCinema`.`Posto`
-----
DROP TABLE IF EXISTS `CatenaCinema`.`Posto` ;

55  CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Posto` (
  `Numero` INT NOT NULL,
  `Fila` VARCHAR(1) NOT NULL,
  `Sala` INT NOT NULL,
  `Cinema` INT NOT NULL,
  PRIMARY KEY (`Numero`, `Fila`, `Sala`, `Cinema`),
  CONSTRAINT `fk_Posto_Sala1`
    FOREIGN KEY (`Sala`, `Cinema`)
    REFERENCES `CatenaCinema`.`Sala` (`NumeroDiSala`, `Cinema`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

60  ON UPDATE NO ACTION)
ENGINE = InnoDB;

65  CREATE INDEX `fk_Posto_Sala1_idx` ON `CatenaCinema`.`Posto` (`Sala` ASC, `Cinema` ASC);
```

```

-----
5   -- Table `CatenaCinema`.`Film`
-----
10  DROP TABLE IF EXISTS `CatenaCinema`.`Film` ;

15  CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Film` (
16    `Nome` VARCHAR(128) NOT NULL,
17    `Durata` INT NOT NULL,
18    `CasaCinematografica` VARCHAR(45) NOT NULL,
19    PRIMARY KEY (`Nome`)
20  ) ENGINE = InnoDB;

25  -----
26  -- Table `CatenaCinema`.`Dipendente`
27  -----
30  DROP TABLE IF EXISTS `CatenaCinema`.`Dipendente` ;

35  CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Dipendente` (
36    `CodiceFiscale` VARCHAR(45) NOT NULL,
37    `Nome` VARCHAR(45) NOT NULL,
38    `Cognome` VARCHAR(45) NOT NULL,
39    `Tipo` VARCHAR(45) NOT NULL,
40    `Cinema` INT NOT NULL,
41    PRIMARY KEY (`CodiceFiscale`),
42    CONSTRAINT `fk_Dipendente_Cinema1`
43      FOREIGN KEY (`Cinema`)
44        REFERENCES `CatenaCinema`.`Cinema` (`ID`)
45        ON DELETE NO ACTION
46        ON UPDATE NO ACTION)
47  ENGINE = InnoDB;

48  CREATE INDEX `fk_Dipendente_Cinema1_idx` ON `CatenaCinema`.`Dipendente` (`Cinema` ASC);

50  -----
51  -- Table `CatenaCinema`.`Proiezione`
52  -----
55  DROP TABLE IF EXISTS `CatenaCinema`.`Proiezione` ;

56  CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Proiezione` (
57    `Data` DATE NOT NULL,
58    `Ora` TIME NOT NULL,
59    `IncassoTotale` INT NULL,
60    `Film` VARCHAR(128) NOT NULL,
61    `Sala` INT NOT NULL,
62    `Dipendente` VARCHAR(45) NULL,
63    `Cinema` INT NOT NULL,
64    `Costo` INT NOT NULL,
65    PRIMARY KEY (`Data`, `Film`, `Sala`, `Ora`, `Cinema`),
66    CONSTRAINT `fk_Proiezione_Film1`
67      FOREIGN KEY (`Film`)
68        REFERENCES `CatenaCinema`.`Film` (`Nome`)
69        ON DELETE NO ACTION
70        ON UPDATE NO ACTION,
71    CONSTRAINT `fk_Proiezione_Sala1`
72      FOREIGN KEY (`Sala`, `Cinema`)
73        REFERENCES `CatenaCinema`.`Sala` (`NumeroDiSala`, `Cinema`)
74        ON DELETE NO ACTION
75        ON UPDATE NO ACTION,
76    CONSTRAINT `fk_Proiezione_Dipendente1`
77      FOREIGN KEY (`Dipendente`)
78        REFERENCES `CatenaCinema`.`Dipendente` (`CodiceFiscale`)
79        ON DELETE NO ACTION
80        ON UPDATE NO ACTION)
81  ENGINE = InnoDB;

82  CREATE INDEX `fk_Proiezione_Film1_idx` ON `CatenaCinema`.`Proiezione` (`Film` ASC);
83  CREATE INDEX `fk_Proiezione_Sala1_idx` ON `CatenaCinema`.`Proiezione` (`Sala` ASC, `Cinema` ASC);
84  CREATE INDEX `fk_Proiezione_Dipendente1_idx` ON `CatenaCinema`.`Proiezione` (`Dipendente` ASC);

```

```

-----  

-- Table `CatenaCinema`.`Attore`  

-----  

5  DROP TABLE IF EXISTS `CatenaCinema`.`Attore` ;  

CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Attore` (  

  `Nome` VARCHAR(45) NOT NULL,  

  `Cognome` VARCHAR(45) NOT NULL,  

10 PRIMARY KEY (`Nome`, `Cognome`))  

ENGINE = InnoDB;  

-----  

15 -- Table `CatenaCinema`.`AttoriNelFilm`  

-----  

DROP TABLE IF EXISTS `CatenaCinema`.`AttoriNelFilm` ;  

CREATE TABLE IF NOT EXISTS `CatenaCinema`.`AttoriNelFilm` (  

20  `NomeAttore` VARCHAR(45) NOT NULL,  

  `CognomeAttore` VARCHAR(45) NOT NULL,  

  `NomeFilm` VARCHAR(128) NOT NULL,  

  PRIMARY KEY (`NomeAttore`, `CognomeAttore`, `NomeFilm`),  

  CONSTRAINT `fk_Attore_has_Film_Attore`  

25    FOREIGN KEY (`NomeAttore`, `CognomeAttore`)  

      REFERENCES `CatenaCinema`.`Attore` (`Nome`, `Cognome`)  

      ON DELETE NO ACTION  

      ON UPDATE NO ACTION,  

      CONSTRAINT `fk_Attore_has_Film_Film1`  

30    FOREIGN KEY (`NomeFilm`)  

      REFERENCES `CatenaCinema`.`Film` (`Nome`)  

      ON DELETE NO ACTION  

      ON UPDATE NO ACTION)  

ENGINE = InnoDB;  

35 CREATE INDEX `fk_Attore_has_Film_Film1_idx` ON `CatenaCinema`.`AttoriNelFilm` (`NomeFilm` ASC);  

CREATE INDEX `fk_Attore_has_Film_Attore_idx` ON `CatenaCinema`.`AttoriNelFilm` (`NomeAttore` ASC, `CognomeAttore` ASC);  

40 -----  

-- Table `CatenaCinema`.`Turno`  

-----  

DROP TABLE IF EXISTS `CatenaCinema`.`Turno` ;  

45 CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Turno` (  

  `FasciaOraria` VARCHAR(10) NOT NULL,  

  `Giorno` DATE NOT NULL,  

  `Cinema` INT NOT NULL,  

50 PRIMARY KEY (`FasciaOraria`, `Giorno`, `Cinema`),  

  CONSTRAINT `fk_Turno_Cinema1`  

    FOREIGN KEY (`Cinema`)  

      REFERENCES `CatenaCinema`.`Cinema` (`ID`)  

      ON DELETE NO ACTION  

55      ON UPDATE NO ACTION)  

ENGINE = InnoDB;  

CREATE INDEX `fk_Turno_Cinema1_idx` ON `CatenaCinema`.`Turno` (`Cinema` ASC);  

60 -----  

-- Table `CatenaCinema`.`TurnoDelDipendente`  

-----  

DROP TABLE IF EXISTS `CatenaCinema`.`TurnoDelDipendente` ;  

65 CREATE TABLE IF NOT EXISTS `CatenaCinema`.`TurnoDelDipendente` (  

  `FasciaOrariaTurno` VARCHAR(10) NOT NULL,  

  `GiornoTurno` DATE NOT NULL,  

  `CodiceFiscaleDipendente` VARCHAR(45) NOT NULL,  

70 PRIMARY KEY (`FasciaOrariaTurno`, `GiornoTurno`, `CodiceFiscaleDipendente`),  

  CONSTRAINT `fk_Turno_has_Dipendente_Turno1`  

    FOREIGN KEY (`FasciaOrariaTurno`, `GiornoTurno`)  

      REFERENCES `CatenaCinema`.`Turno` (`FasciaOraria`, `Giorno`)  

      ON DELETE NO ACTION  

75      ON UPDATE NO ACTION,

```

```

CONSTRAINT `fk_Turno_has_Dipendente_Dipendente1`  

  FOREIGN KEY (`CodiceFiscaleDipendente`)  

    REFERENCES `CatenaCinema`.`Dipendente` (`CodiceFiscale`)  

    ON DELETE NO ACTION  

    ON UPDATE NO ACTION  

5   ENGINE = InnoDB;  

CREATE INDEX `fk_Turno_has_Dipendente_Dipendente1_idx` ON `CatenaCinema`.`TurnoDelDipendente` (`CodiceFiscaleDipendente` ASC);  

10 -----
-- Table `CatenaCinema`.`CartaDiCredito`  

-----  

DROP TABLE IF EXISTS `CatenaCinema`.`CartaDiCredito` ;  

15 CREATE TABLE IF NOT EXISTS `CatenaCinema`.`CartaDiCredito` (  

  `Numero` VARCHAR(16) NOT NULL,  

  `NomeIntestatario` VARCHAR(45) NOT NULL,  

  `CognomeIntestatario` VARCHAR(45) NOT NULL,  

20  `MeseScadenza` INT UNSIGNED NOT NULL,  

  `AnnoScadenza` INT UNSIGNED NOT NULL,  

  `CVV` INT UNSIGNED NOT NULL,  

  PRIMARY KEY (`Numero`))  

ENGINE = InnoDB;  

25 -----
-- Table `CatenaCinema`.`Biglietto`  

-----  

30 DROP TABLE IF EXISTS `CatenaCinema`.`Biglietto` ;  

CREATE TABLE IF NOT EXISTS `CatenaCinema`.`Biglietto` (  

  `CodiceDiPrenotazione` VARCHAR(45) NOT NULL,  

  `Tipo` VARCHAR(45) NOT NULL,  

35  `NumeroPosto` INT NOT NULL,  

  `FilaPosto` VARCHAR(1) NOT NULL,  

  `DataProiezione` DATE NOT NULL,  

  `Film` VARCHAR(128) NOT NULL,  

  `SalaDellaProiezione` INT NOT NULL,  

40  `OraProiezione` TIME NOT NULL,  

  `CinemaDellaProiezione` INT NOT NULL,  

  `NumeroCarta` VARCHAR(16) NOT NULL,  

  PRIMARY KEY (`CodiceDiPrenotazione`, `DataProiezione`, `Film`, `OraProiezione`, `NumeroPosto`, `FilaPosto`, `SalaDellaProiezione`,  

`CinemaDellaProiezione`),  

45  CONSTRAINT `fk_Biglietto_Posto1`  

    FOREIGN KEY (`NumeroPosto`, `FilaPosto`)  

      REFERENCES `CatenaCinema`.`Posto` (`Numero`, `Fila`)  

      ON DELETE NO ACTION  

      ON UPDATE NO ACTION,  

50  CONSTRAINT `fk_Biglietto_Proiezione1`  

    FOREIGN KEY (`DataProiezione`, `Film`, `SalaDellaProiezione`, `OraProiezione`, `CinemaDellaProiezione`)  

      REFERENCES `CatenaCinema`.`Proiezione` (`Data`, `Film`, `Sala`, `Ora`, `Cinema`)  

      ON DELETE NO ACTION  

      ON UPDATE NO ACTION,  

55  CONSTRAINT `fk_Biglietto_CartaDiCredito1`  

    FOREIGN KEY (`NumeroCarta`)  

      REFERENCES `CatenaCinema`.`CartaDiCredito` (`Numero`)  

      ON DELETE NO ACTION  

      ON UPDATE NO ACTION)  

60 ENGINE = InnoDB;  

CREATE INDEX `fk_Biglietto_Posto1_idx` ON `CatenaCinema`.`Biglietto` (`NumeroPosto` ASC, `FilaPosto` ASC);  

CREATE INDEX `fk_Biglietto_Proiezione1_idx` ON `CatenaCinema`.`Biglietto` (`DataProiezione` ASC, `Film` ASC, `SalaDellaProiezione` ASC,  

65 `OraProiezione` ASC, `CinemaDellaProiezione` ASC);  

CREATE INDEX `fk_Biglietto_CartaDiCredito1_idx` ON `CatenaCinema`.`Biglietto` (`NumeroCarta` ASC);  

USE `CatenaCinema` ;  

70 -----
-- procedure popolaSalePosti
-----  

75 USE `CatenaCinema` ;

```

```

DROP procedure IF EXISTS `CatenaCinema`.`popolaSalePosti`;

DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `popolaSalePosti`(in sistema_reale int)
BEGIN
    declare i int default 0;
    declare j int;
    declare k int;
    declare larghezzaSala int;
    declare fila varchar(1);
    declare cinema int;
    declare numSale int;
    declare numPosti int;
    declare cur cursor for select ID from Cinema;

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

    #procedura usata per popolare il db, usata solo nel test, ed una sola volta,
    #non mi preoccupo del livello di isolamento
    set message_text = 'Delle sale sono già presenti in db';
    end if;

    open cur;
    while i < (select count(*) from Cinema) do
        fetch cur into cinema;
        #ora a tale cinema inserisco un numero casuale di sale, da 6 a 10.
        select floor((10-6)*RAND()+6) into numSale;
        set j = 1;
        while j <= numSale do
            #ogni sala ha un numero di posti casuali, da 40 a 60
            select floor((60-40)*RAND()+40) into numPosti;
            insert into Sala values (j, numPosti, cinema);

            #procedo con la popolazione dei posti
            set fila = 'A';
            #larghezza delle sale (data dal numero di posti per fila, assumo vari da 7 a 10)
            select floor((10-7)*RAND()+7) into larghezzaSala;
            set k = 0;
            while k < numPosti do
                insert into Posto values (k%larghezzaSala+1, fila, j, cinema);
                set k = k +1;
            #se ho raggiunto la fine della fila, 'scalo dietro'. Es. dalla fila A passo alla B
            if (not k%larghezzaSala) then
                set fila = CHAR(ASCII(fila) + 1);
            end if;

            end while;
            set j = j + 1;
        end while;

        #per evitare lungo tempo di attesa, possibilita di scegliere se simulare sistema reale oppure no
        # set i = i + 1;
        if sistema_reale then
            set i = i +1;
        else
            set i = 100;
        end if;

        end while;
    close cur;

```

```

END$$

DELIMITER ;

5 -----
-- procedure aggiungiAttoriAlFilm
-----

10 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`aggiungiAttoriAlFilm`;

15 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `aggiungiAttoriAlFilm`(in nome_attore varchar(30), in cognome_attore varchar(20), in nome_film varchar(128))
BEGIN

    set transaction isolation level serializable;

    insert into AttoriNelFilm values (nome_attore, cognome_attore, nome_film);

20 END$$

DELIMITER ;

25 -----
-- procedure popolaAttoriNelFilm
-----

30 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`popolaAttoriNelFilm`;

35 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `popolaAttoriNelFilm`()
BEGIN

    declare i int default 0;
    declare j int;
    declare nomeFilm varchar(128);
40    declare nomeAtt varchar(30);
    declare cognomeAtt varchar(20);
    declare num_attori int;
    declare cur cursor for select Nome from Film;

45    declare exit handler for sqlexception
begin
        rollback;
        resignal;
    end;
50
#procedura usata per popolare il db, usata solo nel test, ed una sola volta,
#non mi preoccupo del livello di isolamento

55    #se la tabella non è vuota, è preferibile usare la procedura 'AggiungiAttoriAlFilm'
if ((select NomeAttore from AttoriNelFilm limit 1) is not null) then
    signal sqlstate '45001'
    set message_text = 'Dei record in AttoriNelFilm sono già presenti in db';
end if;

60    #se non sono presenti dei record in Attore o Film, non posso far nulla
if ((select Nome from Attore limit 1) is null) then
    signal sqlstate '45002'
    set message_text = 'Devi prima inserire degli attori';
end if;
65    if ((select Nome from Film limit 1) is null) then
        signal sqlstate '45003'
        set message_text = 'Devi prima inserire dei film';
    end if;

70    #assegno a tutti i film in db un 'cast di attori protagonisti'
#che varia da 6 a 10 attori protagonisti

    open cur;
        while (i < (select count(*) from Film)) do
75

```

```

fetch cur into nomeFilm;
select floor((10-6)*RAND() + 6) into num_attori;
set j = 0;
while _att: while ( j < num_attori ) do
5
    select Nome, Cognome from Attore order by RAND() limit 1 into nomeAtt, cognomeAtt;
    #se e stato scelto lo stesso attore per lo stesso film ripeto
    if ((select A.Nome
10      ANF.CognomeAttore
       where Nome = nomeAtt and Cognome = cognomeAtt and ANF.NomeFilm = nomeFilm) is not null) then
        iterate while_att;
        end if;
15
    insert into AttoriNelFilm values (nomeAtt, cognomeAtt, nomeFilm);
    set j = j + 1;
    end while while_att;
20
    set i = i + 1;
end while;
close cur;

25 END$$
DELIMITER ;

30 -----
-- procedure popolaProiezioneSenzaProiezionista
-----

35 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`popolaProiezioneSenzaProiezionista`;

DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `popolaProiezioneSenzaProiezionista`(in cinema int)
40 BEGIN
    declare i int default 0;
    declare j int;
    declare k int;
45
    #assumo che inizialmente ci siano 4 nuovi film di cui si
    #vogliono fissare le proiezioni
    declare num_film int default 4;
    declare numProiezGiorn int;
    declare giornoCorrente varchar(10);
50
    declare costo_p int;
    declare sala_casuale int;
    declare ora_casuale TIME;
    declare dataCorrente DATE;
    declare finito int default false;
55
    declare nome_film varchar(128);
    declare durata_p int;
    declare cur_film cursor for select Nome, Durata from Film order by RAND() limit num_film;
    declare continue handler for not found set finito = true;
    declare exit handler for sqlexception
60 begin
        rollback;
        resignal;
        end;
65
    #procedura usata per popolare il db, usata solo nel test, ed una sola volta,
    #non mi preoccupo del livello di isolamento

    #se sono già presenti delle proiezioni, mi fermo.
70
    #in questo caso aggiungerne di nuove con la procedura 'aggiungiProiezione'
    if ((select Ora from Proiezione limit 1) is not null) then
        signal sqlstate '45006';
        set message_text = 'La tabella Proiezione e già stata popolata, usare aggiungiProiezione';
        end if;
75

```

```

open cur_film;
  while _tag: while (i < num_film) do
    #prendo un film a caso
    fetch cur_film into nome_film, durata_p;
    if finito then
      leave while_tag;
    end if;
10   select CURDATE() into dataCorrente;
    #e ne aggiungo un numero a caso di proiezioni giornaliere che va da 2 a 4
    #per 2 settimane lavorative, a partire dalla data corrente
    set j = 0;
15   while_loop: while (j < 12) do
      select DATE_ADD(dataCorrente, interval 1 day) into dataCorrente;
      select DAYNAME(dataCorrente) into giornoCorrente;
      #se sono a lunedì vado avanti, senza incrementare il contatore, il lunedì assumo cinema chiuso
20     if ((select giornoCorrente where giornoCorrente like 'Mon%') is not null) then
       iterate while_loop;
     end if;
     set j = j + 1;
25   #per il giorno in cui mi trovo aggiungo o 2 o 4 proiezioni
   select floor((4-2)*RAND()+2) into numProiezGiorn;
   set k = 0;
   scegli_ora_sala: while k < numProiezGiorn do
     #scelgo un costo casuale che va dai 7 ai 10 euro
30     select (FLOOR((11-7)*RAND()+7)) into costo_p;
     #scelgo un ora (di inizio film) casuale che vada dalle 16 alle 22 (film può avere durata massima di 2 ore e
10), intervallate di 100 minuti
     select (
35       case FLOOR((RAND())*100)%5
         when 0 then '16:00:00'
         when 1 then '17:40:00'
         when 2 then '19:20:00'
         when 3 then '21:00:00'
         when 4 then '22:40:00'
40         else 0 end)
           into ora_casuale;
     #seleziono una sala disponibile casuale
45     select NumeroDiSala as SaleDisponibili
       from Sala
       where Cinema = cinema and NumeroDiSala not in (
         select distinct S.NumeroDiSala
         from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
         where S.Cinema = cinema and P.Data = dataCorrente
50         and (
           (ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) >= P.Ora and
           ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
           or (ora_casuale >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <=
55           ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
           or (P.Ora <= ora_casuale and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >=
           ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
           or (ora_casuale <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <=
60           ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60))))
           ))
         order by RAND() limit 1
       into sala_casuale;
       if sala_casuale is null then
90         select 'Non ci sono sale disponibili';
         leave while_tag;
       end if;
65     #se e' stata scelta la stessa sala e lo stesso orario nello stesso giorno, ripeto
     if ((select Data from Proiezione
       where Data = dataCorrente and Film = nome_film and Ora = ora_casuale and Sala =
       sala_casuale and Cinema = Cinema)
       is not null) then
       iterate scegli_ora_sala;
     end if;
75

```

```

insert into Proiezione(Data, Ora, Film, Sala, Cinema, Costo)
values (dataCorrente, ora_casuale, nome_film, sala_casuale, cinema, costo_p);

5      set k = k + 1;
      end while scegli_ora_sala;

      end while while_loop;

      set i = i + 1;
10     end while while_tag;
      close cur_film;

END$$

15    DELIMITER ;

-----
-- procedure aggiungiProiezione
20    -----
USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`aggiungiProiezione`;

25    DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `aggiungiProiezione`(in data_p DATE, in ora_p TIME, in film varchar(128), in sala_p int, in cinema int, in proiezionista
varchar(45), in costo_p int)
BEGIN
30      declare durata_p int;

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

#non voglio permettere nessun tipo di anomalia perchè l'aggiunta della proiezione
40      #coinvolge la ricerca di sala e proiezionista "liberi" e innesca dei trigger per
      #la costruzione dei turni dei dipendenti, quindi una anomalia potrebbe
      #provocare il comportamento inaspettato di tale procedura

      set transaction isolation level serializable;
45      start transaction;

      #estratto la durata del film che si vuole proiettare
      select Durata from Film where Nome = film into durata_p;
50      #verifico che la sala in input si effettivamente disponibile
      if(select sala_p where sala_p not in (
          select distinct S.NumeroDiSala
          from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
          where S.Cinema = cinema and P.Data = data_p
55          and (
              (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))
              <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
              or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
              SEC_TO_TIME(durata_p*60)))
              or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p,
              SEC_TO_TIME(durata_p*60)))
              or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
              SEC_TO_TIME(durata_p*60)))) is null) then
60          signal sqlstate '45004';
65          set message_text = 'Sala occupata, usa saleDisponibili per vedere quale sala scegliere';
          end if;

      #verifico che il dipendente scelto sia un proiezionista
      if (select proiezionista where proiezionista in (
          select CodiceFiscale
          from Dipendente
          where Tipo = 'Proiezionista' and Cinema = cinema) is null) then
70          signal sqlstate '45006';
          set message_text = 'Il dipendente scelto non e nel cinema dato in input, o non e un proiezionista';
          end if;

```

```

#verifico che il proiezionista selezionato sia effettivamente disponibile
if (select proiezionista where proiezionista in (
    select distinct CodiceFiscale
5      from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.Film = F.Nome
      where P.Data = data_p and D.Tipo = 'Proiezionista'
      and (
          (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))
<= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
10     or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60)))
     or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60)))
15     or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60)))) )is not null) then
      signal sqlstate '45005'
      set message_text = 'Proiezionista occupato, usa proiezionistiDisponibili per vedere quale proiezionista scegliere';
      end if;
20
      insert into Proiezione (Data, Ora, Film, Sala, Dipendente, Cinema, Costo)
      values (data_p, ora_p, film, sala_p, proiezionista, cinema, costo_p);

      commit;
END$$
25 DELIMITER ;

-----
-- procedure saleDisponibili
30 -----
USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`saleDisponibili`;

35 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `saleDisponibili`(in cinema int, in ora_p TIME, in data_p DATE, in durata_p int)
BEGIN
40
      set transaction isolation level repeatable read;

