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

Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

Sistema per la gestione di noleggio di videocassette e DVD

0243651

Pierpaolo Spaziani

**Indice**

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

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

[3. Progettazione concettuale 5](#_Toc2081466291)

[4. Progettazione logica 6](#_Toc2147004904)

[5. Progettazione fisica 8](#_Toc518560220)

[Appendice: Implementazione 9](#_Toc403811585)

# Descrizione del Minimondo

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

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

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

### Specifica disambiguata

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

## Glossario dei Termini

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

## Raggruppamento dei requisiti in insiemi omogenei

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

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

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

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

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

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

# Progettazione concettuale

## Costruzione dello schema E-R

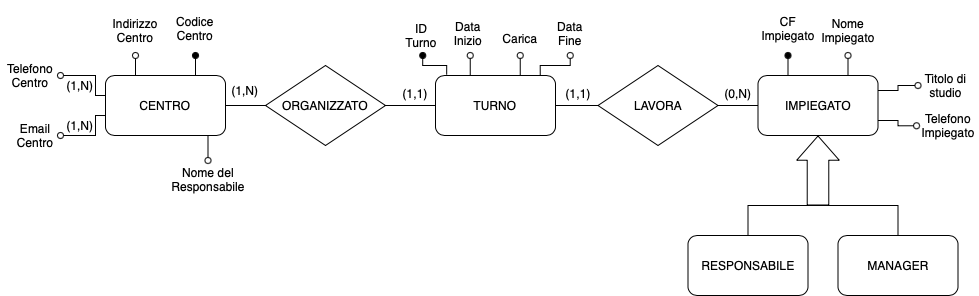
Per lo sviluppo dello schema E-R è stato scelto di utilizzare una strategia bottom-up.

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

isolare e risolvere le ambiguità e dubbi;

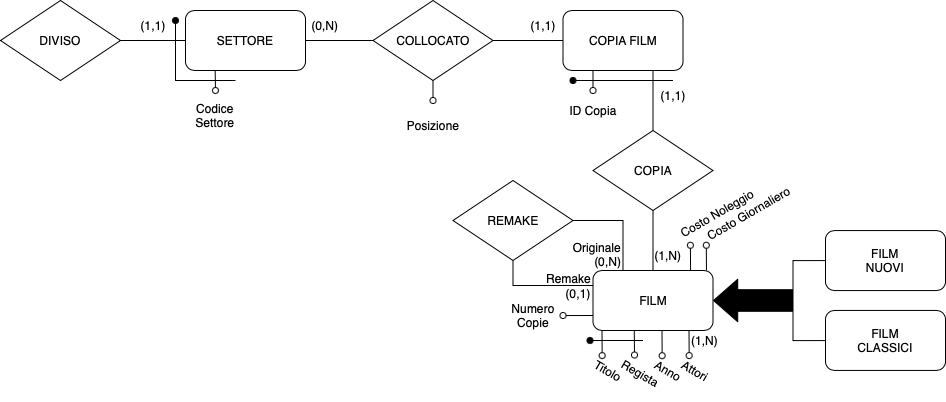
individuare le Entità che avrebbero costituito lo schema coi i relativi attributi;

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

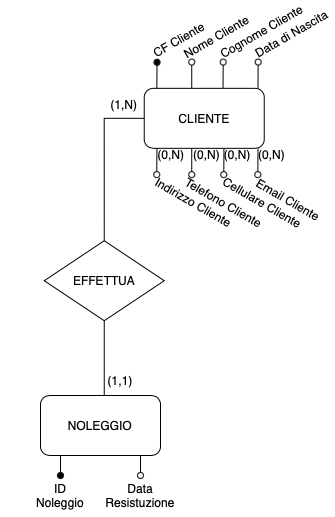
La prima sezione individuata e sviluppata è stata quella riguardante il Centro, gli Impiegati e i Turni:

