Set Cinematografico

creare il file .sql

```
mysqldump -u studente -p cinema > cinema.sql
```

Struttura del progetto

La presente documentazione tratta nel dettaglio la progettazione e l'implementazione dell'elaborato "Set cinematografico" di Michele Nardini() Santoro Matteo(0000881608) Manuel Luzietti()

Struttura:

Introduzione

Il gruppo si pone come obbiettivo quello di realizzare un database per la gestione
di un set cinematografico coprendone tutti gli aspetti.
Saranno memorizzate all'interno del database le varie figure che partecipano
alle realizzazione di un girato, gli enti che si occuperanno della distribuzione
la gesione delle scene e dei ciak quindi gli stipendi e gli incassi.

2 Analisi requisiti

2.11 Intervista

/* SERVE AGGIUNGERE LE QUERY IN QUESTA ROBA /

Netflix Italia possiede un sistema software da per la gestione delle persone e cose coinvolte della realizzazione di serie Tv e film, tale sistema necessità
di una nuova versione. Si richiede quindi la progettazione di una database che
possa gestire gli aspetti fondamentali per girare uno seceneggiato, il database
sarà utilizzato da persone che si occuperanno della gestione.
Saranno memorizzate tutte le informazioni sui membri della troupe che lavorano
alla realizzazione di un film, il quale sarà il fulcro attorno al quale gira l'
applicativo, di cui memorizzeremo le principali informazioni e al quale uniremo
un eventuale serie letterarie da cui è tratto. Sarà fondamentale anche l'ente che
si occuperà della distribuzione e gli incassi che verranno generati. Come accade spesso in questo ambiente, per poter esistere acluni film vengono
stanziati dei finanziamenti che possono provenire principalmente da sponsor o
finanziatori, ossia persone che possiedono un enorme capitale che effettuano
un investimento nella riuscita di un film e che dal quale ricaveranno un guadagno
che è calcolato in percentuale al guadagno del film. Le figure professionali che
routano attorno la creazione di un girato sono fondamentali e vanno inserite all'
interno dell'applicativo tramite un'interfaccia di inserimento, ruoli come
sceneggiatore, produttore(con annessa percentuale guadagno), produttore esecutivo ,aiuto regista, capo regista (con annessa percentuale guadagno),regista, attore, stilista, operatore fonico e operatore fotografico e di loro
memorizzare l'anagrafica, telefono e IBAN che sarà poi utilizzato per l'accredito
della paga mensile calcolata nel software, che dovrà poi contenere uno storico
delle buste paga. Si dovrà poi indicare l'indirizzo di residenza dei vari membri

della troupe per il loro recapito, gli attori utilizzeranno costumi progrettati
da stilisti che vengono conservati in magazzini assieme agli oggetti di scena.
Saranno infine regsitrati nell'applicativo i ciak presi durante le riprese del
film, nello specifico il rullo su cui è impresso, il numero del ciak, la location
nel quale viene girato e i costumi ed oggetti di scena presenti nel ciak.
/* l'applicativo deve quindi anche poter ordinare gli oggetti e i costumi temporalmente utilizzati nelle riprese*/

2.2 Estrazione Concetti Fondamentali (va cambiato se cambiamo il paragrafo sopra)

Netflix Italia possiede un sistema software da per la gestione delle persone e
cose coinvolte della realizzazione di serie Tv e **film**, tale sistema necessità
di una nuova versione. Si richiede quindi la progettazione di una database che
possa gestire gli aspetti fondamentali per girare uno seceneggiato, il database
sarà utilizzato da persone che si occuperanno della gestione.
Saranno memorizzate tutte le informazioni sui **membri della troupe** che lavorano
alla realizzazione di un film, il quale sarà il fulcro attorno al quale gira l'
applicativo, di cui memorizzeremo le principali informazioni e al quale uniremo
un eventuale **serie letterarie** da cui è tratto. Sarà fondamentale anche l'**ente** che
si occuperà della **distribuzione** e gli **incassi** che verranno generati. Come accade spesso in questo ambiente,
per poter esistere acluni film vengono
stanziati dei **finanziamenti** che possono provenire principalmente da **sponsor** o
finanziatori, ossia persone che possiedono un enorme capitale che effettuano
un investimento nella riuscita di un film e che dal quale ricaveranno un **guadagno**
che è calcolato in percentuale al guadagno del film. Le figure professionali che
routano attorno la creazione di un girato sono fondamentali e vanno inserite all'
interno dell'applicativo tramite un'interfaccia di inserimento, ruoli come
sceneggiatore, produttore(con annessa percentuale guadagno), produttore esecutivo ,aiuto regista, capo regista (con annessa percentuale guadagno),regista, attore, stilista, operatore fonico e operatore fotografico e di loro
memorizzare l'anagrafica, telefono e IBAN che sarà poi utilizzato per l'accredito
della **paga mensile** calcolata nel software, che dovrà poi contenere uno storico
delle **buste paga**. Si dovrà poi indicare l'**indirizzo** di residenza dei vari membri
della troupe per il loro recapito, gli attori utilizzeranno **costumi** progrettati
da **stilisti** che vengono conservati in **magazzini** assieme agli **oggetti di scena**.
Saranno infine regsitrati nell'applicativo i **ciak** presi durante le riprese del
film, nello specifico il rullo su cui è impresso, il numero del ciak, la location
nel quale viene girato e i costumi ed oggetti di scena presenti nel ciak.
/* l'applicativo deve quindi anche poter ordinare gli oggetti e i costumi temporalmente utilizzati nelle riprese*/