      #1)da tutte le sale del cinema in input
      select NumeroDiSala as SaleDisponibili
      from Sala
45      where Cinema = cinema and NumeroDiSala not in (
          #2)tolgo le sale in cui ho una proiezione in quella data a quell'ora
          select distinct S.NumeroDiSala
          from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
          where S.Cinema = cinema and P.Data = data_p
50      and (
          (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))
<= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
          or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60)))
55      or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60)))
          or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60))));
END$$
50
DELIMITER ;
60 -----
-- procedure proiezionistiDisponibili
65 -----
USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`proiezionistiDisponibili`;

70 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `proiezionistiDisponibili`(in cinema int, in ora_p TIME,in data_p DATE, in durata_p int)
BEGIN
75
      set transaction isolation level repeatable read;

```

```

#1)da tutti i proiezionisti di un determinato cinema
select CodiceFiscale as DipendenteDisponibile
      from Dipendente
5   where Tipo = 'Proiezionario' and Cinema = cinema and CodiceFiscale not in(
          #2)togli quelli occupati nell'ora e data in input, piu' la durata
          select distinct CodiceFiscale
from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.Film = F.Nome
where P.Data = data_p and D.Tipo = 'Proiezionario'
10  and (
          (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))
           <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
          or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
15    SEC_TO_TIME(durata_p*60)))
          or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p,
           SEC_TO_TIME(durata_p*60)))
          or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
           SEC_TO_TIME(durata_p*60))) );
20  END$$

DELIMITER ;

-----  

25 -- procedure popolaProiezione  
-----

USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`popolaProiezione`;

30 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `popolaProiezione`(in cinema int, in sistema_reale int, in num_film int)
BEGIN

35     declare i int;
     declare j int;
     declare k int;
     #assumo che inizialmente ci siano 4 nuovi film di cui si
40     #vogliono fissare le proiezioni
     declare costo_p int;
     declare numProiezGiorn int;
     declare giornoCorrente varchar(10);
     declare sala_casuale int;
45     declare ora_casuale TIME;
     declare proiezionario_casuale varchar(45);
     declare dataCorrente DATE;
     declare durata_p int;
     declare finito int default false;
50     declare nome_film varchar(128);
     declare cur_film cursor for select Nome, Durata from Film order by RAND() limit num_film;

     declare exit handler for sqlexception
begin
55         rollback;
         resignal;
         end;
#procedura usata per popolare il db, usata solo nel test, ed una sola volta,
60 #non mi preoccupo del livello di isolamento

#se sono già presenti delle proiezioni, mi fermo.
#in questo caso aggiungerne di nuove con la procedura aggiungiProiezione
if ((select Ora from Proiezione where Cinema = cinema limit 1) is not null) then
65         signal sqlstate '45006'
         set message_text = 'La tabella Proiezione è già stata popolata per il cinema inserito, usare aggiungiProiezione';
         end if;

70     if sistema_reale then
             set cinema = 1;
         end if;

while (cinema <= (select count(*) from Cinema)) do
        select cinema;
75     set i = 0;

```

```

open cur_film;
while_tag: while (i < num_film) do
    #prendo un film a caso
    fetch cur_film into nome_film, durata_p;
    select CURDATE() into dataCorrente;
    #e ne aggiungo un numero a caso di proiezioni giornaliere che va da 2 a 4
    #per 2 settimane lavorative, a partire dalla data corrente + un giorno
    set j = 0;
    while_loop: while (j < 12) do
        select DATE_ADD(dataCorrente, interval 1 day) into dataCorrente;
        select DAYNAME(dataCorrente) into giornoCorrente;
        #se sono a lunedì' vado avanti, senza incrementare il contatore, il lunedì' assumo cinema chiuso
        if ((select giornoCorrente where giornoCorrente like 'Mon%') is not null) then
            iterate while_loop;
        end if;
        set j = j + 1;
    20
    #per il giorno in cui mi trovo aggiungo o 2 o 4 proiezioni
    select floor((4-2)*RAND())+2 into numProiezGiorn;
    set k = 0;
    25
    set sala_casuale = 1;
    scegli_ora_sala: while k < numProiezGiorn do
        #scelgo un costo casuale che va dai 7 ai 10 euro
        select (FLOOR((11-7)*RAND())+7)) into costo_p;
        #scelgo un ora (di inizio film) casuale che vada dalle 16 alle 22 (film può avere durata massima
30    di 2 ore e 10), intervallate di 100 minuti
        select (
            case FLOOR((RAND()*100)%5)
            when 0 then '16:00:00'
            when 1 then '17:40:00'
            when 2 then '19:20:00'
            when 3 then '21:00:00'
            when 4 then '22:40:00'
            else 0 end)
        into ora_casuale;
    40
        #e una sala disponibile casuale
        select SS.NumeroDiSala
        from Sala SS
        where SS.Cinema = cinema and SS.NumeroDiSala not in (
            select distinct S.NumeroDiSala
            from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film =
        45
            F.Nome
            where S.Cinema = cinema and P.Data = dataCorrente
            and (
                (ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) >= P.Ora and
                ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
                or (ora_casuale >= P.Ora and ADDTIME(P.Ora,
                SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
                or (P.Ora <= ora_casuale and ADDTIME(P.Ora,
55    SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
                or (ora_casuale <= P.Ora and ADDTIME(P.Ora,
                SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
                ))
            order by RAND() limit 1
            into sala_casuale;
    60
            #se e' stata scelta la stessa sala e lo stesso orario nello stesso giorno, ripeto
            if ((select P.Data from Proiezione P
                where P.Data = dataCorrente and P.Film = nome_film and P.Ora = ora_casuale and
65    P.Sala = sala_casuale and P.Cinema = cinema)
                is not null) then
                    iterate scegli_ora_sala;
            end if;
            if sala_casuale is null then
                select 'Non ci sono sale disponibili';
                leave while_tag;
            end if;
            #devo selezionare un proiezionista casuale disponibile,
            select DD.CodiceFiscale
    70
    75

```

```

from Dipendente DD
where DD.Tipo = 'Proiezionario' and DD.Cinema = cinema and DD.CodiceFiscale not in(
    select distinct D.CodiceFiscale
    from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and
5    D.Cinema = P.Cinema join Film F on P.Film = F.Nome
        where P.Data = dataCorrente and D.Tipo = 'Proiezionario'
        and (
            (ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) >= P.Ora and
10           ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)) <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
            or (ora_casuale >= P.Ora and ADDTIME(P.Ora,
                SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
            or (P.Ora <= ora_casuale and ADDTIME(P.Ora,
                SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
            or (ora_casuale <= P.Ora and ADDTIME(P.Ora,
15           SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_casuale, SEC_TO_TIME(durata_p*60)))
        )
        order by RAND() limit 1
        into proiezionario_casuale;
        if proiezionario_casuale is null then
            select 'Non ci sono proiezionisti disponibili';
            leave while_tag;
        end if;
20        select cinema, j, k;
        insert into Proiezione(Data, Ora, Film, Sala, Dipendente, Cinema, Costo)
        values (dataCorrente, ora_casuale, nome_film, sala_casuale, proiezionario_casuale, cinema,
25        costo_p);
        set k = k + 1;
        end while scegli_ora_sala;
30        end while while_loop;
        set i = i + 1;
        end while while_tag;
35        if sistema_reale then
            set cinema = cinema + 1;
        else
            set cinema = 100;
40        end if;
        close cur_film;
        end while;
45        END$$
DELIMITER ;
50 -----
-- procedure aggiungiProiezioneSenzaProiezionario
-----
55 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`aggiungiProiezioneSenzaProiezionario`;
DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `aggiungiProiezioneSenzaProiezionario`(in data_p DATE, in ora_p TIME, in film varchar(128), in cinema int, in sala int, in
60 costo_p int)
BEGIN
    declare durata_p int;
    declare exit handler for sqlexception
65 begin
        rollback;
        resignal;
    end;
    #non voglio permettere nessun tipo di anomalia perchè l'aggiunta della proiezione
    #coinvolge la ricerca di sale "libere"
    #quindi una anomalia potrebbe
    #provocare il comportamento inaspettato di tale procedura
70    set transaction isolation level serializable;

```

```

start transaction;
select Durata from Film where Nome = film into durata_p;
#verifico che la sala in input si effettivamente disponibile
5 if (select sala where sala not in (
    select distinct S.NumeroDiSala
    from Sala S join Proiezione P on S.NumeroDiSala = P.Sala join Film F on P.Film = F.Nome
    where S.Cinema = cinema and P.Data = data_p
    and (
        (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))
10      <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
        or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
        SEC_TO_TIME(durata_p*60)))
        or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p,
15      SEC_TO_TIME(durata_p*60)))
        or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
        SEC_TO_TIME(durata_p*60))) )is null) then
    signal sqlstate '45004'
    set message_text = 'Sala occupata, usa saleDisponibili per vedere quale sala scegliere';
20 end if;

insert into Proiezione(Data, Ora, Film, Sala, Cinema, Costo)
values (data_p, ora_p, film, sala, cinema, costo_p);
commit;
25 END$$

DELIMITER ;

-----
30 -- procedure aggiornaProiezionistaDellaProiezione
-----

USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`aggiornaProiezionistaDellaProiezione` ;
35 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `aggiornaProiezionistaDellaProiezione` (in proiezionista varchar(45), in data_p DATE, in ora_p TIME, in film varchar(128), in
sala int, in cinema int)
40 BEGIN
    #prima dell'aggiunta del proiezionista alla proiezione,
    #assumo che il gestore abbia consultato i proiezionisti disponibili
    #con la procedura 'proiezionistiDisponibili'
    declare durata_p int;
    declare exit handler for sqlexception
45 begin
        rollback;
        resignal;
    end;
50 set transaction isolation level serializable;
start transaction;
select Durata from Film where Nome = film into durata_p;
#verifico che il dipendente scelto sia un proiezionista
55 if (select proiezionista where proiezionista in (
    select CodiceFiscale
    from Dipendente
    where Tipo = 'Proiezionista' and Cinema = cinema) is null) then
    signal sqlstate '45006'
60     set message_text = 'Il dipendente scelto non e nel cinema dato in input, o non e un proiezionista';
    end if;

#verifico che il proiezionista selezionato sia effettivamente disponibile
if (select proiezionista where proiezionista in (
65     select distinct D.CodiceFiscale
    from Dipendente D join Proiezione P on D.CodiceFiscale = P.Dipendente and D.Cinema = P.Cinema join Film F on P.Film = F.Nome
    where P.Data = data_p and D.Tipo = 'Proiezionista'
    and (
        (ADDTIME(ora_p, SEC_TO_TIME(durata_p*60)) >= P.Ora and ADDTIME(ora_p, SEC_TO_TIME(durata_p*60))
70      <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)))
        or (ora_p >= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
        SEC_TO_TIME(durata_p*60)))
        or (P.Ora <= ora_p and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) >= ADDTIME(ora_p,
        SEC_TO_TIME(durata_p*60))))
```

```

        or (ora_p <= P.Ora and ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) <= ADDTIME(ora_p,
SEC_TO_TIME(durata_p*60))) )is not null) then
      signal sqlstate '45005'
      set message_text = 'Proiezionista occupato, usa proiezionistiDisponibili per vedere quale proiezionista scegliere';
5       end if;

#aggiungo il proiezionista alla proiezione
update Proiezione P
set Dipendente = proiezionista
10 where P.Cinema = cinema and P.Data = data_p and P.Ora = ora and P.Film = film and P.Sala = sala;
commit;

END$$
15 DELIMITER ;

-----
-- procedure calendarioTurni
20 -----
USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`calendarioTurni`;

25 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `calendarioTurni`(in cinema int)
BEGIN
  declare data_corrente DATE default CURDATE();
30
  #voglio evitare le letture sporche e inconsistenti che potrebbero derivare
  #dalla chiamata di tale procedura contemporaneamente alla modifica di un turno di un dipendente

  set transaction isolation level repeatable read;
35
  #stampa i turni a partire dalla data corrente fino la domenica piu' vicina
  week_loop: loop
    select TD.FasciaOrariaTurno as FasciaOraria, TD.GiornoTurno as Giorno, TD.CodiceFiscaleDipendente as Dipendente
40   from TurnoDelDipendente TD join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
    where D.Cinema = cinema and TD.GiornoTurno = data_corrente;

    #se sono arrivato a domenica, mi fermo
    if (DAYNAME(data_corrente) like 'Sun%') then
45      leave week_loop;
      end if;

    #altrimenti incremento il giorno
    set data_corrente = DATE_ADD(data_corrente, interval 1 day);
50
  end loop week_loop;

END$$
55 DELIMITER ;

-----
-- procedure postiDisponibiliDellaProiezione
-----

60 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`postiDisponibiliDellaProiezione`;

65 DELIMITER $$
USE `CatenaCinema`$$
CREATE PROCEDURE `postiDisponibiliDellaProiezione`(in data_p DATE, in ora_p TIME, in film varchar(128), in sala int, in cinema int)
BEGIN
  #voglio evitare le letture sporche e le letture inconsistenti che potrebbero
70  #non far funzionare come dovuto tale procedura

  set transaction isolation level repeatable read;

  #solo se la proiezione esiste vermente
  if ((select count(*)
75

```

```

from Proiezione P
where P.Data = data_p and P.Ora = ora_p and P.Film = film and P.Sala = sala and P.Cinema = cinema) > 0) then
5      select P.Fila, P.Numero, (case
                                when ((select CodiceDiPrenotazione
                                         from Biglietto
                                         where
CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
                                         OraProiezione = ora_p and DataProiezione = data_p and Film = film and
                                         NumeroPosto = P.Numero and FilaPosto = P.Fila) is null
                                         or
                                         (select
                                         from Biglietto
                                         where
15     CodiceDiPrenotazione
                                         CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
                                         OraProiezione = ora_p and DataProiezione = data_p and Film = film and
                                         NumeroPosto = P.Numero and FilaPosto = P.Fila and Tipo = 'Annullato') is not null) then '0'
                                         else '1'
                                         end
                                         ) as Occupato
25
                                         from Posto P
                                         where P.Cinema = cinema and P.Sala = sala
                                         order by P.Fila asc;
                                         end if;
30
END$$

DELIMITER ;

35 -----
-- procedure confermaBiglietto
-----

40 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`confermaBiglietto`;

45 DELIMITER $$$
USE `CatenaCinema`$$
CREATE PROCEDURE `confermaBiglietto`(in codice varchar(45))
BEGIN
    declare exit handler for sqlexception
    begin
        rollback;
50      resignal;
        end;
    #scelta del livello di isolamento come per la procedura 'annullaBiglietto'
55      set transaction isolation level serializable;
        start transaction;

        #verifico l'esistenza del biglietto da annullare
60      if (select CodiceDiPrenotazione
             from Biglietto
             where CodiceDiPrenotazione = codice) is null then
            signal sqlstate '45011'
            set message_text = 'CodiceDiPrenotazione NON trovato';
65      end if;

        #verifico che il biglietto non sia di tipo 'Annullato'
        if (select CodiceDiPrenotazione
             from Biglietto
             where CodiceDiPrenotazione = codice and Tipo = 'Annullato') is not null then
            signal sqlstate '45014'
            set message_text = 'Questo biglietto e stato annullato';
70      end if;

        #verifico che il biglietto non sia di tipo 'Confermato', non posso confermarlo 2 volte
75

```

```

if (select CodiceDiPrenotazione
     from Biglietto
     where CodiceDiPrenotazione = codice and Tipo = 'Confermato') is not null then
    signal sqlstate '45015'
5      set message_text = 'Biglietto già confermato';
     end if;

update Biglietto set Tipo = 'Confermato' where CodiceDiPrenotazione = codice;
10    commit;

END$$

DELIMITER ;

15 --
-- procedure annullaBiglietto
-----

20 USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`annullaBiglietto`;

DELIMITER $$
USE `CatenaCinema`$$
25 CREATE PROCEDURE `annullaBiglietto`(in codice varchar(45))
BEGIN

    declare ora_p TIME;
    declare data_p DATE;

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

#voglio evitare tutte le anomalie, potrei incorrere in lettura inconsistente ed aggiornamento fantasma
#(ad esempio se procedura chiamata contemporaneamente ad un report, per contare la tipologia dei biglietti in un mese)
#se non mettessi il massimo livello di isolamento
40    set transaction isolation level serializable;

        start transaction;

45    #verifico l'esistenza del biglietto da annullare
    if (select CodiceDiPrenotazione
         from Biglietto
         where CodiceDiPrenotazione = codice) is null then
        signal sqlstate '45011'
50        set message_text = 'CodiceDiPrenotazione NON trovato';
     end if;

#verifico che il biglietto non sia di tipo 'Confermato', sono questi posso annullare
55    if (select CodiceDiPrenotazione
         from Biglietto
         where CodiceDiPrenotazione = codice and Tipo = 'Confermato') is not null then
        signal sqlstate '45012'
50        set message_text = 'Puoi annullare solo biglietto di tipo Confermato';
     end if;

60    select OraProiezione, DataProiezione from Biglietto where CodiceDiPrenotazione = codice into ora_p, data_p;

        #verifico che non mi trovo nella mezz'ora prima di inizio del film
65    if ((CURDATE() = data_p and ADDTIME(ora_p, -CURTIME()) < '00:30:00') or
        CURDATE() > data_p) then
        signal sqlstate '45013'
        set message_text = 'NON puoi annullare il biglietto prima di mezz'ora dall'inizio del film o dopo la scadenza dello stesso';
     end if;

70    update Biglietto set Tipo = 'Annullato' where CodiceDiPrenotazione = codice;

        commit;

END$$
75

```

DELIMITER ;

-- procedure proiezioniDisponibili

5

USE `CatenaCinema`;  
DROP procedure IF EXISTS `CatenaCinema`.`proiezioniDisponibili`;

10 DELIMITER \$\$

USE `CatenaCinema`\$\$

CREATE PROCEDURE `proiezioniDisponibili`(in film varchar(128), in cinema int)  
BEGIN

15 # mostro tutte le proiezioni a partire dalla data corrente in poi,  
# con ora di inizio maggiore o uguale all ora attuale,

# voglio evitare letture sporche ed inconsistenti

20 set transaction isolation level repeatable read;

select P.Data, P.Ora, P.Sala  
from Proiezione P  
where P.Data >= CURDATE() and P.Ora >= CURTIME() and P.Film = film and P.Cinema = cinema;

25

END\$\$

DELIMITER ;

30

-- procedure popolaBiglietti

35 USE `CatenaCinema`;  
DROP procedure IF EXISTS `CatenaCinema`.`popolaBiglietti`;

DELIMITER \$\$

USE `CatenaCinema`\$\$

40 CREATE PROCEDURE `popolaBiglietti`(in cinema int, in sistema\_reale int, in num\_proiezioni int)  
BEGIN

#procedura che aggiunge dei biglietti casuali alle proiezioni di un cinema

#se sono presenti dei biglietti nel db, meglio chiamare acquistaBiglietto

45 declare i int;

declare j int;

declare data\_p DATE;

declare ora\_p TIME;

declare num\_bigl int;

50 declare capienza\_sala int;

declare numero\_posto int;

declare fila\_posto varchar(1);

declare nomeFilm varchar(128);

declare sala int;

55

declare exit handler for sqlexception

begin

rollback;

resignal;

60 end;

#procedura usata per popolare il db, usata solo nel test, ed una sola volta,  
#non mi preoccupo del livello di isolamento

65 if (select count(\*)

from Biglietto where CinemaDellaProiezione = cinema) > 0 then

signal sqlstate '45016'

set message\_text = 'Dei biglietti sono già presenti in db. Popola con acquistaBiglietto';

end if;

70

if sistema\_reale = 1 then

set cinema = 1;

end if;

75 if num\_proiezioni > 12 then

```

    signal sqlstate '45016'
    set message_text = 'NON puo essere num_proiezioni > 12';
end if;

5   while cinema <= (select count(*) from Cinema) do

    set i = 0;
    #1)scelgo num_proiezioni a cui far associare i biglietti
    while _proi: while i < num_proiezioni do
10      #2)estraggo i dati della proiezione casuale nelle variabili dichiarate,
      select P.Data, P.Ora, P.Film, P.Sala
      from Proiezione P
      where P.Data >= CURDATE() and P.Ora >= CURTIME() and P.Film = film and P.Cinema = cinema
      order by RAND() limit 1 into data_p, ora_p, nomeFilm, sala;
15      #se ha già dei biglietti associati ritento,
      if(select CodiceDiPrenotazione
          from Biglietto
          where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
20          OraProiezione = ora_p and DataProiezione = data_p and Film = film limit 1) is not null then
          iterate while _proi;
      end if;

      #3)associo tale proiezione ad un numero casuale di biglietti,
25      # che va dal 5% al 90% della capienza della sala
      select S.NumeroDiPosti from Sala S where S.NumeroDiSala = sala and S.Cinema = cinema into capienza_sala;
      select FLOOR(((RAND()*(90-5))+5)/100)*capienza_sala) into num_bigl;

30      set j = 0;
      while j < num_bigl do

        #4)estraggo un numero e fila di posto casuale, tra quelli disponibili
        select P.Fila, P.Numero
            from Posto P
            where P.Cinema = cinema and P.Sala = sala and ((select CodiceDiPrenotazione
35              from Biglietto
              where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
40              OraProiezione = ora_p and DataProiezione = data_p and Film = nomeFilm and
                  NumeroPosto = P.Numero and FilaPosto = P.Fila) is null
                  or
45              (select CodiceDiPrenotazione
                  from Biglietto
                  where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and
                      OraProiezione = ora_p and DataProiezione = data_p and Film = nomeFilm and
                          NumeroPosto = P.Numero and FilaPosto = P.Fila and Tipo = 'Annullato') is not null)
50              order by RAND() limit 1
55              into fila_posto, numero_posto;

        #per completare la procedura di acquisto biglietto un cliente dovrebbe inserire i dati relativi alla
        #sua carta di credito. Inseriti gli stessi dati relativi alla CC per tutti i biglietto inseriti in fase di
60        test

        call acquistaBiglietto(cinema, sala, data_p, ora_p, nomeFilm, numero_posto, fila_posto, '5255294813425786', 'Test', 'Tester', 2, 22, 884,
@cdp);
65        set j = j + 1;

        end while;

        set i = i + 1;

70        end while while _proi;

        if sistema_reale = 1 then
            set cinema = cinema + 1;
        else
            set cinema = 100;
75

```

```

        end if;

    end while;

5    END$$

DELIMITER ;

-----
10 -- procedure acquistaBiglietto
-----

USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`acquistaBiglietto`;

15 DELIMITER $$$
USE `CatenaCinema`$$
CREATE PROCEDURE `acquistaBiglietto`(in cinema int, in sala int, in data_p DATE, in ora_p TIME, in film varchar(128), in numero_posto int, in
20 fila_posto varchar(1), in numero varchar(16), in nomeIntest varchar(45), in cognomeIntest varchar(45), in meseScad int, in annoScad int, in cvv int, out
codice_di_p varchar(45))
BEGIN

    declare codice varchar(45);

25    declare exit handler for sqlexception
begin
        rollback;
        resignal;
        end;
   30    declare exit handler for 1062
begin
        rollback;
        signal sqlstate '45014';
   35        set message_text = 'Sei stato battuto sul tempo. Qualcuno ha appena occupato il tuo stesso posto.';
        end;

    set transaction isolation level serializable;

40    start transaction;

    #prime di procedere controllo che la carta di credito sia presente nel db,
    #se cosi non fosse devo inserirla prima di continuare la procedura di acquisto del biglietto

45    if (not cartaEsistente('numero')) then
        insert into CartaDiCredito values(numero, nomeIntest, cognomeIntest, meseScad, annoScad, cvv);
        end if;

50    set codice = concat(data_p);
    set codice = concat(codice, cinema);
    set codice = concat(codice, ora_p);
    set codice = concat(codice, sala);
    set codice = concat(codice, CURTIME());
    set codice = concat(codice, fila_posto);
   55    set codice = concat(codice, numero_posto);