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

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

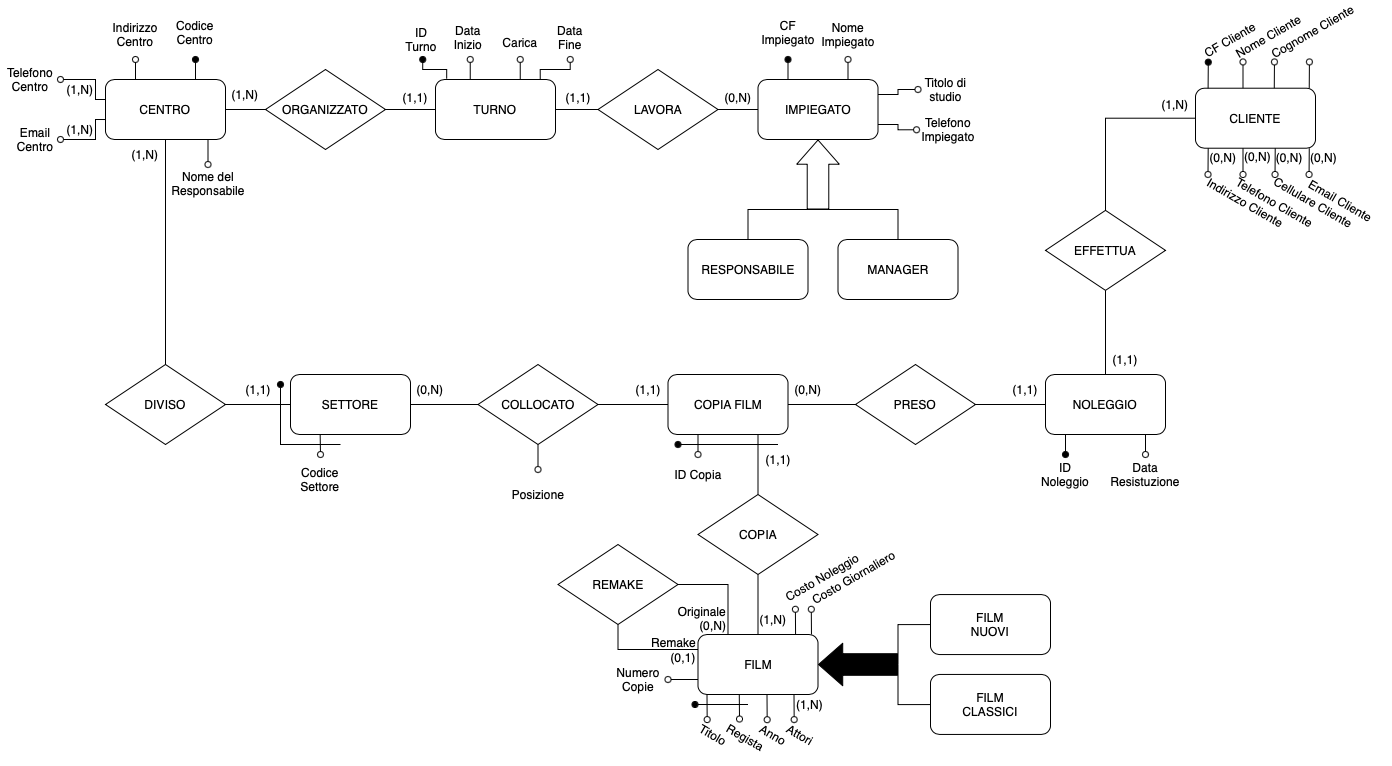
La seconda sezione è quella riguardante i Film ed i Settori:

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

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

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

### Integrazione finale

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

## Regole aziendali

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

## Dizionario dei dati

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

# Progettazione logica

## Volume dei dati

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

## Tavola delle operazioni

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

## Costo delle operazioni

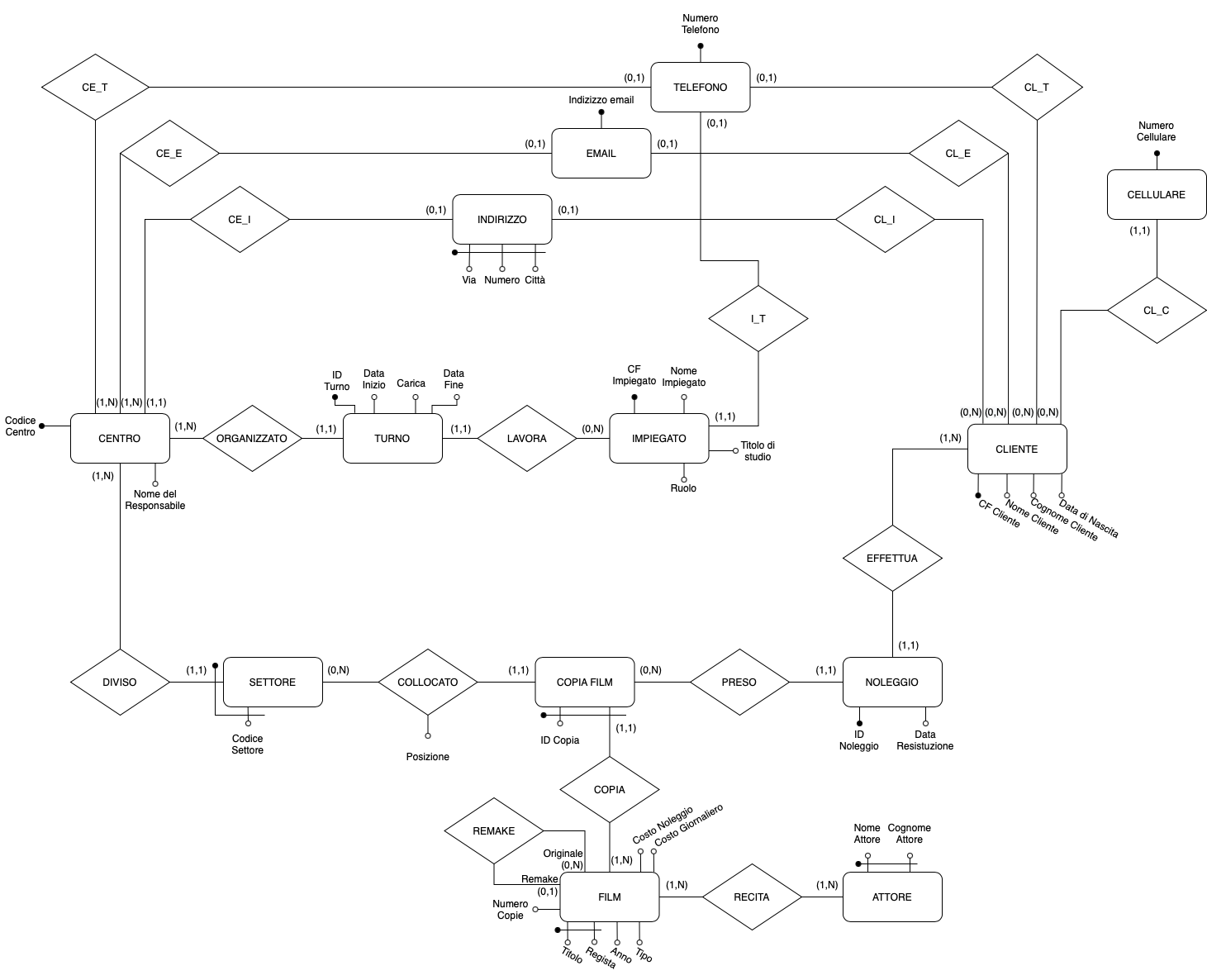
1. 1S Impiegato = 2 \* 100/mese = 200/mese
2. 1L Impiegato + 2S Impiegato + 1L Responsabile+ 2S Responsabile = (1+4+1+4) \* 100/mese = 1.000/mese
3. 1L Turno + 1L Impiegati = (1+1) \* 1.490/mese = 2980/mese
4. 1L Turno + 1L Impiegati = (1+1) \* 1.490/mese = 2980/mese
5. 1L Impiegato + 1S Turno = (1+2)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

## Ristrutturazione dello schema E-R

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

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

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

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

## Trasformazione di attributi e identificatori

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

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

Per le entità deboli sono state scelte le seguenti modifiche:

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

## Traduzione di entità e associazioni

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

Indirizzo.Via → Centro.Indirizzo\_Centro

Indirizzo.Numero → Centro.Numero\_Civico

Indirizzo.Città → Centro.Sede

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

Centro.Codice\_Centro → Turno.Centro\_Tunro

Impiegato.CF\_Impiegato → Turno.Impiegato\_Turno

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

Telefono.Numero\_Telefono → Impiegato.Telefono

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

Centro.Codice\_Centro → Settore.Centro\_Settore

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

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

Film.Titolo → Copia\_Film.Titolo\_Copia

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

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

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

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

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

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

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

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

Film.Titolo → Recita.Titolo\_Recita

Film.Regista → Recita.Regista\_Recita

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

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

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

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

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

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

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

Cliente.CF\_Cliente → Noleggio.Cliente\_Noleggio

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

Cliente.CF\_Cliente → Indirizzo.Cliente\_Indirizzo

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

Centro.Codice\_Centro → Indirizzo.Centro\_Email

Cliente.CF\_Cliente → Email.Cliente\_Email

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

Centro.Codice\_Centro → Indirizzo.Centro\_Telefono

Cliente.CF\_Cliente → Telefono.Cliente\_Telefono

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

Cliente.CF\_Cliente → Cellulare.Cliente\_Cellulare

## Normalizzazione del modello relazionale

**1NF**

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

**2NF**

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

**3NF**

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

Tutte le relazioni dello schema rispecchiano tali caratteristiche.

# Progettazione fisica

## Utenti e privilegi

MANAGER:

Ha tutti i privilegi su tutte le tabelle.

IMPIEGATO:

Centro: SELECT

Impiegato: SELECT

Turno: SELECT

Settore: SELECT

Copia\_Film: SELECT – INSERT – UPDATE – DELETE

Film: SELECT – UPDATE

Remake: SELECT

Recita: SELECT

Attore: SELECT

Noleggio: SELECT – INSERT – UPDATE – DELETE

Cliente: SELECT – INSERT – UPDATE

Indirizzo: SELECT – INSERT – UPDATE – DELETE

Email: SELECT – INSERT – UPDATE – DELETE

Telefono: SELECT – INSERT – UPDATE – DELETE

Cellulare: SELECT – INSERT – UPDATE – DELETE

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

## Strutture di memorizzazione

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## Indici

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

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

## Trigger

## CENTRO

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

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

BEGIN

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

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

end if;

END

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

BEGIN

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

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

end if;

END

## NOLEGGIO

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

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

BEGIN

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

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

end if;

END

## TURNO

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

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

BEGIN

if ((select `Ruolo` from `Impiegato`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

end if;

END

## Eventi

## Viste

## Stored Procedures e transazioni

**aggiungi\_attore\_recita**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

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

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

end if;

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

commit;

END

**aggiungi\_cellulare**

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

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

BEGIN

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

END

**aggiungi\_cliente**

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

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

BEGIN

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

END

**aggiungi\_copia**

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

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

BEGIN

declare num int;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

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

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

order by `ID\_Copia` desc

LIMIT 1 into num;

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

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

where `Titolo` = titolo

and `Regista` = regista;

commit;

END

**aggiungi\_email**

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

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

BEGIN

if tipo is true then

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

else

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

end if;

END

**aggiungi\_film**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read uncommitted;

start transaction;

if nuovo is true then

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

else

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

end if;

if remake is true then

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

end if;

commit;

END

**aggiungi\_impiegato**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read uncommitted;

start transaction;

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

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

commit;

END

**aggiungi\_indirizzo**

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

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

BEGIN

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

END

**aggiungi\_telefono**

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

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

BEGIN

if tipo is true then

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

else

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

end if;

END

**aggiungi\_turno**

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

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

BEGIN

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

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

END

**cambia\_responsabile**

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

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

BEGIN

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

END

**dequalifica\_film**

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

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

BEGIN

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

END

**elimina\_cellulare**

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

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

BEGIN

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

END

**elimina\_email**

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

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

BEGIN

if tipo is true then

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

else

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

end if;

END

**elimina\_indirizzo**

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

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

BEGIN

delete from `Indirizzo`

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

END

**elimina\_telefono**

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

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

BEGIN

if tipo is true then

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

else

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

end if;

END

**film\_disponibili\_noleggio**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

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

`Titolo\_Copia` as `Film`,

`Regista\_Copia` as `Movie Director`,

`Anno` as `Year`,

`Settore\_Copia` as `Sector`,

`Posizione` as `Position`

from `Copia\_Film` left join `Noleggio`

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

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

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

join `Film`

on `Titolo` = `Titolo\_Copia`

and `Regista` = `Regista\_Copia`

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

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

commit;