3 Progetto dello schema concettuale

Lo schema concettuale è stato realizzato attorno ad alcune entità di base come
i membri della troupe dai quali deriviamo i vari lavori e maestranze e l'entità

scena-Ciak fondamentale per la realizzazione di un film

3.1 Distribuzione e Incasso

Ogni Film ha bisogno di un ente specifico che si occupi della distribuzione,
come ad esempio l'azienda UCI che dopo aver comprato i diritti per la riproduzione
porta nelle sue sedi la pellicola, ogni ente da noi gestito attraverso
un entita composta da nome, indirizzo, P.IVA ha poi una dislocazione locale.

3.2 Inspirazione e Sceneggiatura

E' comune che i film siano stati inspirati da alcune serie letterarie, come ad
esempio la saga di Harry Potter, abbiamo quindi deciso di inserire la possibilità
che un dato film sia tratto da una Serie Letteraria che abbiamo modellato come
un entita, spesso capita nell' ambiente cinematografico che lo Scenegiatore
sia anche l'autore della Serie

3.3 Gestione Fondi

Un film per poter essere realizzato ha un enorme bisogno di soldi quindi di fondi
dato che i costi sono molti, spesso questo patrimonio si crea dagli investimenti
che la pellicola riesce a raccogliere, abbiamo creato le entità Sponsor che
rappresentano le possibili aziende che possono in cambio di una sponsorizzazione
del loro prodotto pagare una somma di denaro e l' entità Finanziatore per la
persona o azienda che decidono di investire una somma di denaro per poi poter
guadagnare grazie ad una percentuale una volta ricevuti gli incassi del film tramite un associazione che ha come
attributo la data di investimento e il Produttore esecutivo abbiamo modellato la gestione dei fondi. A livello
concettuale avremo una gerarchia con entità padre il fondo ed entità figlie
i corrispettivi fondo_sponsor e fondo_finanziatore che erediteranno gli attributi
di fondo con la chiave esterna di sponsor o finanziatore.

3.4 Membro della Troupe

In un set cinematografico le persone che lavorano alle realizzazione di un film
sono tante e compiono lavori diversi, ma abbiamo deciso di utilizzare una
gerarchia per poter più comodamente rappresentarle.

3.5 Stipendio

Ogni addetto ai lavori in un set cinematografico ha uno stipendio, modellato
attraverso un entità nella quale vengono registrati i codici singoli per busta

paga, le ore accumulate sul set, i contributi e il periodo sul quale poi verrà calcolato un vero e proprio stipendio.

3.6 Scena

Schema Completo

4 Il progetto Logico

Qui riportati gli schemi di navigazione delle operazioni

4.1 Traduzione delle operazioni in query

4.11 Aggiunta Film

Aggiungere un film consiste nell'aggiungere un instanza dell' entità FILM

```
INSERT IGNORE INTO Film(codF, titolo, genere, durata, dataUscita, idSerie)

VALUES (00001, 'Star Wars: Episodio VI - Il ritorno dello Jedi',
    'fantascienza', 134, 25/05/2983, 2551983);
```

4.12 Aggiunta Membro Troupe

Aggiungiamo prima al database un membro generico, il quale verra poi collegato al film tramite la tupla Film_Membro_Troupe che rappresenta l'associazione *lavora* e ad il lavoro che svolgerà all' interno di questo progetto con una tupla della tabella RuoloMembroTroupe

```
INSERT IGNORE INTO MembroTroupe(CF, nome, cognome, iban, dataNascita, telefono,
                            codInd, percentualeContributo)
    -- produttore esecutivo George Lucas
    VALUES ('GRGLCS14ES44', 'George', 'Lucas', 'GB98BARC20040156884556',
                '1944/05/14', '516-527-8719', 24673, 1.0);

INSERT IGNORE INTO RuoloMembroTroupe(CF, nomeRuolo)
    VALUES ('GRGLCS14ES44', 'Produttore Esecutivo');

INSERT IGNORE INTO Film_Membro_Troupe(codF, CF)
    -- George lucas
    VALUES (1, 'GRGLCS14ES44'),
```