        #verifico che la proiezione inserita esista
if( select P.Cinema
60        from Proiezione P
        where P.Data = data_p and P.Ora = ora_p and P.Film = film and P.Cinema = cinema and P.Sala = sala
            and
                #e che i posti inseriti siano effettivamente disponibili
                (fila_posto, numero_posto) in (select P.Fila, P.Numero
65                    from Posto P
                    where P.Cinema = cinema and P.Sala = sala and ((select B.CodiceDiPrenotazione
                        from Biglietto B
                            where B.CinemaDellaProiezione = cinema and B.SalaDellaProiezione = sala and
                                B.OraProiezione = ora_p and B.DataProiezione = data_p and B.Film = film and
                                    B.NumeroPosto = P.Numero and B.FilaPosto = P.Fila) is null
75                            or
                                )) ) ) )

```

```

(select B.CodiceDiPrenotazione
from Biglietto B
where B.CinemaDellaProiezione = cinema and B.SalaDellaProiezione = sala and
      B.OraProiezione = ora_p and B.DataProiezione = data_p and B.Film = film and
      B.NumeroPosto = P.Numero and B.FilaPosto = P.Fila and B.Tipo = 'Annullato' is not null)) is
5      not null then
10      not null then
15      insert into Biglietto values (codice, 'AcquistatoNonConfermato', numero_posto, fila_posto, data_p, film, sala, ora_p, cinema,
numero);
           select codice into codice_di_p;
           end if;
20      commit;
25      END$$
DELMITER ;
-----  

-- procedure simulaPostProiezione
-----  

30      USE `CatenaCinema`;
DROP procedure IF EXISTS `CatenaCinema`.`simulaPostProiezione`;  

35      DELIMITER $$  

USE `CatenaCinema`$$  

CREATE PROCEDURE `simulaPostProiezione` (in cinema int, in sala int, in film varchar(128), in data_p DATE, in ora_p TIME)
BEGIN  

#procedura usata per test,  

#per simulare l'acquisto e conferma e annullamento di biglietti di una proiezione  

40      declare rand int;
declare codice varchar(45);
declare num_bigl int;
declare i int default 0;
45      declare cur cursor for select CodiceDiPrenotazione
           from Biglietto
           where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and Film = film
                 and DataProiezione = data_p and OraProiezione = ora_p;
50      declare exit handler for sqlexception
begin
      rollback;
      resignal;
55      end;
      select count(CodiceDiPrenotazione)
           from Biglietto
           where CinemaDellaProiezione = cinema and SalaDellaProiezione = sala and Film = film
                 and DataProiezione = data_p and OraProiezione = ora_p into num_bigl;
60      #verifico che ci siano dei biglietti associati alla proiezione
      if num_bigl = 0 then
          signal sqlstate '45017';
65          set message_text = 'Nessun biglietto presente per la prenotazione in input';
      end if;
      open cur;
      while (i < num_bigl) do
70          fetch cur into codice;
          set rand = FLOOR(RAND()*101);
          if( 0 <= rand and rand <= 5 ) then
              call annullaBiglietto(codice);
          end if;
75

```

```

if(6 <= rand and rand <= 90) then
    call confermaBiglietto(codice);
end if;
set i = i + 1;
5
end while;
close cur;

END$$

10 DELIMITER ;
SET SQL_MODE = "";
GRANT USAGE ON *.* TO Gestore;
DROP USER Gestore;
15 SET SQL_MODE="TRADITIONAL,ALLOW_INVALID_DATES";
CREATE USER 'Gestore' IDENTIFIED BY 'gestore';

GRANT ALL ON TABLE CatenaCinema.* TO 'Gestore';
GRANT ALL ON CatenaCinema.* TO 'Gestore';
20 SET SQL_MODE = "";
GRANT USAGE ON *.* TO Cliente;
DROP USER Cliente;
SET SQL_MODE="TRADITIONAL,ALLOW_INVALID_DATES";
CREATE USER 'Cliente' IDENTIFIED BY 'cliente';

25 GRANT SELECT ON TABLE `CatenaCinema`.`Cinema` TO 'Cliente';
GRANT SELECT ON TABLE `CatenaCinema`.`Sala` TO 'Cliente';
GRANT SELECT ON TABLE `CatenaCinema`.`Posto` TO 'Cliente';
GRANT SELECT ON TABLE `CatenaCinema`.`Proiezione` TO 'Cliente';
30 GRANT SELECT ON TABLE `CatenaCinema`.`Film` TO 'Cliente';
GRANT SELECT ON TABLE `CatenaCinema`.`Attore` TO 'Cliente';
GRANT SELECT ON TABLE `CatenaCinema`.`AttoriNelFilm` TO 'Cliente';
GRANT ALL ON procedure `CatenaCinema`.`acquistaBiglietto` TO 'Cliente';
GRANT ALL ON procedure `CatenaCinema`.`annullaBiglietto` TO 'Cliente';
35 GRANT ALL ON procedure `CatenaCinema`.`postiDisponibiliDellaProiezione` TO 'Cliente';
GRANT ALL ON procedure `CatenaCinema`.`proiezioniDisponibili` TO 'Cliente';
SET SQL_MODE = "";
GRANT USAGE ON *.* TO Visitatore;
DROP USER Visitatore;
40 SET SQL_MODE="TRADITIONAL,ALLOW_INVALID_DATES";
CREATE USER 'Visitatore' IDENTIFIED BY 'visitatore';

GRANT SELECT ON TABLE `CatenaCinema`.`Cinema` TO 'Visitatore';
GRANT SELECT ON TABLE `CatenaCinema`.`Sala` TO 'Visitatore';
45 GRANT SELECT ON TABLE `CatenaCinema`.`Posto` TO 'Visitatore';
GRANT SELECT ON TABLE `CatenaCinema`.`Proiezione` TO 'Visitatore';
GRANT SELECT ON TABLE `CatenaCinema`.`Film` TO 'Visitatore';
GRANT SELECT ON TABLE `CatenaCinema`.`Attore` TO 'Visitatore';
GRANT SELECT ON TABLE `CatenaCinema`.`AttoriNelFilm` TO 'Visitatore';
50 GRANT ALL ON procedure `CatenaCinema`.`proiezioniDisponibili` TO 'Visitatore';
SET SQL_MODE = "";
GRANT USAGE ON *.* TO Maschera;
DROP USER Maschera;
SET SQL_MODE="TRADITIONAL,ALLOW_INVALID_DATES";
55 CREATE USER 'Maschera' IDENTIFIED BY 'maschera';

GRANT ALL ON TABLE `CatenaCinema`.`Biglietto` TO 'Maschera';
GRANT ALL ON procedure `CatenaCinema`.`confermaBiglietto` TO 'Maschera';

60 SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

-----
65 -- Data for table `CatenaCinema`.`Cinema`
-----
START TRANSACTION;
USE `CatenaCinema`;
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (1);
70 INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (2);
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (3);
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (4);
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (5);
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (6);
75 INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (7);

```

```

INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (8);
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (9);
INSERT INTO `CatenaCinema`.`Cinema` ('ID') VALUES (10);

5   COMMIT;

USE `CatenaCinema`;

DELIMITER $$

10  USE `CatenaCinema$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`Proiezione_BEFORE_INSERT` $$
USE `CatenaCinema$$
CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Proiezione_BEFORE_INSERT` BEFORE INSERT ON `Proiezione` FOR
15  EACH ROW
BEGIN

    declare pom varchar(10) default '16-20';
    declare ser varchar(10) default '20-24';
20    declare entrambi int default 0;
    declare maschereDisponibili int;
    declare maschereDisponibiliSerali int;
    declare maschereDisponibiliPomeridiane int;
    declare mascheraCasuale varchar(45);
25    declare durata_p int;
    declare fasciaOraria varchar(10);
    #se un proiezionista viene associato ad una proiezione, si deve verificare che
    #quel giorno a quell'ora sia presente in 'TurnoDelDipendente', altrimenti deve essere aggiunto
    #e dunque che quel giorno (data della proiezione) in una certa fascia oraria(da determinare in base
30    #all'ora della proiezione) deve essere presente una certa tupla un 'Turno'

    select F.Durata
    from Film F
    where F.Nome = NEW.Film
35    into durata_p;

    #identifico la faccia oraria in base all'ora della proiezione
    if ('16:00:00' <= NEW.Ora and NEW.Ora <= '20:00:00') then
        set fasciaOraria = pom;
40        end if;
    if ('20:00:00' <= NEW.Ora and NEW.Ora <= '24:00:00') then
        set fasciaOraria = ser;
        end if;
    #se ho una proiezione che si sovrappone tra i 2 turni
45    if (NEW.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(NEW.Ora, SEC_TO_TIME(durata_p*60))) then
        set entrambi = 1;
    end if;
    if ('24:00:00' < NEW.Ora or NEW.Ora < '16:00:00') then
        signal sqlstate '45009';
50        set message_text = 'Ora della proiezione non valida, deve essere compresa tra le 16 e le 24';
    end if;

    if NEW.Dipendente is not null then
        if not entrambi then
55            #prima verifico che sia presente nella tabella Turno la tupla:
            if ((select FasciaOraria
                  from Turno T
                  where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = fasciaOraria)
60                is null) then
                    #se non è presente, la creo
                    insert into Turno values (fasciaOraria, NEW.Data, NEW.Cinema);
                end if;
                #se è presente devo verificare che il dipendente (se è stato selezionato per la proiezione
                #allora è sicuramente libero per come funzionano le procedure sviluppate) sia associato a quel turno
65                #se non lo fosse lo associo
                if ((select CodiceFiscaleDipendente
                      from TurnoDelDipendente TD
                      where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and
TD.FasciaOrariaTurno = fasciaOraria)
70                    is null )then
                    insert into TurnoDelDipendente
                    values (fasciaOraria, NEW.Data, NEW.Dipendente);
                end if;
            end if;
        end if;
75        #faccio in modo che ci siano almeno 2 maschere differenti per la fascia oraria in cui ho la proiezione

```

```

select count(distinct TD.CodiceFiscaleDipendente)
      from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
      where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = fasciaOraria and
5   D.Tipo = 'Maschera'
      into maschereDisponibili;
      #per tante volte quante sono le maschere mancanti per arrivare a 2
      #aggiungo al turno una maschera a caso disponibile.
      #definisco una maschera disponibile per un cinema se:
      #1)lavora per quel cinema
10    #2)non ha un turno associato alla fascia oraria della proiezione
      while maschereDisponibili < 2 do
          select CodiceFiscale
              from Dipendente
15      where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
                      select D.CodiceFiscale
                          from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale =
TD.CodiceFiscaleDipendente
                      and D.Cinema = NEW.Cinema
20      where TD.FasciaOrariaTurno = fasciaOraria and TD.GiornoTurno = NEW.Data and D.Tipo =
'Maschera')
                      order by RAND() limit 1 into mascheraCasuale;
25      insert into TurnoDelDipendente
                      values (fasciaOraria, NEW.Data, mascheraCasuale);
      set maschereDisponibili = maschereDisponibili +1;
      end while;
30      #se il proiezionista deve gestire una proiezione tale per cui gli occupa entrambi i turni della giornata:
      else
          #devo verificare che sia presente sia il turno pomeridiano che quello serale
          if ((select FasciaOraria
                  from Turno T
                  where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = pom)
              is null) then
              insert into Turno values (pom, NEW.Data, NEW.Cinema);
          end if;
          if ((select FasciaOraria
                  from Turno T
                  where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = ser)
              is null) then
              insert into Turno values (ser, NEW.Data, NEW.Cinema);
          end if;
40      #resta da verificare che il dipendente scelto per la gestione della proiezione
      #sia associato ai turni sia serale che pomeridiano
      if ((select CodiceFiscaleDipendente
                  from TurnoDelDipendente TD
                  where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and
50      TD.FasciaOrariaTurno = pom)
          is null )then
          insert into TurnoDelDipendente
                  values (pom, NEW.Data, NEW.Dipendente);
      end if;
      if ((select CodiceFiscaleDipendente
                  from TurnoDelDipendente TD
                  where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and
55      TD.FasciaOrariaTurno = ser)
          is null )then
          insert into TurnoDelDipendente
                  values (ser, NEW.Data, NEW.Dipendente);
      end if;
60      #stessa cosa per le maschere, almeno 2 per il turno di sera e quello pomeridiano
      select count(distinct TD.CodiceFiscaleDipendente)
          from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
          where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = ser and D.Tipo =
'Maschera'
          into maschereDisponibiliSerali;
70      select count(distinct TD.CodiceFiscaleDipendente)
          from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
          where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = pom and D.Tipo =
'Maschera'
          into maschereDisponibiliPomeridiane;
75

```

```

while maschereDisponibiliSerali < 2 do
    select CodiceFiscale
    from Dipendente
    where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
        select D.CodiceFiscale
        from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale =
        TD.CodiceFiscaleDipendente
            and D.Cinema = NEW.Cinema
            where TD.FasciaOrariaTurno = ser and TD.GiornoTurno = NEW.Data and D.Tipo =
        'Maschera')
    order by RAND() limit 1 into mascheraCasuale;

    insert into TurnoDelDipendente
    values (ser, NEW.Data, mascheraCasuale);

    set maschereDisponibiliSerali = maschereDisponibiliSerali +1;

    end while;
    while maschereDisponibiliPomeridiane < 2 do
        select CodiceFiscale
        from Dipendente
        where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
            select D.CodiceFiscale
            from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale =
            TD.CodiceFiscaleDipendente
                and D.Cinema = NEW.Cinema
                where TD.FasciaOrariaTurno = pom and TD.GiornoTurno = NEW.Data and D.Tipo =
        'Maschera')
        order by RAND() limit 1 into mascheraCasuale;

        insert into TurnoDelDipendente
        values (pom, NEW.Data, mascheraCasuale);

        set maschereDisponibiliPomeridiane = maschereDisponibiliPomeridiane +1;

        end while;
        end if;
    end if;
#se e stata inserita una proiezione senza specificare il dipendente
#non faccio nulla

END$$
45
USE `CatenaCinema`$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`Proiezione_AFTER_UPDATE` $$

50 CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Proiezione_AFTER_UPDATE` AFTER UPDATE ON `Proiezione` FOR EACH ROW
BEGIN

    declare pom varchar(10) default '16-20';
    declare ser varchar(10) default '20-24';
    declare fasciaOraria varchar(10);
    declare entrambi int default 0;
    declare durata_p int;

    declare maschereDisponibili int;
    declare maschereDisponibiliSerali int;
    declare maschereDisponibiliPomeridiane int;
    declare mascheraCasuale varchar(45);

60    if (OLD.Dipendente is not null) then

        select F.Durata
        from Film F
        where F.Nome = OLD.Film
        into durata_p;

        if ('16:00:00' <= OLD.Ora and OLD.Ora <= '20:00:00') then
            set fasciaOraria = pom;
        end if;
        if ('20:00:00' <= OLD.Ora and OLD.Ora <= '24:00:00') then
75

```

```

      set fasciaOraria = ser;
      end if;
      #se ho una proiezione che si sovrappone tra i 2 turni
      if (OLD.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(OLD.Ora, SEC_TO_TIME(durata_p*60))) then
      5      set entrambi = 1;
      end if;

      # se il proiezionista non ha associate altre proiezioni per quel giorno a quella fascia oraria
      # gli cancello il turno, o entrambi i turni, qualora la proiezione che verra occupata gli occupi
      10     # entrambi i turni
      if not entrambi then

          if ((select count(*)
               from Proiezione P join Film F on P.Film = F.Nome
               where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
               15           # calcolo la faccia oraria per quel film
               fasciaOraria = (case
                               # se in quel giorno il
                               # entrambi i turni, non
                               when P.Ora <= '20:00:00'
                               when '16:00:00' <= P.Ora
                               when '16:00:00' <= P.Ora
                               and P.Ora <= '20:00:00' then pom
                               20           when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
                               end)) = 0) then
                               #se e 0 elimino il turno del dipendente a quella fascia oraria
                               delete
                               25         from TurnoDelDipendente
                               where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = fasciaOraria and GiornoTurno = OLD.Data;

                               #verifico che ci siano ALTRE proiezioni per quella fascia oraria, se no, elimino le maschere di quella fascia oraria
                               if ((select count(*)
                               30           from Proiezione P join Film F on P.Film = F.Nome
                               where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <> (OLD.Ora, OLD.Dipendente)
                               and fasciaOraria = (case
                               35           when P.Ora <= '20:00:00'
                               when '16:00:00' <= P.Ora
                               and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) then fasciaOraria
                               40           when '16:00:00' <= P.Ora
                               and P.Ora <= '20:00:00' then pom
                               45           when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
                               end)) = 0) then
                               #se non ho proiezioni per quella fasciaOraria, non mi serovno maschere
                               delete
                               50         from TurnoDelDipendente
                               where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
                               (select D.CodiceFiscale
                               from Dipendente D
                               where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');

                               end if;

                               end if;
      55     #se e maggiore di 0, non lo elimino, il dipendente ha un turno associato ad un altra proiezinoe

      else
          #se la proiezione che si vuole cancellare occupava al dipendente entrambi i turni,
          #verifico sia per il turno di sera che di pomeriggio che il dipendente abbia schedulete
          60     # altre proiezinoi, in caso contrario, gli cancello il turno (uno dei due o entrambi)
          if ((select count(*)
               from Proiezione P join Film F on P.Film = F.Nome
               where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
               ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
               65       P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
               delete
               from TurnoDelDipendente
               where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = ser and
               GiornoTurno = OLD.Data;
      70     #analogalemente a sopra
     #elimino le 2 maschere serali se oltre la proiezione da eliminare non ne ho altre nella stessa fscia oraria
     if ((select count(*)
          from Proiezione P join Film F on P.Film = F.Nome

```

```

      where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <>
      (OLD.Ora, OLD.Dipendente) and
      ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
      P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) ) = 0) then
      5   delete
          from TurnoDelDipendente
          where GiornoTurno = OLD.Data and FasciaOrariaTurno = ser and CodiceFiscaleDipendente in
              (select D.CodiceFiscale
               from Dipendente D
               where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
      10  end if;

      end if;

      15  if ((select count(*)
           from Proiezione P join Film F on P.Film = F.Nome
           where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
           ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
           P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) ) = 0) then
      20    delete
          from TurnoDelDipendente
          where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = pom and
          GiornoTurno = OLD.Data;
      25  if ((select count(*)
           from Proiezione P join Film F on P.Film = F.Nome
           where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <>
      (OLD.Ora, OLD.Dipendente) and
           ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
      30    P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) ) = 0) then
          delete
          from TurnoDelDipendente
          where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
              (select D.CodiceFiscale
               from Dipendente D
               where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
      35  end if;

          end if;

      40  end if;

      end if;

      45  select F.Durata
          from Film F
          where F.Nome = NEW.Film
          into durata_p;

      50  #identifico la faccia oraria in base all'ora della proiezione
          if ('16:00:00' <= NEW.Ora and NEW.Ora <= '20:00:00') then
              set fasciaOraria = pom;
          end if;
          if ('20:00:00' <= NEW.Ora and NEW.Ora <= '24:00:00') then
      55    set fasciaOraria = ser;
          end if;
          #se ho una proiezione che si sovrappone tra i 2 turni
          if (NEW.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(NEW.Ora, SEC_TO_TIME(durata_p*60))) then
              set entrambi = 1;
      60  end if;
          if ('24:00:00' < NEW.Ora or NEW.Ora < '16:00:00') then
              signal sqlstate '45009'
              set message_text = 'Ora della proiezione non valida, deve essere compresa tra le 16 e le 24';
          end if;
      65  if NEW.Dipendente is not null then

              if not entrambi then
                  #prima verifico che sia presente nella tabella Turno la tupla:
                  if ((select FasciaOraria
                       from Turno T
                       where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = fasciaOraria)
                      is null) then
                      #se non è presente, la creo
                      insert into Turno values (fasciaOraria, NEW.Data, NEW.Cinema);
      70
      75

```

```

end if;
#se è presente devo verificare che il dipendente (se è stato selezionato per la proiezione
#allora è sicuramente libero per come funzionano le procedure sviluppate) sia associato a quel turno
#e se non lo fosse lo associo
5   if((select CodiceFiscaleDipendente
        from TurnoDelDipendente TD
        where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and
        TD.FasciaOrariaTurno = fasciaOraria)
10    is null )then
        insert into TurnoDelDipendente
        values (fasciaOraria, NEW.Data, NEW.Dipendente);
    end if;

#faccio in modo che ci siano almeno 2 maschere differenti per la fascia oraria in cui ho la proiezione
15  select count(distinct TD.CodiceFiscaleDipendente)
      from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
      where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = fasciaOraria and
      D.Tipo = 'Maschera'
20    into maschereDisponibili;
    #per tante volte quante sono le maschere mancanti per arrivare a 2
    #aggiungo al turno una maschera a caso disponibile.
    #definisco una maschera disponibile per un cinema se:
    #1)lavora per quel cinema
    #2)non ha un turno associato alla fascia oraria della proiezione
25  while maschereDisponibili < 2 do
      select CodiceFiscale
      from Dipendente
      where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
30        select D.CodiceFiscale
        from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale =
        TD.CodiceFiscaleDipendente
          and D.Cinema = NEW.Cinema
          where TD.FasciaOrariaTurno = fasciaOraria and TD.GiornoTurno = NEW.Data and D.Tipo =
35  'Maschera')
      order by RAND() limit 1 into mascheraCasuale;

      insert into TurnoDelDipendente
      values (fasciaOraria, NEW.Data, mascheraCasuale);

40  set maschereDisponibili = maschereDisponibili +1;

end while;
#se il proiezionista deve gestire una proiezione tale per cui gli occupa entrambi i turni della giornata:
45  else
      #devo verificare che sia presente sia il turno pomeridiano che quello serale
      if ((select FasciaOraria
            from Turno T
            where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = pom)
50      is null) then
          insert into Turno values (pom, NEW.Data, NEW.Cinema);

      end if;
      if((select FasciaOraria
            from Turno T
            where T.Giorno = NEW.Data and T.Cinema = NEW.Cinema and T.FasciaOraria = ser)
55      is null) then
          insert into Turno values (ser, NEW.Data, NEW.Cinema);

      end if;
      #resta da verificare che il dipendente scelto per la gestione della proiezione
      #sia associato ai turni sia serale che pomeridiano
60      if((select CodiceFiscaleDipendente
            from TurnoDelDipendente TD
            where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and
            TD.FasciaOrariaTurno = pom)
65      is null )then
          insert into TurnoDelDipendente
          values (pom, NEW.Data, NEW.Dipendente);

      end if;
      if((select CodiceFiscaleDipendente
            from TurnoDelDipendente TD
            where TD.CodiceFiscaleDipendente = NEW.Dipendente and TD.GiornoTurno = NEW.Data and
            TD.FasciaOrariaTurno = ser)
70      is null )then
          insert into TurnoDelDipendente
          values (ser, NEW.Data, NEW.Dipendente);

      end if;

```

```

    end if;
#stessa cosa per le maschere, almeno 2 per il turno di sera e quello pomeridiano
select count(distinct TD.CodiceFiscaleDipendente)
      from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
      where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = ser and D.Tipo =
5      'Maschera'
      into maschereDisponibiliSerali;

      select count(distinct TD.CodiceFiscaleDipendente)
      from TurnoDelDipendente TD right join Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale
      where TD.GiornoTurno = NEW.Data and D.Cinema = NEW.Cinema and TD.FasciaOrariaTurno = pom and D.Tipo =
10     'Maschera'
      into maschereDisponibiliPomeridiane;

15     while maschereDisponibiliSerali < 2 do

          select CodiceFiscale
          from Dipendente
          where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
20          select D.CodiceFiscale
          from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale =
              TD.CodiceFiscaleDipendente
              and D.Cinema = NEW.Cinema
              where TD.FasciaOrariaTurno = ser and TD.GiornoTurno = NEW.Data and D.Tipo =
25     'Maschera')
          order by RAND() limit 1 into mascheraCasuale;

          insert into TurnoDelDipendente
          values (ser, NEW.Data, mascheraCasuale);

30     set maschereDisponibiliSerali = maschereDisponibiliSerali +1;

     end while;
     while maschereDisponibiliPomeridiane < 2 do
35     select CodiceFiscale
     from Dipendente
     where Tipo = 'Maschera' and Cinema = NEW.Cinema and CodiceFiscale not in(
         select D.CodiceFiscale
         from Dipendente D left join TurnoDelDipendente TD on D.CodiceFiscale =
40     TD.CodiceFiscaleDipendente
         and D.Cinema = NEW.Cinema
         where TD.FasciaOrariaTurno = pom and TD.GiornoTurno = NEW.Data and D.Tipo =
45     'Maschera')
     order by RAND() limit 1 into mascheraCasuale;

         insert into TurnoDelDipendente
         values (pom, NEW.Data, mascheraCasuale);