END

**info\_cliente**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

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

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

end if;

select

`CF\_Cliente` as `Tax Code`,

`Nome\_Cliente` as `Name`,

`Cognome\_Cliente` as `Surname`,

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

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

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

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

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

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

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

commit;

END

**login**

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

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

BEGIN

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

select `Ruolo`

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

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

if user\_role = 'Impiegato' then

set var\_role = 1;

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

elseif user\_role = 'Manager' then

set var\_role = 2;

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

else

set var\_role = 3;

end if;

END

**noleggia**

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

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

BEGIN

declare id int;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

select `ID\_Copia`

from `Copia\_Film` left join `Noleggio`

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

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

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

where `Centro\_Copia` = centro

and `Titolo\_Copia` = var\_film

and `Regista\_Copia` = var\_regista

and `ID\_Noleggio` is null

LIMIT 1 into id;

if id is null then

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

else

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

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

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

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

end if;

commit;

END

**noleggi\_cliente**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

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

`Titolo\_Noleggio` as `Film`,

`Regista\_Noleggio` as `Movie Director`,

`Data\_restituzione` as `Return Date`

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

commit;

END

**noleggi\_scaduti**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

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

`Titolo\_Noleggio` as `Film`,

`Data\_Restituzione` as `Return Date`,

`Cliente\_Noleggio` as `Client`

from `Noleggio` join `Copia\_Film`

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

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

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

where `Data\_Restituzione` < curdate()

and `Centro\_Copia` = centro

order by `Data\_Restituzione` asc;

commit;

END

**report\_annuale**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

drop temporary table if exists `intervallo`;

create temporary table

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

set transaction isolation level serializable;

start transaction;

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

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

end if;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

`Data\_Inizio`,

`Data\_Fine`,

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

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

from `Turno` where

`Impiegato\_Turno` = cf

and year(`Data\_Inizio`) = anno

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

insert into `intervallo`

select `Centro\_Turno`,

`Carica`, `Data\_Inizio`,

last\_day(`Data\_Inizio`),

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

`Data\_Inizio`)+1,

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

from `Turno` where

`Impiegato\_Turno` = cf

and year(`Data\_Inizio`) = anno

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

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

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

`Data\_Fine`,

day(`Data\_Fine`),

day(`Data\_Fine`)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and year(`Data\_Fine`) = anno

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

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

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

commit;

drop temporary table `intervallo`;

END

**report\_mensile**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

drop temporary table if exists `intervallo`;

create temporary table

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

set transaction isolation level serializable;

start transaction;

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

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

end if;

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

`Data\_Inizio`,

`Data\_Fine`,

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

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

from `Turno` where

`Impiegato\_Turno` = cf

and month(`Data\_Inizio`) = mese

and year(`Data\_Inizio`) = anno

and month(`Data\_Fine`) = mese

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

insert into `intervallo`

select `Centro\_Turno`,

`Carica`, `Data\_Inizio`,

last\_day(`Data\_Inizio`),

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

`Data\_Inizio`)+1,

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

from `Turno` where

`Impiegato\_Turno` = cf

and month(`Data\_Inizio`) = mese

and year(`Data\_Inizio`) = anno

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

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

insert into `intervallo`

select `Centro\_Turno`,

`Carica`,

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

`Data\_Fine`,

day(`Data\_Fine`),

day(`Data\_Fine`)\*8

from `Turno` where

`Impiegato\_Turno` = cf

and month(`Data\_Fine`) = mese

and year(`Data\_Fine`) = anno

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

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

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

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

commit;

drop temporary table `intervallo`;

END

**restituisci\_film**

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

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

BEGIN

declare var\_data date;

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction isolation level read committed;

start transaction;

set prezzo\_extra = 0;

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

if (var\_data < curdate()) then

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

end if;

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

commit;

END

**turni\_del\_giorno**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

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

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

where `Data\_Inizio` <= giorno

and `Data\_Fine` >= giorno

and `Centro\_Turno` = centro;

commit;

END

**visualizza\_attori\_film**

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

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

BEGIN

declare exit handler for sqlexception

begin

rollback;

resignal;

end;

set transaction read only;

set transaction isolation level read committed;

start transaction;

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

`Cognome\_Attore\_Recita` as `Surname`

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

commit;

END

1. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-1)