4.13 Distribuzione

Avendo precedentemente inserito un Indirizzo e usando quel codice, creiamo l'ente che si occuperà della distribuzione del film, rappresentando quindi l'associazione e l'entita ente
con la sua eventuale sedeTerritoriale.

```
INSERT IGNORE INTO Enti(P_IVA,nome,codInd)
    -- ente di distribuizone
    VALUES (40365320379, "UCI Milano", 18302);

INSERT IGNORE INTO Indirizzo(codInd, citta, via, civico, CAP)
    --sede territoriale uci savignano
    VALUES (16395, 'savignano', 'Piazza Metropolis', 18, 47039),

INSERT IGNORE INTO SediTerritoriali(P_IVA, codInd)
    VALUES (40365320379, 16395);
```

4.14 Costumi e Magazzini

Per poter procedere a realizzare i concetti dell'utlizzo e storage dei costumi si può inserire l'entita Magazzino, le associazioni riportate nell ER collocazioni_costumi e assegnamento attore vengono formalizzate utilizzando un Entità Posizione Magazzino che utlizza la primary key di Magazzino che viene legata a Costume con la foreign key codP e la seconda associazione assegnando ad un costume una foreign key CF (codice fiscale) che viene importato da MembroTroupe.

```
INSERT IGNORE INTO Magazzino(numMagazzino, codInd)
    -- magazzino principale
    VALUES(1,19341);

INSERT IGNORE INTO Costume(codC, tipo, descrizione, CF, codP)
    -- costume Luke
    VALUES(49262 ,'fantasia','Costume di Luke Skywalker, kimono nero,
     stivali, spada Laser Verde','MRKHML25IS51', 25588);

INSERT IGNORE INTO PosizioneMagazzino(codP,numMagazzino,scaffale, percorso)
    -- Posizione costume luke skywalker
VALUES(25588, 1, 2, 'S');
```

4.15 Sponsor, Finanziatori e Fondo

Per poter procedere alla realizzazione di un film, quest'ultimo avrà bisogno di una serie di fondi che andranno a contribuire alle spese. Per fare ciò inseriremo prima una serie di Sponsor e Finanziatori, seguiti da una entità fondo aventi due foreign key associate opzionalmente o allo sponsor o al finanziatore(ipotizzando che un fondo può essere o di uno sponsor o di un finanziatore).

```
INSERT IGNORE INTO Sponsor(P_IVA_SPONSOR, nome)
    -- Samsung
    VALUES('53179880082', 'Samsung Galaxy');

INSERT IGNORE INTO Finanziatore(P_IVA_FINANZIATORE, nome,codInd
,percentualeGuadagno)
    -- Finanziatore 1
    VALUES('31562270996', 'Micheal Barnaby', 49429, 1.8);

INSERT IGNORE INTO Fondo(codFondo, dataAccredito, patrimonio, P_IVA_SPONSOR,
P_IVA_FINANZIATORE,codF)
    -- george lucas ha contribuito al fondo
    VALUES (1, '1977-01-05', 2000000, '53179880082', null, 1);
```

4.16 Retribuzione Membro Troupe

Avendo creato in precedenza un membro troupe, ora potremo creare le varie busta paga per quel determinato membro troupe usando una entità busta paga contentente tutte le informazioni dello stipendio del dipendente e una entità retribuzione che avrà lo scopo di associare un membro troupe alla propria busta paga

```
INSERT IGNORE INTO MembroTroupe(CF, nome, cognome, iban, dataNascita, telefono,
                            codInd, percentualeContributo)
    -- produttore esecutivo George Lucas
    VALUES ('MRKHML25IS51', 'Mark', 'Hamill', 'GB56BARC20038438638758',
                '1951/09/25', '214-989-5138', 37065, NULL);

INSERT IGNORE INTO BustaPaga(codB, retribuzioneOraria, oreLavorate, mese)
    -- luke skywalker
    VALUES (47918, 32, 111.2, 'aprile');

INSERT IGNORE INTO Retribuzione(CF, CodB)
    -- luke
    VALUES  ('MRKHML25IS51', 47918);
```