50     set maschereDisponibiliPomeridiane = maschereDisponibiliPomeridiane +1;

     end while;
     end if;
55     end if;
#se e stata inserita una proiezione senza specificare il dipendente
#non faccio nulla
END$$

60 USE `CatenaCinema`$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`Proiezione_BEFORE_DELETE` $$
USE `CatenaCinema`$$
CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Proiezione_BEFORE_DELETE` BEFORE DELETE ON `Proiezione` FOR
65 EACH ROW
BEGIN
    #prima di cancellare una proiezione devo eliminare l'associazione tra il proiezionista
    #(qualora esistesse) e la proiezione stessa, e controllare se nella data e fascia oraria
    #non ci sono altre proiezioni devo eliminare anche le maschere che dovevano timbrare i biglietti all ingresso
70 #se elimino qualche turno di qualche tipo di dipendente, devo eliminare anche
#il turno associato al cinema, se la proiezione eliminata era l'unica di quella data in quella fascia oraria

    declare pom varchar(10) default '16-20';
    declare ser varchar(10) default '20-24';

```

```

declare fasciaOraria varchar(10);
declare entrambi int default 0;
declare durata_p int;

5   if (OLD.Dipendente is not null) then

       select F.Durata
       from Film F
       where F.Nome = OLD.Film
       into durata_p;

10
15
20
25
30
35
40
45
50
55
60
65
70
75

       if ('16:00:00' <= OLD.Ora and OLD.Ora <= '20:00:00') then
           set fasciaOraria = pom;
       end if;
       if ('20:00:00' <= OLD.Ora and OLD.Ora <= '24:00:00') then
           set fasciaOraria = ser;
       end if;
       #se ho una proiezione che si sovrappone tra i 2 turni
       if (OLD.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(OLD.Ora, SEC_TO_TIME(durata_p*60))) then
           set entrambi = 1;
       end if;

       # se il proiezionista non ha associate altre proiezioni per quel giorno a quella fascia oraria
       # gli cancello il turno, o entrambi i turni, qualora la proiezione che verra occupata gli occupi
       # entrambi i turni
       if not entrambi then

           if ((select count(*)
                 from Proiezione P join Film F on P.Film = F.Nome
                 where P.Ora >< OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
                       # calcolo la faccia oraria per quel film
                       fasciaOraria = (case
                           # se in quel giorno il
                           proiezionista ha un altro film che gli occupa
                           # entrambi i turni, non
                           voglio cancellarglieli
                           when P.Ora <= '20:00:00'
                           when '16:00:00' <= P.Ora
                           and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) then fasciaOraria
                           when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
                           when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then
                               end)) = 0) then
                           #se e 0 elimino il turno del dipendente a quella fascia oraria
                           delete
                           from TurnoDelDipendente
                           where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = fasciaOraria and GiornoTurno = OLD.Data;

                           #verifico che ci siano ALTRE proiezioni per quella fascia oraria, se no, elimino le maschere di quella fascia oraria
                           if ((select count(*)
                                 from Proiezione P join Film F on P.Film = F.Nome
                                 where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) >< (OLD.Ora, OLD.Dipendente)
                                       and fasciaOraria = (case
                                           when P.Ora <= '20:00:00'
                                           when '16:00:00' <= P.Ora
                                           and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) then fasciaOraria
                                           when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then ser
                                           when '20:00:00' <= P.Ora and P.Ora <= '24:00:00' then
                                               end)) = 0) then
                                   #se non ho proiezioni per quella fasciaOraria, non mi servono maschere
                                   delete
                                   from TurnoDelDipendente
                                   where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
                                         (select D.CodiceFiscale
                                         from Dipendente D
                                         where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');

                           end if;

                           end if;
                           #se e maggiore di 0, non lo elimino, il dipendente ha un turno associato ad un altra proiezione
                           else
                               #se la proiezione che si vuole cancellare occupava al dipendente entrambi i turni,
                               #verifico sia per il turno di sera che di pomeriggio che il dipendente abbia schedule
                               # altre proiezini, in caso contrario, gli cancello il turno (uno dei due o entrambi)

```

```

        if ((select count(*)
              from Proiezione P join Film F on P.Film = F.Nome
            where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
                  ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
5           P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
              delete
              from TurnoDelDipendente
              where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = ser and
GiornoTurno = OLD.Data;
10
#analogalemente a sopra
#elimino le 2 maschere serali se oltre la proiezione da eliminare non ne ho altre nella stessa fascia oraria
if ((select count(*)
      from Proiezione P join Film F on P.Film = F.Nome
      where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <>
15
(OLD.Ora, OLD.Dipendente) and
      ('20:00:00' <= P.Ora and P.Ora <= '24:00:00' or
       P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
      delete
20
      from TurnoDelDipendente
      where GiornoTurno = OLD.Data and FasciaOrariaTurno = ser and CodiceFiscaleDipendente in
            (select D.CodiceFiscale
              from Dipendente D
              where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
25
      end if;

      end if;

      if ((select count(*)
            from Proiezione P join Film F on P.Film = F.Nome
            where P.Ora <> OLD.Ora and P.Data = OLD.Data and P.Dipendente = OLD.Dipendente and P.Cinema = OLD.Cinema and
                  ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
30
                  P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
                  delete
                  from TurnoDelDipendente
                  where CodiceFiscaleDipendente = OLD.Dipendente and FasciaOrariaTurno = pom and
35
GiornoTurno = OLD.Data;
40
if ((select count(*)
      from Proiezione P join Film F on P.Film = F.Nome
      where P.Data = OLD.Data and P.Cinema = OLD.Cinema and (P.Ora, P.Dipendente) <>
(OLD.Ora, OLD.Dipendente) and
      ('16:00:00' <= P.Ora and P.Ora <= '20:00:00' or
       P.Ora <= '20:00:00' and '20:00:00' <= ADDTIME(P.Ora, SEC_TO_TIME(F.Durata*60)) )) = 0) then
      delete
45
      from TurnoDelDipendente
      where GiornoTurno = OLD.Data and FasciaOrariaTurno = fasciaOraria and CodiceFiscaleDipendente in
            (select D.CodiceFiscale
              from Dipendente D
              where D.Cinema = OLD.Cinema and D.Tipo = 'Maschera');
50
      end if;

      end if;

      end if;
END$$

60
USE `CatenaCinema`$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`CartaDiCredito_BEFORE_INSERT` $$%
USE `CatenaCinema`$$%
CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`CartaDiCredito_BEFORE_INSERT` BEFORE INSERT ON `CartaDiCredito`%
FOR EACH ROW
BEGIN
70
#verifico che la data sia un numero compreso tra 1 e 12,
#che l'anno sia di 4 cifre
#il CVV di 3 cifre
#e il numero della carta di 16 cifre
80
if (LENGTH(NEW.Numero) != 16) then
    signal sqlstate '45021'
    set message_text = 'Numero carta non valido';
75

```

```

end if;

if ( 13 <= NEW.MeseScadenza or NEW.MeseScadenza <= 0) then
    signal sqlstate '45019'
    set message_text = 'Mese di scadenza non valido';
5 end if;

if ( 0 >= NEW.AnnoScadenza or NEW.AnnoScadenza >= 100) then
    signal sqlstate '45022'
10    set message_text = 'Anno di scadenza non valido';
end if;

if ( 0 >= NEW.CVV or NEW.CVV >= 1000) then
    signal sqlstate '45020'
15    set message_text = 'CVV non valido';
end if;

END$$

20 USE `CatenaCinema`$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`CartaDiCredito_BEFORE_UPDATE` $$

25 CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`CartaDiCredito_BEFORE_UPDATE` BEFORE UPDATE ON
`CartaDiCredito` FOR EACH ROW
BEGIN
    #verifico che la data sia un numero compreso tra 1 e 12,
    #che l'anno sia di 4 cifre
    #il CVV di 3 cifre
30    #e il numero della carta di 16 cifre

    if (LENGTH(NEW.Numero) != 16) then
        signal sqlstate '45021'
        set message_text = 'Numero carta non valido';
35    end if;

    if ( 13 <= NEW.MeseScadenza or NEW.MeseScadenza <= 0) then
        signal sqlstate '45019'
        set message_text = 'Mese di scadenza non valido';
40    end if;

    if ( 0 >= NEW.AnnoScadenza or NEW.AnnoScadenza >= 10000) then
        signal sqlstate '45022'
        set message_text = 'Anno di scadenza non valido';
45    end if;

    if ( 0 >= NEW.CVV or NEW.CVV >= 1000) then
        signal sqlstate '45020'
        set message_text = 'CVV non valido';
50    end if;
END$$

55 USE `CatenaCinema`$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`Biglietto_AFTER_INSERT` $$

56 CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Biglietto_AFTER_INSERT` AFTER INSERT ON `Biglietto` FOR EACH
ROW
BEGIN
    #dopo che è stato inserito un biglietto
    #aggiorno l'incasso totale della relativa proiezione

    declare costo int;
    declare incasso int;
65
    select P.Costo, P.IncassoTotale
    from Proiezione P
        where P.Cinema = NEW.CinemaDellaProiezione and P.Sala = NEW.SalaDellaProiezione and P.Ora = NEW.OraProiezione
              and P.Film = NEW.Film and P.Data = NEW.DataProiezione
70    into costo, incasso;

    if incasso is null then
        set incasso = 0;
    end if;
75

```

```

set incasso = incasso + costo;
update Proiezione
set IncassoTotale = incasso
5 where Cinema = NEW.CinemaDellaProiezione and Sala = NEW.SalaDellaProiezione and Ora = NEW.OraProiezione
and Film = NEW.Film and Data = NEW.DataProiezione;
END$$

10 USE `CatenaCinema`$$
DROP TRIGGER IF EXISTS `CatenaCinema`.`Biglietto_AFTER_UPDATE` $$%
USE `CatenaCinema`$$
CREATE DEFINER = CURRENT_USER TRIGGER `CatenaCinema`.`Biglietto_AFTER_UPDATE` AFTER UPDATE ON `Biglietto` FOR EACH
15 ROW
BEGIN
    #se e' stato aggiornato il suo tipo in 'Annullato' devo decrementare il costo dall'incasso totale
    declare costo int;
    declare incasso int;
20 if(NEW.Tipo = 'Annullato') then
    select P.Costo, P.IncassoTotale
    from Proiezione P
    where P.Cinema = NEW.CinemaDellaProiezione and P.Sala = NEW.SalaDellaProiezione and P.Ora = NEW.OraProiezione
25 and P.Film = NEW.Film and P.Data = NEW.DataProiezione
    into costo, incasso;

    set incasso = incasso - costo;

30     update Proiezione
    set IncassoTotale = incasso
    where Cinema = NEW.CinemaDellaProiezione and Sala = NEW.SalaDellaProiezione and Ora = NEW.OraProiezione
        and Film = NEW.Film and Data = NEW.DataProiezione;
    end if;
35 END$$

DELIMITER ;

40 delimiter !
create function `cartaEsistente`(numero varchar(16))
returns bool
begin
    if ((select count(*) from CartaDiCredito C where C.Numero = numero) > 0) then
45        return true;
    end if;
    return false;
end!
delimiter ;

50 call popolaSalePosti('0');
source dipendenteDump.dump;
source attoriDump.dump;
source filmDump.dump;
55 call popolaAttoriNelFilm();
call popolaProiezione('1', '0', '4');
call popolaBiglietti('1', '0', '6');


```

## Codice del Front-End

```

60 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql.h>
#include <time.h>
65 #include "program.h"

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
70 #include <unistd.h>
#include <limits.h>

```

```

5      #define MAX_TIME_ACQUISTA 10      //tempo in minuti
      #define fflush(stdin) while(!getchar());

      struct configuration conf;

      static void test_error(MYSQL * con, int status)
10     {
11         if (status) {
12             fprintf(stderr, "Error: %s (errno: %d)\n", mysql_error(con),
13                     mysql_errno(con));
14             execl("./applicazione", "./applicazione", NULL);
15             printf("Errore in execl\n");
16         }
17     }

20     static void test_stmt_error(MYSQL_STMT * stmt, int status)
21     {
22         if (status) {
23             fprintf(stderr, "Error: %s (errno: %d)\n",
24                     mysql_stmt_error(stmt), mysql_stmt_erro(stmt));
25             execl("./applicazione", "./applicazione", NULL);
26             printf("Errore in execl\n");
27         }
28     }

30     char *scegli = "Scegli tra le seguenti opzioni\n";
31     char *sceltaScorretta = "Scelta non disponibile, ritenta\n";
32     char *inserisciScelta = "Inserisci scelta: ";
33     const MYSQL const *con;
34

35     void cancella_db()
36     {
37
38         if (mysql_real_connect(con, conf.host, conf.username, conf.password, conf.database, conf.port, NULL, 0) == NULL) {
39             fprintf(stderr, "Connection error: %s\n", mysql_error(con));
40             exit(1);
41         }

42         int status;
43         MYSQL_STMT *stmt;

44         stmt = mysql_stmt_init(con);
45         if (!stmt) {
46             printf("Could not initialize statement\n");
47             exit(1);
48         }

49         status = mysql_stmt_prepare(stmt, "drop schema CatenaCinema", 25);
50         test_stmt_error(stmt, status);

51         status = mysql_stmt_execute(stmt);
52         test_stmt_error(stmt, status);

53         mysql_stmt_close(stmt);
54         mysql_close(con);
55     }

56     void visualizzaProiezioni()
57     {
58         int cinema;
59         char nomeFilm[128];
60         char data[11];
61         char ora[9];
62         //char sala[3];
63         int sala;
64         bool avanti = false;
65         while(!avanti){
66
67
68
69
70
71
72
73
74
75
    }
}

```

```

5      puts("Per quale cinema vuoi visualizzare le proiezioni (ID Cinema da 1 a 10) ?\n");
       scanf("%d", &cinema);
       fflush(stdin);
       if(cinema < 1 || cinema > 10){
           puts(sceltaScorretta);
       } else {
           avanti = true;
       }
   }

10     puts("Per quale film vuoi visualizzare le proiezioni disponibili ?\n");
       scanf ("%[^n]", nomeFilm);
       fflush(stdin);

15     printf("Visualizzo info proiezioni su '%s', del cinema %d\n", nomeFilm, cinema);

20     MYSQL_STMT *stmt;
       MYSQL_BIND ps_params[2];
       unsigned long length[2];
       length[0] = strlen(nomeFilm);
       length[1] = sizeof(int);
       int num_fields, status, i;
       int j = 1;

25     MYSQL_RES *rs_metadata;
       MYSQL_FIELD *fields;
       MYSQL_BIND *rs_bind;
       my_bool is_null[3];

30     stmt = mysql_stmt_init(con);
       if (!stmt) {
           printf("Could not initialize statement\n");
           exit(1);
       }

35     status = mysql_stmt_prepare(stmt, "call proiezioniDisponibili(?,?)", 32);
       test_stmt_error(stmt, status);

       memset(ps_params, 0, sizeof(ps_params));

40     ps_params[1].buffer_type = MYSQL_TYPE_LONG;
       ps_params[1].buffer = &cinema;
       ps_params[1].length = &length[1];
       ps_params[1].buffer_length = sizeof(int);
       ps_params[1].is_null = 0;

45     ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
       ps_params[0].buffer = nomeFilm;
       ps_params[0].length = &length[0];
       ps_params[0].is_null = 0;

50     status = mysql_stmt_bind_param(stmt, ps_params);
       test_stmt_error(stmt, status);

55     status = mysql_stmt_execute(stmt);
       test_stmt_error(stmt, status);

60     do {
       num_fields = mysql_stmt_field_count(stmt);
       if (num_fields > 0) {

65         if (num_fields == 0){
           printf("Il film digitato non ha proiezioni nel DB\n");
           break;
         }

         if (con->server_status & SERVER_PS_OUT_PARAMS)
             printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

70         rs_metadata = mysql_stmt_result_metadata(stmt);
         test_stmt_error(stmt, rs_metadata == NULL);

         // Retrieve the fields associated with OUT/INOUT parameters
         fields = mysql_fetch_fields(rs_metadata);
       }
     }

```

```

rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
if (!rs_bind) {
    printf("Cannot allocate output buffers\n");
    exit(1);
}
5      memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[0].is_null = &is_null[0];
rs_bind[0].buffer = data;
rs_bind[0].buffer_length = sizeof(data);

10     rs_bind[1].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[1].is_null = &is_null[1];
rs_bind[1].buffer = ora;
rs_bind[1].buffer_length = sizeof(ora);

15     rs_bind[2].buffer_type = MYSQL_TYPE_LONG;
rs_bind[2].is_null = &is_null[2];
rs_bind[2].buffer = &sala;
rs_bind[2].buffer_length = sizeof(int);

20     status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);

25     while(1){
        status = mysql_stmt_fetch(stmt);

        if(status == 1 || status == MYSQL_NO_DATA){
            break;
        }else if(!*rs_bind[0].is_null){
            30         printf("\tProiezione disponibile numero %d\n", j);
            printf("Data: %s\n", rs_bind[0].buffer);
            printf("Ora: %s\n", rs_bind[1].buffer);
            printf("Sala: %d\n", sala);
            printf("\t-----\n");
        }
        j++;
    }

40     mysql_free_result(rs_metadata);
free(rs_bind);
} else {
    45     printf("End of procedure output\n");
}

status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);

50 } while (status == 0);

mysql_stmt_close(stmt);
}

55 void visualizzaDettagliFilm()
{
    60     char nomeFilm[128];
    char durata[4];
    char casaCin[20];

    puts("Inserisci il nome del film di cui vuoi sapere i dettagli:\n");
    scanf ("%[^n]", nomeFilm);
    fflush(stdin);
    65     printf("Film: %s\n", nomeFilm);

    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[1];
    unsigned long length[1];
    length[0] = strlen(nomeFilm);
    int num_fields, status, i;

    70     MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;

```

```

my_bool is_null[2];

stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}

status = mysql_stmt_prepare(stmt, "select Durata, CasaCinematografica from Film where Nome = ?", 60);
10 test_stmt_error(stmt, status);

memset(ps_params, 0, sizeof(ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
15 ps_params[0].buffer = nomeFilm;
ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
20 test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

25 do {
    num_fields = mysql_stmt_field_count(stmt);

    if (num_fields > 0) {

        30 if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

        rs_metadata = mysql_stmt_result_metadata(stmt);
        test_stmt_error(stmt, rs_metadata == NULL);

        35 fields = mysql_fetch_fields(rs_metadata);
        rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
        }
        40 memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

        rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = durata;
        rs_bind[0].buffer_length = sizeof(durata);

        rs_bind[1].buffer_type = fields[1].type;
        rs_bind[1].is_null = &is_null[1];
        rs_bind[1].buffer = casaCin;
        rs_bind[1].buffer_length = sizeof(casaCin);

        55 status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

        while(1){
            status = mysql_stmt_fetch(stmt);

            60 if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Durata del film: %s\n", durata);
                printf("Casa Cinematografica del film: %s\n", rs_bind[1].buffer);
            }
        }

        65 mysql_free_result(rs_metadata);
        free(rs_bind);
    }
    70 } else {
        printf("-----\n");
    }
}
75 status = mysql_stmt_next_result(stmt);

```

```

        if (status > 0)
            test_stmt_error(stmt, status);
    } while (status == 0);

5      mysql_stmt_close(stmt);

    }

10     void visualizzaFilm()
{
    char nomeFilm[128];
    MYSQL_STMT *stmt;
    int num_fields, status;
15    int cinema;

    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
20    my_bool is_null[2];

    stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\n");
        exit(1);
25    }

    char *query = "select distinct F.Nome, P.Cinema from Film F join Proiezione P on F.Nome = P.Film";

30    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

35    do {
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {
30
            if (con->server_status & SERVER_PS_OUT_PARAMS)
                printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

            rs_metadata = mysql_stmt_result_metadata(stmt);
            test_stmt_error(stmt, rs_metadata == NULL);

            fields = mysql_fetch_fields(rs_metadata);
            rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
            if (!rs_bind) {
50
                printf("Cannot allocate output buffers\n");
                exit(1);
            }
            memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);
            rs_bind[0].buffer_type = fields[0].type;
            rs_bind[0].is_null = &is_null[0];
            rs_bind[0].buffer = nomeFilm;
            rs_bind[0].buffer_length = sizeof(nomeFilm);

55
            rs_bind[1].buffer_type = fields[1].type;
            rs_bind[1].is_null = &is_null[1];
            rs_bind[1].buffer = &cinema;
            rs_bind[1].buffer_length = sizeof(int);

            status = mysql_stmt_bind_result(stmt, rs_bind);
            test_stmt_error(stmt, status);

60
            while (1) {
                status = mysql_stmt_fetch(stmt);

                if (status == 1 || status == MYSQL_NO_DATA)
                    break;

                if (*rs_bind[0].is_null){
                    printf(" val[%d] = NULL;", 0);
                }else{
70
    
```

```

                printf("Film: %s, Cinema: %d\n", rs_bind[0].buffer, cinema);
            }
        }

5           mysql_free_result(rs_metadata);
        free(rs_bind);
    } else {
        // no columns = final status packet
10      printf("-----\n");
    }

    status = mysql_stmt_next_result(stmt);
    if (status > 0)
        test_stmt_error(stmt, status);
15    } while (status == 0);

    mysql_stmt_close(stmt);

20 }

void castAttori()
{
25   char nomeFilm[128];
   char nome[20];
   char cognome[20];

30   puts("Inserisci il nome del film di cui vuoi sapere il cast di attori protagonisti:\n");
   scanf ("%[^n]", nomeFilm);
   fflush(stdin);
   printf("Film: %s\n", nomeFilm);

35   MYSQL_STMT *stmt;
   MYSQL_BIND ps_params[1];
   unsigned long length[1];
   length[0] = strlen(nomeFilm);
   int num_fields, status, i;
   int j = 1;

40   MYSQL_RES *rs_metadata;
   MYSQL_FIELD *fields;
   MYSQL_BIND *rs_bind;
   my_bool is_null[2];
45   stmt = mysql_stmt_init(con);
   if (!stmt) {
       printf("Could not initialize statement\n");
       exit(1);
   }

50   status = mysql_stmt_prepare(stmt, "select NomeAttore, CognomeAttore from AttoriNelFilm where NomeFilm = ?", 71);
   test_stmt_error(stmt, status);

55   memset(ps_params, 0, sizeof(ps_params));

   ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
   ps_params[0].buffer = nomeFilm;
   ps_params[0].length = &length[0];
60   ps_params[0].is_null = 0;

   status = mysql_stmt_bind_param(stmt, ps_params);
   test_stmt_error(stmt, status);

65   status = mysql_stmt_execute(stmt);
   test_stmt_error(stmt, status);

   do {
        num_fields = mysql_stmt_field_count(stmt);
70     if (num_fields > 0) {

         if (con->server_status & SERVER_PS_OUT_PARAMS)
             printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");
    }
}

```

```

rs_metadata = mysql_stmt_result_metadata(stmt);
test_stmt_error(stmt, rs_metadata == NULL);

5      fields = mysql_fetch_fields(rs_metadata);
rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
if(!rs_bind) {
    printf("Cannot allocate output buffers\n");
    exit(1);
}
10     memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[0].is_null = &is_null[0];
rs_bind[0].buffer = nome;
rs_bind[0].buffer_length = sizeof(nome);

15     rs_bind[1].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[1].is_null = &is_null[1];
rs_bind[1].buffer = cognome;
rs_bind[1].buffer_length = sizeof(cognome);

status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);

20
25     while(1){
        status = mysql_stmt_fetch(stmt);

        if(status == 1 || status == MYSQL_NO_DATA){
            break;
        }else{
            printf("\tAttore protagonista %d\n", j);
            printf("Nome: %s\n", rs_bind[0].buffer);
            printf("Cognome: %s\n", rs_bind[1].buffer);
            j++;
        }
    }

40     mysql_free_result(rs_metadata);
free(rs_bind);

45     } else {
        printf("-----\n");
    }

status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);
} while (status == 0);

50     mysql_stmt_close(stmt);
}

55     char *menu1cli = "Procedura di acquisto del biglietto, dopo aver scelto la proiezione da prenotare (nel menu precedente)\nInserisci i seguenti dati:\n";

void acquistaBiglietto()
{
60     /*
         Seconde le specifiche da questo momento in poi, il cliente ha 10 minuti per completare
         la procedura di acquisto del biglietto.
     */

65     puts(menu1cli);
char film[128];
memset(film, 0, 128);

70     char codice[50];
memset(codice, 0, 50);

char numeroCarta[17];
memset(numeroCarta, 0, 17);
char nomeIntest[45];
memset(nomeIntest, 0, 45);
75

```

```

char cognomeIntest[45];
memset(cognomeIntest, 0, 45);
int meseScad;
5      memset(&meseScad, 0, 4);
int annoScad;
memset(&annoScad, 0, 4);
int cvv;
memset(&cvv, 0, 4);

10     int numeroPosto;
memset(&numeroPosto, 0, 4);
char filaPosto[1];
memset(filaPosto, 0, 1);
int scelta;
15     int cinema;
memset(&cinema, 0, 4);
int sala;
memset(&sala, 0, 4);
MYSQL_TIME data;
20      memset(&data, 0, sizeof(data));
MYSQL_TIME ora;
memset(&ora, 0, sizeof(ora));

data.time_type = MYSQL_TYPE_DATETIME;
25      ora.time_type = MYSQL_TYPE_DATETIME;

double sec;
sec = time(NULL);

30      printf("Cinema: ");
scanf("%d", &cinema);
fflush(stdin);
printf("Sala: ");
scanf("%d", &sala);
fflush(stdin);
printf("Film: ");
scanf("%[^\\n]", film);
fflush(stdin);
printf("Anno: ");
40      scanf("%d", &data.year);
fflush(stdin);
printf("Mese: ");
scanf("%d", &data.month);
fflush(stdin);
printf("Giorno: ");
45      scanf("%d", &data.day);
fflush(stdin);
printf("Ora: ");
scanf("%d", &ora.hour);
fflush(stdin);
printf("Minuti: ");
50      scanf("%d", &ora.minute);
fflush(stdin);
ora.second = 0;
printf("Fila Del Posto Scelto: ");
55      scanf("%s", filaPosto);
fflush(stdin);
printf("Numero Del Posto Scelto: ");
scanf("%d", &numeroPosto);
fflush(stdin);

60      printf("Ora inserisca i dati relativi alla sua carta di credito\n(Tempo rimanente: %.1f minuti)\n", 10-((time(NULL)-sec)/60));

printf("Numero carta: ");
65      scanf("%s", numeroCarta);
fflush(stdin);
printf("Nome intestatario: ");
scanf("%[^\\n]", nomeIntest);
fflush(stdin);
printf("Cognome intestatario: ");
70      scanf("%[^\\n]", cognomeIntest);
fflush(stdin);
printf("Mese scadenza: ");
scanf("%d", &meseScad);
fflush(stdin);

```

```

printf("Anno scadenza: ");
scanf("%d", &annoScad);
fflush(stdin);
printf("Codice CVV: ");
scanf("%d", &cvv);
fflush(stdin);

5      sec = time(NULL) - sec;
double tempo_impiegato = sec/60;

sec = time(NULL) - sec;
double tempo_impiegato = sec/60;

10     //controllo che il tempo impiegato sia <= a MAX_TIME_ACQUISTA

if(tempo_impiegato >= MAX_TIME_ACQUISTA){
    system("clear");
15     printf("Ci hai messo troppo tempo, riprova per favore.\n");
    return;
}

20     MYSQL_RES *rs_metadata;
MYSQL_FIELD *fields;
MYSQL_BIND *rs_bind;
my_bool is_null[1];

MYSQL_STMT *stmt;
25     MYSQL_BIND ps_params[14];
unsigned long length[14];
length[7] = strlen(numeroCarta);
length[8] = strlen(nomeIntest);
length[9] = strlen(cognomeIntest);
30     length[10] = sizeof(int);
length[11] = sizeof(int);
length[12] = sizeof(int);
length[2] = sizeof(data);
length[3] = sizeof(ora);
length[4] = strlen(film);
length[5] = sizeof(int);
length[6] = strlen(filaPosto);
length[0] = sizeof(int);
length[1] = sizeof(int);
40     int num_fields, status, i;

stmt = mysql_stmt_init(con);
if (!stmt) {
    if (!stmt) {
        printf("Could not initialize statement\n");
45         exit(1);
    }
}

char *query = "call acquistaBiglietto(?,?,?,?,?,?,?,?,?,?)";
status = mysql_stmt_prepare(stmt, query, strlen(query));
50     test_stmt_error(stmt, status);

memset(ps_params, 0, sizeof(ps_params));

ps_params[1].buffer_type = MYSQL_TYPE_LONG;
55     ps_params[1].buffer = &sala;
ps_params[1].length = &length[1];
ps_params[1].buffer_length = sizeof(int);
ps_params[1].is_null = 0;

ps_params[0].buffer_type = MYSQL_TYPE_LONG;
ps_params[0].buffer = &cinema;
ps_params[0].length = &length[0];
ps_params[0].buffer_length = sizeof(int);
ps_params[0].is_null = 0;

60     ps_params[4].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[4].buffer = film;
ps_params[4].length = &length[4];
ps_params[4].is_null = 0;

ps_params[5].buffer_type = MYSQL_TYPE_LONG;
70     ps_params[5].buffer = &numeroPosto;
ps_params[5].length = &length[5];
ps_params[5].buffer_length = sizeof(int);
ps_params[5].is_null = 0;

```

```

5      ps_params[6].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[6].buffer = filaPosto;
ps_params[6].length = &length[6];
ps_params[6].is_null = 0;

10     ps_params[2].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[2].buffer = (char *)&data;
ps_params[2].length = 0;
ps_params[2].is_null = 0;

15     ps_params[3].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[3].buffer = (char *)&ora;
ps_params[3].length = 0;
ps_params[3].is_null = 0;

20     ps_params[7].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[7].buffer = numeroCarta;
ps_params[7].length = &length[7];
ps_params[7].is_null = 0;

25     ps_params[8].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[8].buffer = nomeIntest;
ps_params[8].length = &length[8];
ps_params[8].is_null = 0;

30     ps_params[9].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[9].buffer = cognomeIntest;
ps_params[9].length = &length[9];
ps_params[9].is_null = 0;

35     ps_params[10].buffer_type = MYSQL_TYPE_LONG;
ps_params[10].buffer = &meseScad;
ps_params[10].length = &length[10];
ps_params[10].buffer_length = sizeof(int);
ps_params[10].is_null = 0;

40     ps_params[11].buffer_type = MYSQL_TYPE_LONG;
ps_params[11].buffer = &annoScad;
ps_params[11].length = &length[11];
ps_params[11].buffer_length = sizeof(int);
ps_params[11].is_null = 0;

45     ps_params[12].buffer_type = MYSQL_TYPE_LONG;
ps_params[12].buffer = &ccvv;
ps_params[12].length = &length[12];
ps_params[12].buffer_length = sizeof(int);
ps_params[12].is_null = 0;

50     status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

55     do {
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {

60         if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

65         rs_metadata = mysql_stmt_result_metadata(stmt);
test_stmt_error(stmt, rs_metadata == NULL);

        fields = mysql_fetch_fields(rs_metadata);
        rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
        }
        memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

70         rs_bind[0].buffer_type = fields[0].type;

```

```

        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = codice;
        rs_bind[0].buffer_length = sizeof(codice);

5         status = mysql_stmt_bind_result(stmt, rs_bind);
         test_stmt_error(stmt, status);

        while(1){
            status = mysql_stmt_fetch(stmt);

10           if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf(" -----CODICE DI PRENOTAZIONE----- \n");
                printf("!! --> %s <--- !!\n", codice);
            }
        }

20       mysql_free_result(rs_metadata);
        free(rs_bind);

        } else {
            printf("-----\n");
        }

25       status = mysql_stmt_next_result(stmt);
        if (status > 0)
            test_stmt_error(stmt, status);
    } while (status == 0);

30       mysql_stmt_close(stmt);
        if(strlen(codice) > 0){
            printf("Conserva bene questo codice, se vuoi annullare tale biglietto in futuro,\n");
            printf("Oppure confermarlo all'ingresso del cinema, presentandolo alla maschera.\n");
            printf("Grazie !\n");
50       }else{
            printf("Errore nella procedura di acquisto del biglietto, per favore ricontrolla i dati inseriti.\n");
        }
    }

40   char *menu2cli = "Per vedere i posti disponibili per una certa proiezione, inserisci i seguenti dati:\n";

45   void postiDisponibili()
{
    puts(menu2cli);
    char film[128];
50   memset(film, 0, 128);

    char fila[5];
    memset(film, 0, 5);
    char numero[5];
55   memset(film, 0, 5);
    char disp[5];
    memset(film, 0, 5);

    int scelta;
    int cinema;
60   memset(&cinema, 0, 4);
    int sala;
    memset(&sala, 0, 4);
    MYSQL_TIME data;
65   memset(&data, 0, sizeof(data));
    MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));

    data.time_type = MYSQL_TYPE_DATETIME;
70   ora.time_type = MYSQL_TYPE_DATETIME;

    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
    printf("Sala: ");
}

```

```

scanf("%d", &sala);
fflush(stdin);
printf("Film: ");
scanf("%[^\\n]", film);
5
fflush(stdin);
printf("Anno: ");
scanf("%d", &data.year);
fflush(stdin);
printf("Mese: ");
scanf("%d", &data.month);
fflush(stdin);
printf("Giorno: ");
scanf("%d", &data.day);
fflush(stdin);
10
printf("Ora: ");
scanf("%d", &ora.hour);
fflush(stdin);
printf("Minuti: ");
scanf("%d", &ora.minute);
fflush(stdin);
20
ora.second = 0;

MYSQL_RES *rs_metadata;
MYSQL_FIELD *fields;
25
MYSQL_BIND *rs_bind;
my_bool is_null[3];

MYSQL_STMT *stmt;
MYSQL_BIND ps_params[5];
30
unsigned long length[5];
length[0] = sizeof(data);
length[1] = sizeof(ora);
length[2] = strlen(film);
length[3] = sizeof(int);
length[4] = sizeof(int);
35
int num_fields, status, i;

stmt = mysql_stmt_init(con);
if (!stmt) {
40
    printf("Could not initialize statement\\n");
    exit(1);
}

char *query = "call postiDisponibiliDellaProiezione(?,?,?,?,?)";
45
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

memset(ps_params, 0, sizeof(ps_params));

50
ps_params[3].buffer_type = MYSQL_TYPE_LONG;
ps_params[3].buffer = &sala;
ps_params[3].length = &length[3];
ps_params[3].buffer_length = sizeof(int);
ps_params[3].is_null = 0;

55
ps_params[4].buffer_type = MYSQL_TYPE_LONG;
ps_params[4].buffer = &cinema;
ps_params[4].length = &length[4];
ps_params[4].buffer_length = sizeof(int);
ps_params[4].is_null = 0;

60
ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[2].buffer = film;
ps_params[2].length = &length[2];
ps_params[2].is_null = 0;

65
ps_params[0].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[0].buffer = (char *)&data;
ps_params[0].length = 0;
ps_params[0].is_null = 0;

70
ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[1].buffer = (char *)&ora;
ps_params[1].length = 0;
ps_params[1].is_null = 0;

```

```

status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

5      status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

do {
    num_fields = mysql_stmt_field_count(stmt);
10     if (num_fields > 0) {

        if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

15     rs_metadata = mysql_stmt_result_metadata(stmt);
test_stmt_error(stmt, rs_metadata == NULL);

20     fields = mysql_fetch_fields(rs_metadata);
rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
if(!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
}
25     memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[0].is_null = &is_null[0];
rs_bind[0].buffer = fila;
rs_bind[0].buffer_length = sizeof(fila);

30     rs_bind[1].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[1].is_null = &is_null[1];
rs_bind[1].buffer = numero;
rs_bind[1].buffer_length = sizeof(numero);

rs_bind[2].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[2].is_null = &is_null[0];
rs_bind[2].buffer = disp;
rs_bind[2].buffer_length = sizeof(disp);

40     status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);

45     while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MySQL_NO_DATA){
                break;
}
50     }else{
            if(strcmp(disp, "0") == 0)
                printf("Fila Posto: %s, Numero Posto: %s\n", fila, numero);
}
55     mysql_free_result(rs_metadata);
free(rs_bind);

} else {
58     printf("-----\n");
}
60     status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);

65 } while (status == 0);

mysql_stmt_close(stmt);
}

70 char *menu0cli = "1)Visualizza film\n2)Visualizza dettagli film\n3)Visualizza il cast di attori protagonisti di un film\n4)Visualizza proiezioni disponibili di un cinema\n5)Visualizza posti disponibili di una proiezione\n6)Acquista biglietto\n7)Annulla acquisto biglietto\n8)Indietro";
75 void annullaAcquisto(){

```

```

char codice[45];
memset(codice, 0, 45);

5      puts("Inserisci il codice di prenotazione del biglietto acquistato:\n");
scanf("%s", codice);

10     MYSQL_STMT *stmt;
MYSQL_BIND ps_params[1];
unsigned long length[1];
length[0] = strlen(codice);
int status;

15     stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}

20     char *query = "call annullaBiglietto(?);"

status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

25     memset(ps_params, 0, sizeof(ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[0].buffer = codice;
ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

30     status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

35     mysql_stmt_close(stmt);

40     printf("Biglietto annullato correttamente\n");
}

void cliente()
45     {

50         if (mysql_real_connect(con, conf.host, conf.username, conf.password, conf.database, conf.port, NULL, 0) == NULL) {
            fprintf(stderr, "Connection error: %s\n", mysql_error(con));
            exit(1);
        }

55         int scelta;

while(1){
56             puts(scegli);
            puts(menu0cli);
            puts(inserisciScelta);
            scanf("%d", &scelta);
            fflush(stdin);
            system("clear");
            switch (scelta) {

60                 case 1:
                    visualizzaFilm();
                    break;
                case 2:
                    visualizzaDettagliFilm();
                    break;
                case 3:
                    castAttori();
                    break;
                case 4:
                    visualizzaProiezioni();
                    break;
                case 5:

```

```

postiDisponibili();
break;
case 6:
acquistaBiglietto();
break;
case 7:
annullaAcquisto();
break;
case 8:
mysql_close(con);
return;
default:
puts(sceltaScorretta);
break;
}
}
}

20 char *menu0vis = "1)Visualizza film\n2)Visualizza dettagli film\n3)Visualizza il cast di attori protagonisti di un film\n4)Visualizza proiezioni
disponibili di un cinema\n5)Indietro";

25 void visitatore()
{
    if(mysql_real_connect(con, conf.host, conf.username, conf.password, conf.database, conf.port, NULL, 0) == NULL) {
        fprintf(stderr, "Connection error: %s\n", mysql_error(con));
        exit(1);
    }

30     int scelta;

35     while(1){
        puts(scegli);
        puts(menu0vis);
        puts(inserisciScelta);
        scanf("%d", &scelta);
        fflush(stdin);
        system("clear");
        switch (scelta) {

40             case 1:
                visualizzaFilm();
                break;
            case 2:
                visualizzaDettagliFilm();
                break;
            case 3:
                castAttori();
                break;
            case 4:
                visualizzaProiezioni();
                break;
            case 5:
                mysql_close(con);
                return;
            default:
                puts(sceltaScorretta);
                break;
        }
    }
}

45

50

55

60

65 void confermaBiglietto()
{
    char codice[45];
    memset(codice, 0, 45);

70     puts("Inserisci il codice di prenotazione del biglietto acquistato:\n");
    scanf ("%s", codice);

    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[1];
}

```

```

unsigned long length[1];
length[0] = strlen(codice);
int status;

5      stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}

10     char *query = "call confermaBiglietto(?)";

status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

15     memset(ps_params, 0, sizeof(ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[0].buffer = codice;
20     ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

25     status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

mysql_stmt_close(stmt);

30     printf("Biglietto confermato correttamente\n");
}

35     char *menu0mas = "1)Conferma biglietto\n2)Indietro\n";

void maschera()
{
40     if (mysql_real_connect(con, conf.host, conf.username, conf.password, conf.database, conf.port, NULL, 0) == NULL) {
        fprintf(stderr, "Connection error: %s\n", mysql_error(con));
        exit(1);
    }

45     int scelta;

while(1){
    puts(scegli);
    puts(menu0mas);
    puts(inserisciScelta);
50     scanf("%d", &scelta);
    fflush(stdin);
    system("clear");
    switch (scelta) {
55         case 1:
            confermaBiglietto();
            break;
        case 2:
            mysql_close(con);
            return;
        default:
            puts(sceltaScorretta);
            break;
60     }
65 }

70 void mostraProiezionistaDiProiez()
{
    int cinema;
    memset(&cinema, 0, 4);
}

```

```

int sala;
memset(&sala, 0, 4);
char film[128];
memset(film, 0, 128);
5      char proiezionista[45];
memset(proiezionista, 0, 45);
MYSQL_TIME data;
MYSQL_TIME ora;
memset(&ora, 0, sizeof(ora));
10     memset(&data, 0, sizeof(data));
ora.time_type = MYSQL_TYPE_DATETIME;
data.time_type = MYSQL_TYPE_DATETIME;

printf("Inserisci i seguenti dati per individuare il proiezionista della proiezione:\n");
15     printf("Cinema: ");
scanf("%d", &cinema);
fflush(stdin);
printf("Sala: ");
scanf("%d", &sala);
20     fflush(stdin);
printf("Film: ");
scanf("%[^\\n]", film);
fflush(stdin);
printf("Anno: ");
scanf("%d", &data.year);
25     fflush(stdin);
printf("Mese: ");
scanf("%d", &data.month);
fflush(stdin);
printf("Giorno: ");
30      scanf("%d", &data.day);
fflush(stdin);
printf("Ora: ");
scanf("%d", &ora.hour);
fflush(stdin);
printf("Minuti: ");
35      scanf("%d", &ora.minute);
fflush(stdin);
ora.second = 0;

40     MYSQL_STMT *stmt;
MYSQL_BIND ps_params[5];
unsigned long length[5];
length[0] = strlen(film);
45     length[1] = sizeof(data);
length[2] = sizeof(ora);
length[3] = sizeof(int);
length[4] = sizeof(int);
int num_fields, status, i;

50     MYSQL_RES *rs_metadata;
MYSQL_FIELD *fields;
MYSQL_BIND *rs_bind;
my_bool is_null[1];

55     stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}
60

char *query = "select Dipendente from Proiezione where Film = ? and Data = ? and Ora = ? and Cinema = ? and Sala = ?";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

65     memset(ps_params, 0, sizeof(ps_params));

ps_params[3].buffer_type = MYSQL_TYPE_LONG;
ps_params[3].buffer = &cinema;
70     ps_params[3].length = &length[3];
ps_params[3].buffer_length = sizeof(int);
ps_params[3].is_null = 0;

ps_params[4].buffer_type = MYSQL_TYPE_LONG;
ps_params[4].buffer = &sala;
75

```

```

ps_params[4].length = &length[4];
ps_params[4].buffer_length = sizeof(int);
ps_params[4].is_null = 0;

5      ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[0].buffer = film;
ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

10     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[1].buffer = (char *)&data;
ps_params[1].length = 0;
ps_params[1].is_null = 0;

15     ps_params[2].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[2].buffer = (char *)&ora;
ps_params[2].length = 0;
ps_params[2].is_null = 0;

20     status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

25     do {
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {
30         if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

            rs_metadata = mysql_stmt_result_metadata(stmt);
            test_stmt_error(stmt, rs_metadata == NULL);

35         fields = mysql_fetch_fields(rs_metadata);
rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
}
memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

40         rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[0].is_null = &is_null[0];
rs_bind[0].buffer = proiezionista;
rs_bind[0].buffer_length = sizeof(proiezionista);

45         status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);

50         while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Proiezionista che gestisce la proiezione inserita: %s\n", rs_bind[0].buffer);
}
55         }

60         mysql_free_result(rs_metadata);
free(rs_bind);

65         } else {
printf("-----\n");
}

70         status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);
} while (status == 0);

75     mysql_stmt_close(stmt);
}

```

```

void mostraMaschere()
{
    5      char *pom = "16-20";
    char *ser = "20-24";
    char maschera[45];
    memset(maschera, 0, 45);
    int scelta;
    10     int cinema;
    memset(&cinema, 0, 4);
    char fasciaOraria[5];
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));

    15     printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);

    20     puts("Fascia oraria:\n1)16-20\n2)20-24\n");
    scanf("%d", &scelta);
    fflush(stdin);

    25     switch (scelta){
        case 1:
            strcpy(fasciaOraria, pom);
            break;
        case 2:
            strcpy(fasciaOraria, ser);
            break;
        30     default:
            printf("Scelta scorretta\n");
            return;
    }

    35 }

    data.time_type = MYSQL_TYPE_DATETIME;
    printf("Anno: ");
    40     scanf("%d", &data.year);
    fflush(stdin);
    printf("Mese: ");
    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    45     scanf("%d", &data.day);
    fflush(stdin);

    MySQL_STMT *stmt;
    MySQL_BIND ps_params[3];
    50     unsigned long length[3];
    length[0] = sizeof(int);
    length[1] = strlen(fasciaOraria);
    length[2] = sizeof(data);
    int num_fields, status, i;

    55     MySQL_RES *rs_metadata;
    MySQL_FIELD *fields;
    MySQL_BIND *rs_bind;
    my_bool is_null[1];
    60     stmt = mysql_stmt_init(con);
    if (!stmt) {
        if (!stmt)
            printf("Could not initialize statement\n");
            exit(1);
    }

    65     char *query = "select D.CodiceFiscale from TurnoDelDipendente TD join Dipendente D on TD.CodiceFiscaleDipendente =
D.CodiceFiscale where D.Cinema = ? and TD.FasciaOrariaTurno = ? and TD.GiornoTurno = ? and Tipo = 'Maschera'";

    70     status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    memset(ps_params, 0, sizeof(ps_params));
    75     ps_params[0].buffer_type = MYSQL_TYPE_LONG;

```

```

    ps_params[0].buffer = &cinema;
    ps_params[0].length = &length[0];
    ps_params[0].buffer_length = sizeof(int);
    ps_params[0].is_null = 0;

5      ps_params[1].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[1].buffer = fasciaOraria;
    ps_params[1].length = &length[1];
    ps_params[1].is_null = 0;

10     ps_params[2].buffer_type = MYSQL_TYPE_DATETIME;
    ps_params[2].buffer = (char *)&data;
    ps_params[2].length = 0;
    ps_params[2].is_null = 0;

15     status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

20     do {
        num_fields = mysql_stmt_field_count(stmt);

25         if (num_fields > 0) {

            if (con->server_status & SERVER_PS_OUT_PARAMS)
                printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

30             rs_metadata = mysql_stmt_result_metadata(stmt);
test_stmt_error(stmt, rs_metadata == NULL);

            fields = mysql_fetch_fields(rs_metadata);
            rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
if(!rs_bind) {
                printf("Cannot allocate output buffers\n");
                exit(1);
}
            memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

40             rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
            rs_bind[0].is_null = &is_null[0];
            rs_bind[0].buffer = maschera;
            rs_bind[0].buffer_length = sizeof(maschera);

45             status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);

            while(1){
                status = mysql_stmt_fetch(stmt);

50                 if(status == 1 || status == MYSQL_NO_DATA){
                    break;
                }else{
                    printf("Maschera in turno: %s\n", rs_bind[0].buffer);
}
            }

55             mysql_free_result(rs_metadata);
free(rs_bind);

60             } else {
printf("-----\n");
}

65             status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);
} while (status == 0);

70             mysql_stmt_close(stmt);
}

void aggiornaProiezionista()
{
    char film[128];

```

```

    memset(film, 0, 128);
    char proiezionista[45];
    memset(proiezionista, 0, 45);
    int scelta;
    int cinema;
    memset(&cinema, 0, 4);
    int sala;
    memset(&sala, 0, 4);
    MYSQL_TIME data;
    10   memset(&data, 0, sizeof(data));
    MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));

    data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

    15   printf("Proiezionista: ");
    scanf("%s", proiezionista);
    fflush(stdin);
    printf("Cinema: ");
    20   scanf("%d", &cinema);
    fflush(stdin);
    printf("Sala: ");
    scanf("%d", &sala);
    fflush(stdin);
    puts("Film: ");
    25   scanf("%[^\\n]", film);
    fflush(stdin);
    printf("Anno: ");
    30   scanf("%d", &data.year);
    fflush(stdin);
    printf("Mese: ");
    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    35   scanf("%d", &data.day);
    fflush(stdin);
    printf("Ora: ");
    scanf("%d", &ora.hour);
    fflush(stdin);
    printf("Minuti: ");
    40   scanf("%d", &ora.minute);
    fflush(stdin);
    ora.second = 0;

    45   MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[6];
    unsigned long length[6];
    50   length[0] = strlen(proiezionista);
    length[1] = sizeof(data);
    length[2] = sizeof(ora);
    length[3] = strlen(film);
    length[4] = sizeof(int);
    length[5] = sizeof(int);
    55   int num_fields, status, i;

    stmt = mysql_stmt_init(con);
    if (!stmt) {
        60     printf("Could not initialize statement\\n");
        exit(1);
    }

    65   char *query = "call aggiornaProiezionistaDellaProiezione(?,?,?,?,?,?)";
    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    memset(ps_params, 0, sizeof(ps_params));

    70   ps_params[4].buffer_type = MYSQL_TYPE_LONG;
    ps_params[4].buffer = &sala;
    ps_params[4].length = &length[4];
    ps_params[4].buffer_length = sizeof(int);
    ps_params[4].is_null = 0;

    ps_params[5].buffer_type = MYSQL_TYPE_LONG;

```

```

    ps_params[5].buffer = &cinema;
    ps_params[5].length = &length[5];
    ps_params[5].buffer_length = sizeof(int);
    ps_params[5].is_null = 0;

5      ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[0].buffer = proiezionista;
    ps_params[0].length = &length[0];
    ps_params[0].is_null = 0;

10     ps_params[3].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[3].buffer = film;
    ps_params[3].length = &length[3];
    ps_params[3].is_null = 0;

15     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
    ps_params[1].buffer = (char *)&data;
    ps_params[1].length = 0;
    ps_params[1].is_null = 0;

20     ps_params[2].buffer_type = MYSQL_TYPE_DATETIME;
    ps_params[2].buffer = (char *)&ora;
    ps_params[2].length = 0;
    ps_params[2].is_null = 0;

25     status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

30     status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

        mysql_stmt_close(stmt);

        printf("Operazione conclusa. Per verificare esito consultare turni del proiezionista aggiunto\n");

35 }

void aggiungiMaschera()
{
40     char *pom = "16-20";
    char *ser = "20-24";
    char maschera[45];
    memset(maschera, 0, 45);
    int scelta;
45     char fasciaOraria[5];
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));

    printf("Maschera: ");
50     scanf("%[^\\n]", maschera);
    fflush(stdin);

    puts("Fascia oraria:\n1)16-20\n2)20-24\n");
    scanf("%d", &scelta);
    fflush(stdin);

55     switch (scelta){
        case 1:
            strcpy(fasciaOraria, pom);
            break;
        case 2:
            strcpy(fasciaOraria, ser);
            break;
        default:
            printf("Scelta scorretta\n");
            return;
    }

60 }

data.time_type = MYSQL_TYPE_DATETIME;
printf("Anno: ");
scanf("%d", &data.year);
fflush(stdin);
printf("Mese: ");
scanf("%d", &data.month);
75

```

```

fflush(stdin);
printf("Giorno: ");
scanf("%d", &data.day);
fflush(stdin);

5      MYSQL_STMT *stmt;
MYSQL_BIND ps_params[3];
unsigned long length[3];
length[2] = strlen(maschera);
length[0] = strlen(fasciaOraria);
length[1] = sizeof(data);
int num_fields, status, i;

10     stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}

15     char *query = "insert into TurnoDelDipendente values (?,?,?)";

status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

20     memset(ps_params, 0, sizeof(ps_params));

ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[2].buffer = maschera;
ps_params[2].length = &length[2];
ps_params[2].is_null = 0;

25     ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[0].buffer = fasciaOraria;
ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

30     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[1].buffer = (char *)&data;
ps_params[1].length = 0;
ps_params[1].is_null = 0;

35     status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

40     status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

45     mysql_stmt_close(stmt);

50     printf("Dipendente inserito\n");
}

55     void rimuoviProiezionistaDaProiezione()
{
    char film[128];
    memset(film, 0, 128);
    char proiezionista[45];
    memset(proiezionista, 0, 45);
    int scelta;
    int cinema;
    memset(&cinema, 0, 4);
    int sala;
    memset(&sala, 0, 4);
    MySQL_TIME data;
    memset(&data, 0, sizeof(data));
    MySQL_TIME ora;
    memset(&ora, 0, sizeof(ora));

60     data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

65     printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);

70     data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

75     printf("Sala: ");
    scanf("%d", &sala);
    fflush(stdin);
}

```

```

printf("Sala: ");
scanf("%d", &sala);
fflush(stdin);
puts("Film: ");
scanf("%[^\\n]", film);
fflush(stdin);
printf("Anno: ");
scanf("%d", &data.year);
fflush(stdin);
printf("Mese: ");
scanf("%d", &data.month);
fflush(stdin);
printf("Giorno: ");
scanf("%d", &data.day);
fflush(stdin);
printf("Ora: ");
scanf("%d", &ora.hour);
fflush(stdin);
printf("Minuti: ");
scanf("%d", &ora.minute);
fflush(stdin);
ora.second = 0;

25 MySQL_STMT *stmt;
MySQL_BIND ps_params[5];
unsigned long length[5];
length[0] = sizeof(data);
length[1] = sizeof(ora);
length[2] = strlen(film);
length[3] = sizeof(int);
length[4] = sizeof(int);
int num_fields, status, i;

30 stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\\n");
    exit(1);
}

35 char *query = "update Proiezione set Dipendente = null where Data = ? and Ora = ? and Film = ? and Sala = ? and Cinema = ?";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

40 memset(ps_params, 0, sizeof(ps_params));

45 ps_params[3].buffer_type = MYSQL_TYPE_LONG;
ps_params[3].buffer = &sala;
ps_params[3].length = &length[3];
ps_params[3].buffer_length = sizeof(int);
ps_params[3].is_null = 0;

50 ps_params[4].buffer_type = MYSQL_TYPE_LONG;
ps_params[4].buffer = &cinema;
ps_params[4].length = &length[4];
ps_params[4].buffer_length = sizeof(int);
ps_params[4].is_null = 0;

55 ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[2].buffer = film;
ps_params[2].length = &length[2];
ps_params[2].is_null = 0;

60 ps_params[0].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[0].buffer = (char *)&data;
ps_params[0].length = 0;
ps_params[0].is_null = 0;

65 ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[1].buffer = (char *)&ora;
ps_params[1].length = 0;
ps_params[1].is_null = 0;

70 status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

75

```

```

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

mysql_stmt_close(stmt);

5   printf("Operazione conclusa. Per verificare esito consultare turni del proiezionista rimosso\n");

}

10 void rimuoviMaschera()
{
    char *pom = "16-20";
    char *ser = "20-24";
15    char maschera[45];
    memset(maschera, 0, 45);
    int scelta;
    char fasciaOraria[5];
    MYSQL_TIME data;
20    memset(&data, 0, sizeof(data));

    printf("Maschera: ");
    scanf("%[^\\n]", maschera);
    fflush(stdin);

25    puts("Fascia oraria:\n1)16-20\n2)20-24\n");
    scanf("%d", &scelta);
    fflush(stdin);

30    switch (scelta){
        case 1:
            strcpy(fasciaOraria, pom);
            break;
        case 2:
            strcpy(fasciaOraria, ser);
            break;
35        default:
            printf("Scelta scorretta\n");
            return;
40    }
}

data.time_type = MYSQL_TYPE_DATETIME;
printf("Anno: ");
45    scanf("%d", &data.year);
fflush(stdin);
printf("Mese: ");
scanf("%d", &data.month);
fflush(stdin);
50    printf("Giorno: ");
scanf("%d", &data.day);
fflush(stdin);

55    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[3];
    unsigned long length[3];
    length[2] = strlen(maschera);
    length[0] = strlen(fasciaOraria);
    length[1] = sizeof(data);
    int num_fields, status, i;

    stmt = mysql_stmt_init(con);
    if (!stmt) {
60        printf("Could not initialize statement\n");
        exit(1);
    }

    char *query = "delete from TurnoDelDipendente where FasciaOrariaTurno = ? and GiornoTurno = ? and CodiceFiscaleDipendente = ?";

70    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    memset(ps_params, 0, sizeof(ps_params));
    ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;

```

```

    ps_params[2].buffer = maschera;
    ps_params[2].length = &length[2];
    ps_params[2].is_null = 0;

5      ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[0].buffer = fasciaOraria;
    ps_params[0].length = &length[0];
    ps_params[0].is_null = 0;

10     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
    ps_params[1].buffer = (char *)&data;
    ps_params[1].length = 0;
    ps_params[1].is_null = 0;

15     status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

    status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

20     mysql_stmt_close(stmt);

    printf("Operazione completata, verificare con turno del dipendente\n");
}

25

void proiezionistiDisponibili()
{
    int scelta;
30    int cinema;
    memset(&cinema, 0, 4);
    int durata;
    memset(&durata, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
    MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));
    char proiezionista[45];
    memset(proiezionista, 0, 45);

35    data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

    printf("Cinema: ");
40    scanf("%d", &cinema);
    fflush(stdin);
    printf("Anno: ");
    scanf("%d", &data.year);
    fflush(stdin);
    printf("Mese: ");
50    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
    fflush(stdin);
    printf("Ora: ");
55    scanf("%d", &ora.hour);
    fflush(stdin);
    printf("Minuti: ");
    scanf("%d", &ora.minute);
    fflush(stdin);
    ora.second = 0;
    printf("Minuti per cui deve essere disponibile: ");
60    scanf("%d", &durata);
    fflush(stdin);

65    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[4];
    unsigned long length[4];
    length[2] = sizeof(data);
    length[1] = sizeof(ora);
    length[0] = sizeof(int);
    length[3] = sizeof(int);
    int num_fields, status, i;
    int j = 1;

```

```
5      MYSQL_RES *rs_metadata;
       MYSQL_FIELD *fields;
       MYSQL_BIND *rs_bind;
       my_bool is_null[1];
10     stmt = mysql_stmt_init(con);
     if (!stmt) {
         printf("Could not initialize statement\n");
         exit(1);
     }

15     char *query = "call proiezionistiDisponibili(?, ?, ?, ?)";
     status = mysql_stmt_prepare(stmt, query, strlen(query));
     test_stmt_error(stmt, status);

     memset(ps_params, 0, sizeof(ps_params));

20     ps_params[3].buffer_type = MYSQL_TYPE_LONG;
     ps_params[3].buffer = &durata;
     ps_params[3].length = &length[3];
     ps_params[3].buffer_length = sizeof(int);
     ps_params[3].is_null = 0;

25     ps_params[0].buffer_type = MYSQL_TYPE_LONG;
     ps_params[0].buffer = &cinema;
     ps_params[0].length = &length[0];
     ps_params[0].buffer_length = sizeof(int);
     ps_params[0].is_null = 0;

30     ps_params[2].buffer_type = MYSQL_TYPE_DATETIME;
     ps_params[2].buffer = (char *)&data;
     ps_params[2].length = 0;
     ps_params[2].is_null = 0;

35     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
     ps_params[1].buffer = (char *)&ora;
     ps_params[1].length = 0;
     ps_params[1].is_null = 0;

40     status = mysql_stmt_bind_param(stmt, ps_params);
     test_stmt_error(stmt, status);

45     status = mysql_stmt_execute(stmt);
     test_stmt_error(stmt, status);

50     do {
         num_fields = mysql_stmt_field_count(stmt);

         if (num_fields > 0) {

55             if (con->server_status & SERVER_PS_OUT_PARAMS)
                 printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

             rs_metadata = mysql_stmt_result_metadata(stmt);
             test_stmt_error(stmt, rs_metadata == NULL);

             fields = mysql_fetch_fields(rs_metadata);
             rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
             if (!rs_bind) {
                 printf("Cannot allocate output buffers\n");
                 exit(1);
             }
             memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);
             rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
             rs_bind[0].is_null = &is_null[0];
             rs_bind[0].buffer = proiezionista;
             rs_bind[0].buffer_length = sizeof(proiezionista);

65             status = mysql_stmt_bind_result(stmt, rs_bind);
             test_stmt_error(stmt, status);

             while(1){
                 status = mysql_stmt_fetch(stmt);

70             }
         }
     }

```

```

        if(status == 1 || status == MYSQL_NO_DATA){
            break;
        }else{
            printf("Proiezionista libero n.%d: %s\n", j, rs_bind[0].buffer);
        }
        j++;
    }

    mysql_free_result(rs_metadata);
    free(rs_bind);
} else {
    printf("-----\n");
}

status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);
} while (status == 0);

mysql_stmt_close(stmt);

}

25 void saleDisponibili()
{
    int cinema;
    memset(&cinema, 0, 4);
    int durata;
    memset(&durata, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
    MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));
    int sala;
    memset(&sala, 0, 4);

    data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

40    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
    printf("Anno: ");
    scanf("%d", &data.year);
    fflush(stdin);
    printf("Mese: ");
    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
    fflush(stdin);
    printf("Ora: ");
    scanf("%d", &ora.hour);
    fflush(stdin);
    printf("Minuti: ");
    scanf("%d", &ora.minute);
    fflush(stdin);
    ora.second = 0;
    printf("Minuti per cui deve essere disponibile: ");
    scanf("%d", &durata);
    fflush(stdin);

55    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[4];
    unsigned long length[4];
    length[0] = sizeof(int);
    length[1] = sizeof(ora);
    length[2] = sizeof(data);
    length[3] = sizeof(int);
    int num_fields, status, i;
    int j = 1;

    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
}

```

```

MYSQL_BIND *rs_bind;
my_bool is_null[1];

stmt = mysql_stmt_init(con);
5   if (!stmt) {
      printf("Could not initialize statement\n");
      exit(1);
}

10  char *query = "call saleDisponibili(?,?,?,?,?)";

status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

15  memset(ps_params, 0, sizeof(ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_LONG;
ps_params[0].buffer = &cinema;
ps_params[0].length = &length[0];
ps_params[0].buffer_length = sizeof(int);
20   ps_params[0].is_null = 0;

ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[1].buffer = (char *)&ora;
25   ps_params[1].length = 0;
ps_params[1].is_null = 0;

ps_params[2].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[2].buffer = (char *)&data;
30   ps_params[2].length = 0;
ps_params[2].is_null = 0;

ps_params[3].buffer_type = MYSQL_TYPE_LONG;
ps_params[3].buffer = &durata;
35   ps_params[3].length = &length[3];
ps_params[3].buffer_length = sizeof(int);
ps_params[3].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
40   test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

45  do {
      num_fields = mysql_stmt_field_count(stmt);
      if (num_fields > 0) {

50      if (con->server_status & SERVER_PS_OUT_PARAMS)
          printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

      rs_metadata = mysql_stmt_result_metadata(stmt);
      test_stmt_error(stmt, rs_metadata == NULL);

55      fields = mysql_fetch_fields(rs_metadata);
      rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
      if (!rs_bind) {
          printf("Cannot allocate output buffers\n");
60          exit(1);
      }
      memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);
      rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
      rs_bind[0].is_null = &is_null[0];
      rs_bind[0].buffer = &sala;
65      rs_bind[0].buffer_length = sizeof(int);

      status = mysql_stmt_bind_result(stmt, rs_bind);
      test_stmt_error(stmt, status);

70      while(1){
          status = mysql_stmt_fetch(stmt);
          if(status == 1 || status == MYSQL_NO_DATA){
              break;
    
```

```

                }else{
                    printf("Sala libera n.%d: %d\n", j, sala);
                }
                j++;
            }

            mysql_free_result(rs_metadata);
            free(rs_bind);
        } else {
            printf("-----\n");
        }

        status = mysql_stmt_next_result(stmt);
        if (status > 0)
            test_stmt_error(stmt, status);
    } while (status == 0);

    mysql_stmt_close(stmt);

20   }

void aggiungiProiezione()
{
    int cinema;
25   memset(&cinema, 0, 4);
    int sala;
    memset(&sala, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
30   MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));
    char film[128];
    memset(film, 0, 128);
    char proiezionista[45];
35   memset(proiezionista, 0, 45);
    int costo;
    memset(&costo, 0, 4);

    data.time_type = MYSQL_TYPE_DATETIME;
40   ora.time_type = MYSQL_TYPE_DATETIME;

    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
45   printf("Anno: ");
    scanf("%d", &data.year);
    fflush(stdin);
    printf("Mese: ");
    scanf("%d", &data.month);
50   fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
    fflush(stdin);
    printf("Ora: ");
55   scanf("%d", &ora.hour);
    fflush(stdin);
    printf("Minuti: ");
    scanf("%d", &ora.minute);
    fflush(stdin);
60   ora.second = 0;
    printf("Sala: ");
    scanf("%d", &sala);
    fflush(stdin);
    printf("Costo: ");
65   scanf("%d", &costo);
    fflush(stdin);
    printf("Film: ");
    scanf("%[^\\n]", film);
    fflush(stdin);
70   printf("Proiezionista: ");
    scanf("%s", proiezionista);
    fflush(stdin);

75   MYSQL_STMT *stmt;
}

```

```

5      MYSQL_BIND ps_params[7];
       unsigned long length[7];
       length[3] = sizeof(int);
       length[1] = sizeof(ora);
       length[2] = strlen(film);
       length[0] = sizeof(data);
       length[4] = sizeof(int);
       length[5] = strlen(proiezionista);
       length[6] = sizeof(int);
10     int status;

15     stmt = mysql_stmt_init(con);
       if (!stmt) {
           printf("Could not initialize statement\n");
           exit(1);
       }

20     char *query = "call aggiungiProiezione(?,?,?,?,?,?)";
       status = mysql_stmt_prepare(stmt, query, strlen(query));
       test_stmt_error(stmt, status);

25     memset(ps_params, 0, sizeof(ps_params));

30     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
       ps_params[1].buffer = (char *)&ora;
       ps_params[1].length = 0;
       ps_params[1].is_null = 0;

35     ps_params[0].buffer_type = MYSQL_TYPE_DATETIME;
       ps_params[0].buffer = (char *)&data;
       ps_params[0].length = 0;
       ps_params[0].is_null = 0;

40     ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
       ps_params[2].buffer = film;
       ps_params[2].length = &length[2];
       ps_params[2].is_null = 0;

45     ps_params[3].buffer_type = MYSQL_TYPE_LONG;
       ps_params[3].buffer = &sala;
       ps_params[3].length = &length[3];
       ps_params[3].buffer_length = sizeof(int);
       ps_params[3].is_null = 0;

50     ps_params[4].buffer_type = MYSQL_TYPE_LONG;
       ps_params[4].buffer = &cinema;
       ps_params[4].length = &length[4];
       ps_params[4].buffer_length = sizeof(int);
       ps_params[4].is_null = 0;

55     ps_params[5].buffer_type = MYSQL_TYPE_VAR_STRING;
       ps_params[5].buffer = proiezionista;
       ps_params[5].length = &length[5];
       ps_params[5].is_null = 0;

60     ps_params[6].buffer_type = MYSQL_TYPE_LONG;
       ps_params[6].buffer = &costo;
       ps_params[6].length = &length[6];
       ps_params[6].buffer_length = sizeof(int);
       ps_params[6].is_null = 0;

65     status = mysql_stmt_bind_param(stmt, ps_params);
       test_stmt_error(stmt, status);

70     status = mysql_stmt_execute(stmt);
       test_stmt_error(stmt, status);

75     mysql_stmt_close(stmt);
       printf("Operazione completata. Verifica corretto inserimento dalle proiezioni disponibili\n");
   }

```

```

void aggiungiProiezioneNoProiez()
{
    int cinema;
    memset(&cinema, 0, 4);
    int sala;
    memset(&sala, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
    MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));
    char film[128];
    memset(film, 0, 128);
    int costo;
    memset(&costo, 0, 4);

    15   data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
    printf("Anno: ");
    scanf("%d", &data.year);
    fflush(stdin);
    20   printf("Mese: ");
    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
    fflush(stdin);
    printf("Ora: ");
    scanf("%d", &ora.hour);
    fflush(stdin);
    25   printf("Minuti: ");
    scanf("%d", &ora.minute);
    fflush(stdin);
    ora.second = 0;
    printf("Sala: ");
    scanf("%d", &sala);
    fflush(stdin);
    30   printf("Costo: ");
    scanf("%d", &costo);
    fflush(stdin);
    printf("Film: ");
    scanf("%[^\\n]", film);
    fflush(stdin);

    35
    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[6];
    unsigned long length[6];
    length[3] = sizeof(int);
    length[1] = sizeof(ora);
    length[2] = strlen(film);
    40   length[0] = sizeof(data);
    length[4] = sizeof(int);
    length[5] = sizeof(int);
    int status;

    45
    stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\\n");
        exit(1);
    }

    50   char *query = "call aggiungiProiezioneSenzaProiezionista(?,?,?,?,?,?)";
    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    55
    memset(ps_params, 0, sizeof(ps_params));

    60   ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
    ps_params[1].buffer = (char *)&ora;
    ps_params[1].length = 0;

```

```

ps_params[1].is_null = 0;

ps_params[0].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[0].buffer = (char *)&data;
5 ps_params[0].length = 0;
ps_params[0].is_null = 0;

ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[2].buffer = film;
10 ps_params[2].length = &length[2];
ps_params[2].is_null = 0;

ps_params[4].buffer_type = MYSQL_TYPE_LONG;
ps_params[4].buffer = &sala;
15 ps_params[4].length = &length[4];
ps_params[4].buffer_length = sizeof(int);
ps_params[4].is_null = 0;

ps_params[3].buffer_type = MYSQL_TYPE_LONG;
ps_params[3].buffer = &cinema;
20 ps_params[3].length = &length[3];
ps_params[3].buffer_length = sizeof(int);
ps_params[3].is_null = 0;

ps_params[5].buffer_type = MYSQL_TYPE_LONG;
ps_params[5].buffer = &costo;
25 ps_params[5].length = &length[5];
ps_params[5].buffer_length = sizeof(int);
ps_params[5].is_null = 0;

30 status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
35 test_stmt_error(stmt, status);

mysql_stmt_close(stmt);

printf("Operazione completata. Verifica corretto inserimento dalle proiezioni disponibili\n");
40 }
}

void calendarioTurni()
{
45   printf("Procedura che permette di stampare i turni dalla data corrente fino alla domenica piu vicina\n");

    int cinema;
    memset(&cinema, 0, 4);
    char fasciaOraria[6];
50   memset(fasciaOraria, 0, 6);
    char dipendente[45];
    memset(fasciaOraria, 0, 45);
    char giorno[10];
    memset(fasciaOraria, 0, 10);

55   printf("Cinema del calendario dei turni: ");
    scanf("%d", &cinema);
    fflush(stdin);

60   MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[1];
    unsigned long length[1];
    length[0] = sizeof(int);
    int num_fields, status, i;

65   MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[3];
70   stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\n");
        exit(1);
75   }
}

```

```

char *query = "call calendarioTurni(?)";
status = mysql_stmt_prepare(stmt, query, strlen(query));
5 test_stmt_error(stmt, status);

memset(ps_params, 0, sizeof(ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_LONG;
10 ps_params[0].buffer = &cinema;
ps_params[0].length = &length[0];
ps_params[0].buffer_length = sizeof(int);
ps_params[0].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
15 test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

20 do {
    num_fields = mysql_stmt_field_count(stmt);

    if (num_fields > 0) {
25        if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

        rs_metadata = mysql_stmt_result_metadata(stmt);
30        test_stmt_error(stmt, rs_metadata == NULL);

        fields = mysql_fetch_fields(rs_metadata);
        rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        if (!rs_bind) {
35            printf("Cannot allocate output buffers\n");
            exit(1);
        }
        memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);
        rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = fasciaOraria;
        rs_bind[0].buffer_length = sizeof(fasciaOraria);

40        rs_bind[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        rs_bind[1].is_null = &is_null[1];
        rs_bind[1].buffer = giorno;
        rs_bind[1].buffer_length = sizeof(giorno);

        rs_bind[2].buffer_type = MYSQL_TYPE_VAR_STRING;
45        rs_bind[2].is_null = &is_null[2];
        rs_bind[2].buffer = dipendente;
        rs_bind[2].buffer_length = sizeof(dipendente);

50        status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

55        while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }

60            printf("Fascia Oraria: %s, Giorno: %s, Dipendente: %s\n", rs_bind[0].buffer, rs_bind[1].buffer,
65            rs_bind[2].buffer);
        }

        mysql_free_result(rs_metadata);
        free(rs_bind);
70    } else {
        printf("-----\n");
    }

75    status = mysql_stmt_next_result(stmt);
    if (status > 0)
}

```

```

        test_stmt_error(stmt, status);
    } while (status == 0);

5     mysql_stmt_close(stmt);

void turniDelGiorno()
{
10    int cinema;
    memset(&cinema, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
    char fasciaOraria[6];
15    memset(fasciaOraria, 0, 6);
    char dipendente[45];
    memset(fasciaOraria, 0, 45);
    char giorno[10];
    memset(fasciaOraria, 0, 10);
20    char tipo[15];
    memset(tipo, 0, 15);

    printf("Cinema: ");
    scanf("%d", &cinema);
25    fflush(stdin);

    data.time_type = MYSQL_TYPE_DATETIME;
    printf("Anno: ");
    scanf("%d", &data.year);
30    fflush(stdin);
    printf("Mese: ");
    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
35    fflush(stdin);

    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[2];
40    unsigned long length[2];
    length[0] = sizeof(int);
    length[1] = sizeof(data);
    int num_fields, status, i;

45    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[4];

50    stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\n");
        exit(1);
    }

55    char *query = "select TD.FasciaOrariaTurno, TD.GiornoTurno, TD.CodiceFiscaleDipendente, D.Tipo from TurnoDelDipendente TD join
Dipendente D on TD.CodiceFiscaleDipendente = D.CodiceFiscale where D.Cinema = ? and TD.GiornoTurno = ?";

60    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    memset(ps_params, 0, sizeof(ps_params));

65    ps_params[0].buffer_type = MYSQL_TYPE_LONG;
    ps_params[0].buffer = &cinema;
    ps_params[0].length = &length[0];
    ps_params[0].buffer_length = sizeof(int);
    ps_params[0].is_null = 0;

70    ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
    ps_params[1].buffer = (char *)&data;
    ps_params[1].length = 0;
    ps_params[1].is_null = 0;

75    status = mysql_stmt_bind_param(stmt, ps_params);
}

```

```

test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

5      do {
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {

10         if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

15         rs_metadata = mysql_stmt_result_metadata(stmt);
         test_stmt_error(stmt, rs_metadata == NULL);

         fields = mysql_fetch_fields(rs_metadata);
         rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
         if (!rs_bind) {
20             printf("Cannot allocate output buffers\n");
             exit(1);
         }
         memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

25         rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
         rs_bind[0].is_null = &is_null[0];
         rs_bind[0].buffer = fasciaOraria;
         rs_bind[0].buffer_length = sizeof(fasciaOraria);

30         rs_bind[1].buffer_type = MYSQL_TYPE_VAR_STRING;
         rs_bind[1].is_null = &is_null[1];
         rs_bind[1].buffer = giorno;
         rs_bind[1].buffer_length = sizeof(giorno);

35         rs_bind[2].buffer_type = MYSQL_TYPE_VAR_STRING;
         rs_bind[2].is_null = &is_null[2];
         rs_bind[2].buffer = dipendente;
         rs_bind[2].buffer_length = sizeof(dipendente);

40         rs_bind[3].buffer_type = MYSQL_TYPE_VAR_STRING;
         rs_bind[3].is_null = &is_null[3];
         rs_bind[3].buffer = tipo;
         rs_bind[3].buffer_length = sizeof(tipo);

45         status = mysql_stmt_bind_result(stmt, rs_bind);
         test_stmt_error(stmt, status);

         while(1){
             status = mysql_stmt_fetch(stmt);

50             if(status == 1 || status == MYSQL_NO_DATA){
                 break;
             }else{
55                 printf("Fascia Oraria: %s, Giorno: %s, Dipendente: %s, Tipo: %s\n", rs_bind[0].buffer,
                     rs_bind[1].buffer, rs_bind[2].buffer, rs_bind[3].buffer);
             }
         }

60         mysql_free_result(rs_metadata);
         free(rs_bind);
     } else {
65         printf("-----\n");
     }

     status = mysql_stmt_next_result(stmt);
     if (status > 0)
         test_stmt_error(stmt, status);
} while (status == 0);

70     mysql_stmt_close(stmt);
}

void inserisciFilm()
{
}

```

```

char film[128];
memset(film, 0, 128);
char casa[45];
memset(casa, 0, 45);
5      int durata;
memset(&durata, 0, 4);
printf("Nome del film da inserire: ");
scanf("%[^\\n]", film);
fflush(stdin);

10     printf("Nome della casa cinematografica: ");
scanf("%[^\\n]", casa);
fflush(stdin);

15     printf("Durata in minuti: ");
scanf("%d", &durata);
fflush(stdin);

20     MYSQL_STMT *stmt;
MYSQL_BIND ps_params[3];
unsigned long length[3];
length[0] = strlen(film);
length[2] = strlen(casa);
length[1] = sizeof(int);
25      int status;

stmt = mysql_stmt_init(con);
if (!stmt) {
30         printf("Could not initialize statement\\n");
exit(1);
}

char *query = "insert into Film values(?, ?, ?)";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

35      memset(ps_params, 0, sizeof(ps_params));

ps_params[1].buffer_type = MYSQL_TYPE_LONG;
40      ps_params[1].buffer = &durata;
ps_params[1].length = &length[1];
ps_params[1].buffer_length = sizeof(int);
ps_params[1].is_null = 0;

45      ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[0].buffer = film;
ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

50      ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[2].buffer = casa;
ps_params[2].length = &length[2];
ps_params[2].is_null = 0;

55      status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

60      printf("Inserimento effettuato\\n");
}

65 void inserisciAttore()
{
    char nome[45];
memset(nome, 0, 45);
    char cognome[45];
70      memset(cognome, 0, 45);
printf("Nome dell'attore da inserire: ");
scanf("%[^\\n]", nome);
fflush(stdin);

75      printf("Cognome: ");

```

```

scanf("%[^\\n]", cognome);
fflush(stdin);

5      MYSQL_STMT *stmt;
MYSQL_BIND ps_params[2];
unsigned long length[2];
length[0] = strlen(nome);
length[1] = strlen(cognome);
int status;
10     stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\\n");
    exit(1);
15 }

char *query = "insert into Attore values(?, ?)";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

20     memset(ps_params, 0, sizeof(ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[0].buffer = nome;
25     ps_params[0].length = &length[0];
ps_params[0].is_null = 0;

ps_params[1].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[1].buffer = cognome;
30     ps_params[1].length = &length[1];
ps_params[1].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

35     status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

printf("Inserimento effettuato\\n");
40 }
}

void legaFilmAttore()
{
45     char film[128];
memset(film, 0, 128);
char nome[45];
memset(nome, 0, 45);
char cognome[45];
50     memset(cognome, 0, 45);

printf("Nome del film: ");
scanf("%[^\\n]", film);
fflush(stdin);

55     printf("Nome dell'attore: ");
scanf("%[^\\n]", nome);
fflush(stdin);

printf("Cognome: ");
scanf("%[^\\n]", cognome);
fflush(stdin);

60     MYSQL_STMT *stmt;
MYSQL_BIND ps_params[3];
unsigned long length[3];
length[0] = strlen(nome);
length[1] = strlen(cognome);
length[2] = strlen(film);
int status;
70     stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\\n");
    exit(1);
75 }
}

```

```

    }

    char *query = "insert into AttoriNelFilm values(?, ?, ?)";
    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

5      memset(ps_params, 0, sizeof(ps_params));

    ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[0].buffer = nome;
    ps_params[0].length = &length[0];
    ps_params[0].is_null = 0;

10     ps_params[1].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[1].buffer = cognome;
    ps_params[1].length = &length[1];
    ps_params[1].is_null = 0;

15     ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
    ps_params[2].buffer = film;
    ps_params[2].length = &length[2];
    ps_params[2].is_null = 0;

20     status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

25     status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

30     printf("Inserimento effettuato\n");

    }

35 void bigliettiConfermati()
{
    int cinema;
    memset(&cinema, 0, 4);
    MYSQL_TIME data;
    40   memset(&data, 0, sizeof(data));
    int mese;
    memset(&mese, 0, 4);
    int totale;
    memset(&totale, 0, 4);

45   printf("Cinema in cui contare i biglietti confermati: ");
    scanf("%d", &cinema);
    fflush(stdin);

50   data.time_type = MYSQL_TYPE_DATETIME;
    printf("Mese in cui contare biglietti confermati: ");
    scanf("%d", &mese);
    fflush(stdin);

55   MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[2];
    unsigned long length[2];
    length[0] = sizeof(int);
    length[1] = sizeof(int);
    60   int num_fields, status, i;

    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[1];

65   stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\n");
        exit(1);
    }

70   char *query = "select count(*) from Biglietto where MONTH(DataProiezione) = ? and Tipo = 'Confermato' and CinemaDellaProiezione
= ?";
    75   status = mysql_stmt_prepare(stmt, query, strlen(query));

```

```

test_stmt_error(stmt, status);

memset(ps_params, 0, sizeof(ps_params));

5      ps_params[1].buffer_type = MYSQL_TYPE_LONG;
      ps_params[1].buffer = &cinema;
      ps_params[1].length = &length[1];
      ps_params[1].buffer_length = sizeof(int);
      ps_params[1].is_null = 0;

10     ps_params[0].buffer_type = MYSQL_TYPE_LONG;
      ps_params[0].buffer = &mese;
      ps_params[0].length = &length[0];
      ps_params[0].buffer_length = sizeof(int);
15     ps_params[0].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

20     status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

do {
    num_fields = mysql_stmt_field_count(stmt);
25
    if (num_fields > 0) {
        if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");
30
        rs_metadata = mysql_stmt_result_metadata(stmt);
        test_stmt_error(stmt, rs_metadata == NULL);

        fields = mysql_fetch_fields(rs_metadata);
        rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
        }
40        memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

        rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = &totale;
        rs_bind[0].buffer_length = sizeof(int);

        status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

50        while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Numero biglietti confermati nel mese inserito: --> %d <--\n", totale);
            }
        }

60        mysql_free_result(rs_metadata);
        free(rs_bind);
    } else {
        printf("-----\n");
    }
65        status = mysql_stmt_next_result(stmt);
        if (status > 0)
            test_stmt_error(stmt, status);
} while (status == 0);

70     mysql_stmt_close(stmt);

}

```

```

void bigliettiAnnullati()
{
    int cinema;
    memset(&cinema, 0, 4);
    5      MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
    int mese;
    memset(&mese, 0, 4);
    int totale;
    10     memset(&totale, 0, 4);

    printf("Cinema in cui contare i biglietti annullati: ");
    scanf("%d", &cinema);
    fflush(stdin);

    15     data.time_type = MYSQL_TYPE_DATETIME;
    printf("Mese in cui contare biglietti annullati: ");
    scanf("%d", &mese);
    fflush(stdin);

    20     MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[2];
    unsigned long length[2];
    25     length[0] = sizeof(int);
    length[1] = sizeof(int);
    int num_fields, status, i;

    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    30     MYSQL_BIND *rs_bind;
    my_bool is_null[1];

    stmt = mysql_stmt_init(con);
    if (!stmt) {
        35         printf("Could not initialize statement\n");
        exit(1);
    }

    char *query = "select count(*) from Biglietto where MONTH(DataProiezione) = ? and Tipo = 'Annullato' and CinemaDellaProiezione = ?";
    40     status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    memset(ps_params, 0, sizeof(ps_params));

    45     ps_params[1].buffer_type = MYSQL_TYPE_LONG;
    ps_params[1].buffer = &cinema;
    ps_params[1].length = &length[1];
    ps_params[1].buffer_length = sizeof(int);
    ps_params[1].is_null = 0;

    50     ps_params[0].buffer_type = MYSQL_TYPE_LONG;
    ps_params[0].buffer = &mese;
    ps_params[0].length = &length[0];
    ps_params[0].buffer_length = sizeof(int);
    ps_params[0].is_null = 0;

    55     status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

    status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

    do {
        60         num_fields = mysql_stmt_field_count(stmt);
        if (num_fields > 0) {

            if (con->server_status & SERVER_PS_OUT_PARAMS)
                printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

            65         rs_metadata = mysql_stmt_result_metadata(stmt);
            test_stmt_error(stmt, rs_metadata == NULL);

            fields = mysql_fetch_fields(rs_metadata);
            rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        }
    }
}

```

```

        if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
        }
5      memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

        rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = &totale;
        rs_bind[0].buffer_length = sizeof(int);

10     status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

15     while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Numero biglietti annullati nel mese inserito: --> %d <--\n", totale);
            }
        }

20     mysql_free_result(rs_metadata);
        free(rs_bind);
25     } else {
            printf("-----\n");
        }

30     status = mysql_stmt_next_result(stmt);
        if (status > 0)
            test_stmt_error(stmt, status);
    } while (status == 0);

35     mysql_stmt_close(stmt);
}

40 void bigliettiAcquistatiNonConfermati()
{
45     int cinema;
        memset(&cinema, 0, 4);
        MYSQL_TIME data;
        memset(&data, 0, sizeof(data));
        int mese;
        memset(&mese, 0, 4);
        int totale;
50     memset(&totale, 0, 4);

        printf("Cinema in cui contare i biglietti acquistati non confermati: ");
        scanf("%d", &cinema);
        fflush(stdin);

55     data.time_type = MYSQL_TYPE_DATETIME;
        printf("Mese in cui contare biglietti acquistati non confermati: ");
        scanf("%d", &mese);
        fflush(stdin);

60     MYSQL_STMT *stmt;
        MYSQL_BIND ps_params[2];
        unsigned long length[2];
        length[0] = sizeof(int);
65     length[1] = sizeof(int);
        int num_fields, status, i;

        MYSQL_RES *rs_metadata;
        MYSQL_FIELD *fields;
        MYSQL_BIND *rs_bind;
        my_bool is_null[1];

70     stmt = mysql_stmt_init(con);
        if (!stmt) {
            printf("Could not initialize statement\n");
}

```

```

        exit(1);
    }

    char *query = "select count(*) from Biglietto where MONTH(DataProiezione) = ? and Tipo = 'AcquistatoNonConfermato' and
5     CinemaDellaProiezione = ?";
    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    memset(ps_params, 0, sizeof(ps_params));
10
    ps_params[1].buffer_type = MYSQL_TYPE_LONG;
    ps_params[1].buffer = &cinema;
    ps_params[1].length = &length[1];
    ps_params[1].buffer_length = sizeof(int);
15
    ps_params[1].is_null = 0;

    ps_params[0].buffer_type = MYSQL_TYPE_LONG;
    ps_params[0].buffer = &mese;
    ps_params[0].length = &length[0];
20
    ps_params[0].buffer_length = sizeof(int);
    ps_params[0].is_null = 0;

    status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

25
    status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

    do {
30
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {

            if (con->server_status & SERVER_PS_OUT_PARAMS)
                printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

35
            rs_metadata = mysql_stmt_result_metadata(stmt);
            test_stmt_error(stmt, rs_metadata == NULL);

            fields = mysql_fetch_fields(rs_metadata);
            rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
            if (!rs_bind) {
                printf("Cannot allocate output buffers\n");
                exit(1);
40
            }
            memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

            rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
            rs_bind[0].is_null = &is_null[0];
            rs_bind[0].buffer = &totale;
            rs_bind[0].buffer_length = sizeof(int);

50
            status = mysql_stmt_bind_result(stmt, rs_bind);
            test_stmt_error(stmt, status);

            while(1){
                status = mysql_stmt_fetch(stmt);

                if(status == 1 || status == MYSQL_NO_DATA){
                    break;
                }else{
                    printf("Numero biglietti acquistati ma non confermati nel mese inserito: --> %d <--\n", totale);
                }
45
            }

            mysql_free_result(rs_metadata);
            free(rs_bind);
55
        } else {
            printf("-----\n");
        }

        status = mysql_stmt_next_result(stmt);
        if(status > 0)
            test_stmt_error(stmt, status);
70
    } while (status == 0);
75
}

```

```

    mysql_stmt_close(stmt);

}

5

void bigliettiConfermatiProiez()
{
    char film[128];
10   memset(film, 0, 128);
    int totale;
    memset(&totale, 0, 4);
    int scelta;
    int cinema;
15   memset(&cinema, 0, 4);
    int sala;
    memset(&sala, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
20   MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));

    data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

25   printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
    printf("Sala: ");
30   scanf("%d", &sala);
    fflush(stdin);
    puts("Film: ");
    scanf("%[^\\n]", film);
    fflush(stdin);
35   printf("Anno: ");
    scanf("%d", &data.year);
    fflush(stdin);
    printf("Mese: ");
    scanf("%d", &data.month);
40   fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
    fflush(stdin);
    printf("Ora: ");
45   scanf("%d", &ora.hour);
    fflush(stdin);
    printf("Minuti: ");
    scanf("%d", &ora.minute);
    fflush(stdin);
50   ora.second = 0;

    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
55   my_bool is_null[1];

    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[5];
    unsigned long length[5];
60   length[0] = sizeof(data);
    length[1] = sizeof(ora);
    length[2] = strlen(film);
    length[3] = sizeof(int);
    length[4] = sizeof(int);
65   int num_fields, status, i;

    stmt = mysql_stmt_init(con);
    if (!stmt) {
        70   printf("Could not initialize statement\\n");
        exit(1);
    }

    char *query = "select count(*) from Biglietto where Tipo = 'Confermato' and DataProiezione = ? and OraProiezione = ? and Film = ? and
SalaDellaProiezione = ? and CinemaDellaProiezione = ?";
75   status = mysql_stmt_prepare(stmt, query, strlen(query));

```

```

test_stmt_error(stmt, status);

memset(ps_params, 0, sizeof(ps_params));

5      ps_params[3].buffer_type = MYSQL_TYPE_LONG;
      ps_params[3].buffer = &sala;
      ps_params[3].length = &length[3];
      ps_params[3].buffer_length = sizeof(int);
      ps_params[3].is_null = 0;

10     ps_params[4].buffer_type = MYSQL_TYPE_LONG;
      ps_params[4].buffer = &cinema;
      ps_params[4].length = &length[4];
      ps_params[4].buffer_length = sizeof(int);
      ps_params[4].is_null = 0;

15     ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
      ps_params[2].buffer = film;
      ps_params[2].length = &length[2];
      ps_params[2].is_null = 0;

20     ps_params[0].buffer_type = MYSQL_TYPE_DATETIME;
      ps_params[0].buffer = (char *)&data;
      ps_params[0].length = 0;
      ps_params[0].is_null = 0;

25     ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
      ps_params[1].buffer = (char *)&ora;
      ps_params[1].length = 0;
      ps_params[1].is_null = 0;

30     status = mysql_stmt_bind_param(stmt, ps_params);
     test_stmt_error(stmt, status);

35     status = mysql_stmt_execute(stmt);
     test_stmt_error(stmt, status);

do {
    num_fields = mysql_stmt_field_count(stmt);
40
    if (num_fields > 0) {
        if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");
45
        rs_metadata = mysql_stmt_result_metadata(stmt);
        test_stmt_error(stmt, rs_metadata == NULL);

        fields = mysql_fetch_fields(rs_metadata);
        rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
        }
50
        memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

        rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = &totale;
        rs_bind[0].buffer_length = sizeof(int);

55
        status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

60
        while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Numero biglietti confermati per la proiezione inserita: --> %d <--\n", totale);
            }
4
        }

75
        mysql_free_result(rs_metadata);
}

```

```

        free(rs_bind);
    } else {
        printf("-----\n");
    }
5
    status = mysql_stmt_next_result(stmt);
    if (status > 0)
        test_stmt_error(stmt, status);
} while (status == 0);

10 mysql_stmt_close(stmt);

}

15 void bigliettiAnnullatiProiez()
{
    char film[128];
    memset(film, 0, 128);
    int totale;
20    memset(&totale, 0, 4);
    int scelta;
    int cinema;
    memset(&cinema, 0, 4);
    int sala;
25    memset(&sala, 0, 4);
    MYSQL_TIME data;
    memset(&data, 0, sizeof(data));
    MYSQL_TIME ora;
    memset(&ora, 0, sizeof(ora));

30    data.time_type = MYSQL_TYPE_DATETIME;
    ora.time_type = MYSQL_TYPE_DATETIME;

35    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
    printf("Sala: ");
    scanf("%d", &sala);
    fflush(stdin);
40    puts("Film: ");
    scanf("%[^\\n]", film);
    fflush(stdin);
    printf("Anno: ");
    scanf("%d", &data.year);
    fflush(stdin);
45    printf("Mese: ");
    scanf("%d", &data.month);
    fflush(stdin);
    printf("Giorno: ");
    scanf("%d", &data.day);
    fflush(stdin);
50    printf("Ora: ");
    scanf("%d", &ora.hour);
    fflush(stdin);
    printf("Minuti: ");
    scanf("%d", &ora.minute);
    fflush(stdin);
55    ora.second = 0;

60    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[1];

65    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[5];
    unsigned long length[5];
    length[0] = sizeof(data);
    length[1] = sizeof(ora);
70    length[2] = strlen(film);
    length[3] = sizeof(int);
    length[4] = sizeof(int);
    int num_fields, status, i;

75    stmt = mysql_stmt_init(con);
}

```

```

if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}

5      char *query = "select count(*) from Biglietto where Tipo = 'Annullato' and DataProiezione = ? and OraProiezione = ? and Film = ? and
SalaDellaProiezione = ? and CinemaDellaProiezione = ?";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

10     memset(ps_params, 0, sizeof(ps_params));

        ps_params[3].buffer_type = MYSQL_TYPE_LONG;
        ps_params[3].buffer = &sala;
15        ps_params[3].length = &length[3];
        ps_params[3].buffer_length = sizeof(int);
        ps_params[3].is_null = 0;

        ps_params[4].buffer_type = MYSQL_TYPE_LONG;
20        ps_params[4].buffer = &cinema;
        ps_params[4].length = &length[4];
        ps_params[4].buffer_length = sizeof(int);
        ps_params[4].is_null = 0;

        ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
25        ps_params[2].buffer = film;
        ps_params[2].length = &length[2];
        ps_params[2].is_null = 0;

30        ps_params[0].buffer_type = MYSQL_TYPE_DATETIME;
        ps_params[0].buffer = (char *)&data;
        ps_params[0].length = 0;
        ps_params[0].is_null = 0;

35        ps_params[1].buffer_type = MYSQL_TYPE_DATETIME;
        ps_params[1].buffer = (char *)&ora;
        ps_params[1].length = 0;
        ps_params[1].is_null = 0;

40        status = mysql_stmt_bind_param(stmt, ps_params);
        test_stmt_error(stmt, status);

        status = mysql_stmt_execute(stmt);
        test_stmt_error(stmt, status);

45        do {
            num_fields = mysql_stmt_field_count(stmt);

            if (num_fields > 0) {

50                if (con->server_status & SERVER_PS_OUT_PARAMS)
                    printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

55                rs_metadata = mysql_stmt_result_metadata(stmt);
                test_stmt_error(stmt, rs_metadata == NULL);

                fields = mysql_fetch_fields(rs_metadata);
                rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
                if (!rs_bind) {
60                    printf("Cannot allocate output buffers\n");
                    exit(1);
                }
                memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

65                rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
                rs_bind[0].is_null = &is_null[0];
                rs_bind[0].buffer = &totale;
                rs_bind[0].buffer_length = sizeof(int);

70                status = mysql_stmt_bind_result(stmt, rs_bind);
                test_stmt_error(stmt, status);

                while(1){
                    status = mysql_stmt_fetch(stmt);
}

```

75

```

        if(status == 1 || status == MYSQL_NO_DATA){
            break;
        }else{
            printf("Numero biglietti annullati per la proiezione inserita: --> %d <--\n", totale);
        }
    }

    mysql_free_result(rs_metadata);
    free(rs_bind);
10   } else {
        printf("-----\n");
    }

    status = mysql_stmt_next_result(stmt);
15   if (status > 0)
        test_stmt_error(stmt, status);
    } while (status == 0);

mysql_stmt_close(stmt);
20 }

void simulaPostProiezione()
{
25   char film[128];
   memset(film, 0, 128);
   int totale;
   memset(&totale, 0, 4);
   int scelta;
30   int cinema;
   memset(&cinema, 0, 4);
   int sala;
   memset(&sala, 0, 4);
   MYSQL_TIME data;
   memset(&data, 0, sizeof(data));
35   MYSQL_TIME ora;
   memset(&ora, 0, sizeof(ora));

   data.time_type = MYSQL_TYPE_DATETIME;
   ora.time_type = MYSQL_TYPE_DATETIME;

40   printf("Funzione che simula la conferma e l'annullamento dei biglietti\n di una proiezione a cui sono associati dei biglietti\n");

   printf("Cinema: ");
45   scanf("%d", &cinema);
   fflush(stdin);
   printf("Sala: ");
   scanf("%d", &sala);
   fflush(stdin);
50   printf("Film: ");
   scanf("%[^\\n]", film);
   fflush(stdin);
   printf("Anno: ");
   scanf("%d", &data.year);
   fflush(stdin);
55   printf("Mese: ");
   scanf("%d", &data.month);
   fflush(stdin);
   printf("Giorno: ");
   scanf("%d", &data.day);
   fflush(stdin);
60   printf("Ora: ");
   scanf("%d", &ora.hour);
   fflush(stdin);
   printf("Minuti: ");
65   scanf("%d", &ora.minute);
   fflush(stdin);
   ora.second = 0;

70   MYSQL_STMT *stmt;
   MYSQL_BIND ps_params[5];
   unsigned long length[5];
   length[3] = sizeof(data);
   length[4] = sizeof(ora);
   length[2] = strlen(film);
75
}

```

```

length[0] = sizeof(int);
length[1] = sizeof(int);
int status;

5      stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
}

10     char *query = "call simulaPostProiezione(?,?,?,?,?)";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

15     memset(ps_params, 0, sizeof(ps_params));

ps_params[1].buffer_type = MYSQL_TYPE_LONG;
ps_params[1].buffer = &sala;
ps_params[1].length = &length[1];
ps_params[1].buffer_length = sizeof(int);
ps_params[1].is_null = 0;

20     ps_params[0].buffer_type = MYSQL_TYPE_LONG;
ps_params[0].buffer = &cinema;
ps_params[0].length = &length[0];
ps_params[0].buffer_length = sizeof(int);
ps_params[0].is_null = 0;

25     ps_params[2].buffer_type = MYSQL_TYPE_VAR_STRING;
ps_params[2].buffer = film;
ps_params[2].length = &length[2];
ps_params[2].is_null = 0;

30     ps_params[3].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[3].buffer = (char *)&data;
ps_params[3].length = 0;
ps_params[3].is_null = 0;

35     ps_params[4].buffer_type = MYSQL_TYPE_DATETIME;
ps_params[4].buffer = (char *)&ora;
ps_params[4].length = 0;
ps_params[4].is_null = 0;

40     status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

45     status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

50     mysql_stmt_close(stmt);

printf("Simulazione completata\n");
}

55 void incassoFilmCinema()
{
    char film[128];
    memset(film, 0, 128);
    int totale;
60    memset(&totale, 0, 4);
    int cinema;
    memset(&cinema, 0, 4);

    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);
    puts("Film: ");
    scanf("%[^\\n]", film);
70    fflush(stdin);

    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[1];
}

```

```

5      MYSQL_STMT *stmt;
       MYSQL_BIND ps_params[2];
       unsigned long length[2];
       length[1] = strlen(film);
       length[0] = sizeof(int);
       int num_fields, status, i;

10     stmt = mysql_stmt_init(con);
       if (!stmt) {
           printf("Could not initialize statement\n");
           exit(1);
       }

15     char *query = "select sum(IncassoTotale) from Proiezione where Cinema = ? and Film = ?";
       status = mysql_stmt_prepare(stmt, query, strlen(query));
       test_stmt_error(stmt, status);

20     memset(ps_params, 0, sizeof(ps_params));
       ps_params[0].buffer_type = MYSQL_TYPE_LONG;
       ps_params[0].buffer = &cinema;
       ps_params[0].length = &length[0];
       ps_params[0].buffer_length = sizeof(int);
       ps_params[0].is_null = 0;

25     ps_params[1].buffer_type = MYSQL_TYPE_VAR_STRING;
       ps_params[1].buffer = film;
       ps_params[1].length = &length[1];
       ps_params[1].is_null = 0;

30     status = mysql_stmt_bind_param(stmt, ps_params);
       test_stmt_error(stmt, status);

35     status = mysql_stmt_execute(stmt);
       test_stmt_error(stmt, status);

40     do {
       num_fields = mysql_stmt_field_count(stmt);
       if (num_fields > 0) {

45         if (con->server_status & SERVER_PS_OUT_PARAMS)
           printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

50         rs_metadata = mysql_stmt_result_metadata(stmt);
         test_stmt_error(stmt, rs_metadata == NULL);

         fields = mysql_fetch_fields(rs_metadata);
         rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
         if (!rs_bind) {
             printf("Cannot allocate output buffers\n");
             exit(1);
         }
55         memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

         rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
         rs_bind[0].is_null = &is_null[0];
         rs_bind[0].buffer = &totale;
         rs_bind[0].buffer_length = sizeof(int);

60         status = mysql_stmt_bind_result(stmt, rs_bind);
         test_stmt_error(stmt, status);

65         while(1){
             status = mysql_stmt_fetch(stmt);

             if(status == 1 || status == MYSQL_NO_DATA){
                 break;
             }else{
                 printf("Incasso totale dato da tutte le poiezioni del film inserito: --> %d € <--\n", totale);
             }
         }

75         mysql_free_result(rs_metadata);

```

```

        free(rs_bind);
    } else {
        printf("-----\n");
    }
5
    status = mysql_stmt_next_result(stmt);
    if (status > 0)
        test_stmt_error(stmt, status);
} while (status == 0);

10   mysql_stmt_close(stmt);
}

15 void incassoFilm()
{
    char film[128];
    memset(film, 0, 128);
    int totale;
20    memset(&totale, 0, 4);

    puts("Film: ");
    scanf("%[^\\n]", film);
    fflush(stdin);

25    MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[1];

30    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[1];
    unsigned long length[1];
    length[0] = strlen(film);
    int num_fields, status, i;

35    stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\n");
        exit(1);
    }

40    char *query = "select sum(IncassoTotale) from Proiezione where Film = ?";
    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

45    memset(ps_params, 0, sizeof(ps_params));

    ps_params[0].buffer_type = MYSQL_TYPE_VAR_STRING;
50    ps_params[0].buffer = film;
    ps_params[0].length = &length[0];
    ps_params[0].is_null = 0;

    status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

55    status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

    do {
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {
            if (con->server_status & SERVER_PS_OUT_PARAMS)
                printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

60            rs_metadata = mysql_stmt_result_metadata(stmt);
            test_stmt_error(stmt, rs_metadata == NULL);

65            fields = mysql_fetch_fields(rs_metadata);
            rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
            if (!rs_bind) {
                printf("Cannot allocate output buffers\n");
                exit(1);
            }
        }
    }
}

```

```

        }
        memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

5      rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = &totale;
        rs_bind[0].buffer_length = sizeof(int);

10     status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

        while(1){
            status = mysql_stmt_fetch(stmt);

15         if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Incasso totale dato da tutte le poiezioni del film inserito: --> %d € <--\n", totale);
            }
        }

20     mysql_free_result(rs_metadata);
        free(rs_bind);

25     } else {
        printf("-----\n");
    }

30     status = mysql_stmt_next_result(stmt);
        if(status > 0)
            test_stmt_error(stmt, status);
    } while (status == 0);

35     mysql_stmt_close(stmt);

void incassoCinema()
{
40     int totale;
        memset(&totale, 0, 4);
        int cinema;
        memset(&cinema, 0, 4);

45     printf("Cinema: ");
        scanf("%d", &cinema);
        fflush(stdin);

50     MYSQL_RES *rs_metadata;
        MYSQL_FIELD *fields;
        MYSQL_BIND *rs_bind;
        my_bool is_null[1];

55     MYSQL_STMT *stmt;
        MYSQL_BIND ps_params[1];
        unsigned long length[1];
        length[0] = sizeof(int);
        int num_fields, status, i;

60     stmt = mysql_stmt_init(con);
        if (!stmt) {
            printf("Could not initialize statement\n");
            exit(1);
        }

65     char *query = "select sum(IncassoTotale) from Proiezione where Cinema = ?";
        status = mysql_stmt_prepare(stmt, query, strlen(query));
        test_stmt_error(stmt, status);

70     memset(ps_params, 0, sizeof(ps_params));

        ps_params[0].buffer_type = MYSQL_TYPE_LONG;
        ps_params[0].buffer = &cinema;
        ps_params[0].length = &length[0];
        ps_params[0].buffer_length = sizeof(int);

```

```

ps_params[0].is_null = 0;

status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

5      status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

do {
10     num_fields = mysql_stmt_field_count(stmt);

     if (num_fields > 0) {

         if (con->server_status & SERVER_PS_OUT_PARAMS)
             printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

15     rs_metadata = mysql_stmt_result_metadata(stmt);
test_stmt_error(stmt, rs_metadata == NULL);

         fields = mysql_fetch_fields(rs_metadata);
         rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
         if (!rs_bind) {
             printf("Cannot allocate output buffers\n");
             exit(1);
25         }
         memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

         rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
         rs_bind[0].is_null = &is_null[0];
         rs_bind[0].buffer = &totale;
         rs_bind[0].buffer_length = sizeof(int);

30         status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);

         while(1){
35             status = mysql_stmt_fetch(stmt);

             if(status == 1 || status == MYSQL_NO_DATA){
                 break;
             }else{
40                 printf("Incasso totale dato da tutte le poiezioni di tutti i film nel cinema inserito: --> %d €
<--\n", totale);
             }
         }

45         mysql_free_result(rs_metadata);
         free(rs_bind);

         } else {
50             printf("-----\n");
         }

         status = mysql_stmt_next_result(stmt);
         if (status > 0)
             test_stmt_error(stmt, status);
55     } while (status == 0);

     mysql_stmt_close(stmt);

60 }

void incassoCatena()
{
65     int totale;
     memset(&totale, 0, 4);

     MYSQL_RES *rs_metadata;
     MYSQL_FIELD *fields;
     MYSQL_BIND *rs_bind;
70     my_bool is_null[1];

     MYSQL_STMT *stmt;
     int num_fields, status, i;
75
}

```

```

stmt = mysql_stmt_init(con);
if (!stmt) {
    printf("Could not initialize statement\n");
    exit(1);
5 }

char *query = "select sum(IncassoTotale) from Proiezione";
status = mysql_stmt_prepare(stmt, query, strlen(query));
test_stmt_error(stmt, status);

10 status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

do {
    num_fields = mysql_stmt_field_count(stmt);

    if (num_fields > 0) {

        if (con->server_status & SERVER_PS_OUT_PARAMS)
            printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

20 rs_metadata = mysql_stmt_result_metadata(stmt);
test_stmt_error(stmt, rs_metadata == NULL);

        fields = mysql_fetch_fields(rs_metadata);
        rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
        if (!rs_bind) {
            printf("Cannot allocate output buffers\n");
            exit(1);
        }
30     memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

        rs_bind[0].buffer_type = MYSQL_TYPE_LONG;
        rs_bind[0].is_null = &is_null[0];
        rs_bind[0].buffer = &totale;
        rs_bind[0].buffer_length = sizeof(int);

        status = mysql_stmt_bind_result(stmt, rs_bind);
        test_stmt_error(stmt, status);

40     while(1){
            status = mysql_stmt_fetch(stmt);

            if(status == 1 || status == MYSQL_NO_DATA){
                break;
            }else{
                printf("Incasso totale della catena di cinema: --> %d € <--\n", totale);
            }
45     }

50     mysql_free_result(rs_metadata);
     free(rs_bind);
} else {
55     printf("-----\n");
}

status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);

60 } while (status == 0);

mysql_stmt_close(stmt);
}

65 void maschereCinema()
{
70     char dipendente[45];
     memset(dipendente, 0, 45);
     int cinema;
     memset(&cinema, 0, 4);

75     printf("Cinema: ");
     scanf("%d", &cinema);
}

```

```

fflush(stdin);

5      MYSQL_RES *rs_metadata;
      MYSQL_FIELD *fields;
      MYSQL_BIND *rs_bind;
      my_bool is_null[1];

10     MYSQL_STMT *stmt;
      MYSQL_BIND ps_params[1];
      unsigned long length[1];
      length[0] = sizeof(int);
      int num_fields, status, i;
      int j = 0;

15     stmt = mysql_stmt_init(con);
      if (!stmt) {
          printf("Could not initialize statement\n");
          exit(1);
      }

20     char *query = "select CodiceFiscale from Dipendente where Cinema = ? and Tipo = 'Maschera'";
      status = mysql_stmt_prepare(stmt, query, strlen(query));
      test_stmt_error(stmt, status);

25     memset(ps_params, 0, sizeof(ps_params));

      ps_params[0].buffer_type = MYSQL_TYPE_LONG;
      ps_params[0].buffer = &cinema;
      ps_params[0].length = &length[0];
      ps_params[0].buffer_length = sizeof(int);
      ps_params[0].is_null = 0;

30     status = mysql_stmt_bind_param(stmt, ps_params);
      test_stmt_error(stmt, status);

35     status = mysql_stmt_execute(stmt);
      test_stmt_error(stmt, status);

40     do {
          num_fields = mysql_stmt_field_count(stmt);

          if (num_fields > 0) {

45           if (con->server_status & SERVER_PS_OUT_PARAMS)
              printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

              rs_metadata = mysql_stmt_result_metadata(stmt);
              test_stmt_error(stmt, rs_metadata == NULL);

50           fields = mysql_fetch_fields(rs_metadata);
              rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
              if (!rs_bind) {
                  printf("Cannot allocate output buffers\n");
                  exit(1);
              }
              memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

55           rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
              rs_bind[0].is_null = &is_null[0];
              rs_bind[0].buffer = dipendente;
              rs_bind[0].buffer_length = sizeof(dipendente);

60           status = mysql_stmt_bind_result(stmt, rs_bind);
              test_stmt_error(stmt, status);

65           while(1){
              status = mysql_stmt_fetch(stmt);

              if(status == 1 || status == MYSQL_NO_DATA){
                  break;
              }else{
                  printf("Maschera n.%d: -> %s\n", j, dipendente);
              }
              j++;
            }
        }
    }
}

```

```

        mysql_free_result(rs_metadata);
        free(rs_bind);
    } else {
        printf("-----\n");
    }

    status = mysql_stmt_next_result(stmt);
    if (status > 0)
        test_stmt_error(stmt, status);
} while (status == 0);

mysql_stmt_close(stmt);
}

void proiezionistiCinema()
{
    char dipendente[45];
    memset(dipendente, 0, 45);
    int cinema;
    memset(&cinema, 0, 4);

    printf("Cinema: ");
    scanf("%d", &cinema);
    fflush(stdin);

    30   MYSQL_RES *rs_metadata;
    MYSQL_FIELD *fields;
    MYSQL_BIND *rs_bind;
    my_bool is_null[1];

    MYSQL_STMT *stmt;
    MYSQL_BIND ps_params[1];
    unsigned long length[1];
    length[0] = sizeof(int);
    int num_fields, status, i;
    int j = 0;

    40   stmt = mysql_stmt_init(con);
    if (!stmt) {
        printf("Could not initialize statement\n");
        exit(1);
    }

    45   char *query = "select CodiceFiscale from Dipendente where Cinema = ? and Tipo = 'Proiezionista'";
    status = mysql_stmt_prepare(stmt, query, strlen(query));
    test_stmt_error(stmt, status);

    50   memset(ps_params, 0, sizeof(ps_params));

    ps_params[0].buffer_type = MYSQL_TYPE_LONG;
    ps_params[0].buffer = &cinema;
    ps_params[0].length = &length[0];
    ps_params[0].buffer_length = sizeof(int);
    ps_params[0].is_null = 0;

    status = mysql_stmt_bind_param(stmt, ps_params);
    test_stmt_error(stmt, status);

    60   status = mysql_stmt_execute(stmt);
    test_stmt_error(stmt, status);

    do {
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {
            if (con->server_status & SERVER_PS_OUT_PARAMS)
                printf("The stored procedure has returned output in OUT/INOUT parameter(s)\n");

            70   rs_metadata = mysql_stmt_result_metadata(stmt);
            test_stmt_error(stmt, rs_metadata == NULL);

            fields = mysql_fetch_fields(rs_metadata);
        }
    }
}

```

```

rs_bind = (MYSQL_BIND *)malloc(sizeof(MYSQL_BIND) * num_fields);
if (!rs_bind) {
    printf("Cannot allocate output buffers\n");
    exit(1);
}
5      memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

rs_bind[0].buffer_type = MYSQL_TYPE_VAR_STRING;
rs_bind[0].is_null = &is_null[0];
rs_bind[0].buffer = dipendente;
rs_bind[0].buffer_length = sizeof(dipendente);

status = mysql_stmt_bind_result(stmt, rs_bind);
test_stmt_error(stmt, status);
10

while(1){
    status = mysql_stmt_fetch(stmt);

    if(status == 1 || status == MYSQL_NO_DATA){
        break;
    }else{
        15      printf("Proiezionista n.%d: -> %s\n", j, dipendente);
    }
    j++;
}

mysql_free_result(rs_metadata);
free(rs_bind);
20

} else {
30      printf("-----\n");
}

status = mysql_stmt_next_result(stmt);
if (status > 0)
    test_stmt_error(stmt, status);
35 } while (status == 0);

mysql_stmt_close(stmt);
40 }

char *menu0ges = "----- AZIONI PER EFFETTUARE LA 'SEGNALAZIONE'\n| 1)Mostra proiezionista che gestisce una proiezione\n| 2)Mostra
maschere in una fascia oraria di un giorno\n----- \n3)Aggiorna proiezionista della proiezione\n4)Aggiungi maschera in una fascia oraria di un
45 giorno\n";
char *menu1ges = "5)Rimuovi proiezionista da proiezione\n6)Rimuovi maschera in una fascia oraria\n7)Mostra proiezionisti disponibili\n8)Mostra
sale disponibili\n9)Aggiungi proiezione\n";
char *menu2ges = "10)Aggiungi proiezione senza proiezionista\n11)Mostra calendario dei turni\n12)Mostra turni di un giorno specifico\n13)Inserisci
film\n14)Inserisci attore\n15)Lega attore al film\n";
50 char *menu3ges = "16)Numero biglietti confermati nel mese\n17)Numero biglietti annullati nel mese\n18)Numero biglietti acquistati ma non
confermati nel mese\n";
char *menu4ges = "19)Numero biglietti confermati per una certa proiezione\n20)Numero biglietti annullati per una certa proiezione\n";
char *menu5ges = "21)Incasso di un film in un cinema\n22)Incasso di un film nella catena di cinema\n23)Incasso totale di un cinema\n24)Incasso
totale della catena di cinema\n25)Simula post proiezione\n26)Mostra maschere di un cinema\n27)Mostra proiezionisti di un cinema\n28)Indietro\n";
55 void gestore()
{
    char *menuGes[1024];
    memset(menuGes, 0, 1024);
60    strcat(menuGes, menu0ges);
    strcat(menuGes, menu1ges);
    strcat(menuGes, menu2ges);
    strcat(menuGes, menu3ges);
    strcat(menuGes, menu4ges);
    strcat(menuGes, menu5ges);
65

    if (mysql_real_connect(con, conf.host, conf.username, conf.password, conf.database, conf.port, NULL, 0) == NULL) {
        fprintf(stderr, "Connection error: %s\n", mysql_error(con));
        exit(1);
70    }

    int scelta;

    while(1){
        puts(scgli);
75

```

```

5           puts(menuGes);
10          //fflush(stdin);
15          puts(inserisciScelta);
20          scanf("%d", &scelta);
25          fflush(stdin);
30          system("clear");
35          switch (scelta) {
40
45            case 1:
50              mostraProiezionistaDiProiez();
55              break;
60              case 2:
65              mostraMaschere();
70              break;
75              case 3:
80              aggiornaProiezionista();
85              break;
90              case 4:
95              aggiungiMaschera();
100             break;
105             case 5:
110             rimuoviProiezionistaDaProiezione();
115             break;
120             case 6:
125             rimuoviMaschera();
130             break;
135             case 7:
140             proiezionistiDisponibili();
145             break;
150             case 8:
155             saleDisponibili();
160             break;
165             case 9:
170             aggiungiProiezione();
175             break;
180             case 10:
185             aggiungiProiezioneNoProiez();
190             break;
195             case 11:
200             calendarioTurni();
205             break;
210             case 12:
215             turniDelGiorno();
220             break;
225             case 13:
230             inserisciFilm();
235             break;
240             case 14:
245             inserisciAttore();
250             break;
255             case 15:
260             legaFilmAttore();
265             break;
270             case 16:
275             bigliettiConfermati();
280             break;
285             case 17:
290             bigliettiAnnullati();
295             break;
300             case 18:
305             bigliettiAcquistatiNonConfermati();
310             break;
315             case 19:
320             bigliettiConfermatiProiez();
325             break;
330             case 20:
335             bigliettiAnnullatiProiez();
340             break;
345             case 21:
350             incassoFilmCinema();
355             break;
360             case 22:
365             incassoFilm();
370             break;
375             case 23:
380

```

```

                incassoCinema();
                break;
        case 24:
                incassoCatena();
                break;
        case 25:
                simulaPostProiezione();
                break;
        case 26:
                maschereCinema();
                break;
        case 27:
                proiezionistiCinema();
                break;
        case 28:
                mysql_close(con);
                return;
        default:
                puts(sceltaScorretta);
                break;
}
}

25 }

char *menu0 = "1)Entra come Gestore\n2)Entra come Cliente\n3)Entra come Visitatore\n4)Entra come Maschera\n5)Cancella DB\n6)Esci\n";

30 void main(int argc, char **argv)
{
    con = mysql_init(NULL);

    if (con == NULL) {
        fprintf(stderr, "Initialization error: %s\n", mysql_error(con));
        exit(1);
    }

    int scelta;
    char elim[1];

    while(1){
        puts(scegli);
        puts(menu0);
        puts(inserisciScelta);
        scanf("%d", &scelta);
        fflush(stdin);
        system("clear");
        switch (scelta) {
50
            case 1:
                load_file(&config, "config_gestore.json");
                parse_config();
                gestore();
                break;
55
            case 2:
                load_file(&config, "config_cliente.json");
                parse_config();
                cliente();
                break;
60
            case 3:
                load_file(&config, "config_visitatore.json");
                parse_config();
                visitatore();
                break;
65
            case 4:
                load_file(&config, "config_maschera.json");
                parse_config();
                maschera();
                break;
70
            case 5:
                memset(elim, 0, 1);
                printf("Sicuro di voler eliminare il DB ? (y per andare avanti): ");
                scanf("%os", elim);
                if(strcmp(elim, "y") == 0){

```

```
    load_file(&config, "config_gestore.json");
    parse_config();
    cancella_db();
5    }
    break;
    case 6:
        mysql_close(con);
        return;
    default:
        puts(sceltaScorretta);
        break;
10   }

//tutte le scelte (se inserite correttamente) mi chiudono la connessione, la riapro
if (scelta == 1 || scelta == 2 || scelta == 3 || scelta == 4 || scelta == 5){
15   con = mysql_init(NULL);

    if (con == NULL) {
        fprintf(stderr, "Initialization error: %s\n", mysql_error(con));
        exit(1);
    }
20   }

25 }
```