5 Specifiche funzionali

Ora andremo a svolgere le seguenti operazioni

5.1 Stipendio membri della troupe

Query creata per poter calcolare lo stipendio della troupe per un mese, ottenuta collezionando le busta paga dei lavoratori, in altre parole il costo della troupe mensile, il tutto ottenuto selezionando il Film per il quale si desidera conoscere la cifra.

```
        select @stipendi :=  sum(retribuzioneOraria * oreLavorate) as
CostoTroupe_Mese from
        BustaPaga b join Retribuzione r on (b.codB = r.codB )
        join Film_Membro_Troupe flm on (r.CF = flm.CF)  where mese = ?
        and codF = ?;
```

5.2 Elenco oggetti acquistati in magazzino

Trova la posizione in un dato magazzino di tutti gli oggetti acquistati specificandone magazzino.

```
    select ods.tipo, ods.descrizione, pm.scaffale, pm.percorso
    from PosizioneMagazzino pm, OggettiDiScena ods
    where pm.codP = ods.codP
    and pm.numMagazzino = ?;
```

5.3 Profitto finanziatori

Query creata poter calcolare il profitto ottenuto dai finanziatori grazie al finanziamento di uno specifico film. Tale profitto è calcolato anche in base alla percentuale di guadagno stabilita.

```
    select @Denaro := sum(incasso) as money FROM Incasso;
    select distinct F.nome, F.percentualeGuadagno, (F.percentualeGuadagno / 100 *
@Denaro ) as guadagno
    from Finanziatore F join Fondo ff on (F.P_IVA_FINANZIATORE =
ff.P_IVA_FINANZIATORE)
    where F.percentualeGuadagno is not null
    and codF = ?
    order by guadagno DESC;
```

5.4 Costumi da usare per scena

Query creata per definire i costumi che saranno da usare nella realizzazione di una determinate scena e che dovrà indossare un attore X

```
    select c.descrizione, c.tipo
    from ScenaCiak sc join CostumeScena cs on (cs.codScena=sc.codScena)
    join Costume c on (c.codC = cs.codC)
    where sc.noteDiProduzione = ?;
```

5.5 Dipendenti in scena

```
select mt.*
from ScenaCiak sc join MembroTroupeScena mts on (sc.codScena = mts.codScena)
join MemtroTroupe mt on (mts.CF = mt.CF)
where sc.codScena = ?;
```

5.6 Oggetti in scena

```
select ods.*
from ScenaCiak sc join OggettoScena os on (sc.codScena=os.codScena)
join OggettiDiScena ods on (os.codO=ods.codO)
where sc.codScena=?;
```

5.7 Stipendio netto dipendente

Definisce lo stipendio di un determinato membro della troupe in un dato mese, calcolandolo moltiplicando la retribuzione oraria stabilita mediante contratto e le ore lavorate

```
select @stipendio := sum(retribuzioneOraria * oreLavorate) as Stipendio
from BustaPaga bp join Retribuzione r on (bp.codB = r.codB)
where r.CF = ?;
```

5.8 Profitto produttori

Server per poter calcolare quanti soldi riescono a ricavare dalla produzione di un film i produttori
e chiunque lavori all'interno e riceva una percentuale dagli incassi.

```
  select @Denaro := sum(incasso) as money FROM Incasso;
  select distinct M.nome, M.cognome,M.percentualeContributo,
(M.percentualeContributo / 100 * @Denaro ) as guadagno, Rm.nomeRuolo
  from Incasso I, MembroTroupe M, RuoloMembroTroupe Rm join Film_Membro_Troupe flm
on (Rm.CF = flm.CF)
  where (M.CF = Rm.CF) and flm.codF = ? and M.percentualeContributo is not null
  order by guadagno DESC;
```

6 Il Progetto Logico

6.1 Frequenza e costo degli accessi

6.2 Volume dati del database

Riportiamo in seguito la tabella dei volumi in cui sono elencante tutte le entità, le relazioni e i relativi volumi.

Nome	Tipo	Volume
Serie Letteraria	E	300
Film	E	500
Indirizzo	E	2000
Ente	E	10
Sede Territoriale	E	500
Fondo	E	5000
Sponsor	E	100
Finanziatore	E	100
Membro Troupe	E	5000
Ruolo	E	10
Busta Paga	E	30000
Scena Ciak	E	10000
Magazzino	E	1
Posizione Magazzino	E	500
Costume	E	1000
Oggetto di Scena	E	1000
Acquisto	E	1000
Ditta	E	50
Distribuzione	R	5000
Incasso	R	10000
Ruolo Membro Troupe	R	6000
Retribuzione	R	30000
Film Membro Troupe	R	10000
Membro Troupe Scena	R	100000
Stilista Costume	R	500
Costume Scena	R	50000
Oggetto Scena	R	50000
Acquisto costume	R	500
Acquisto oggetto	R	1000

6.3 Tabella degli accessi

6.4 Traduzione delle entita

- Film(***codF***, titolo, genere ,durata, dataUscita, idSerie[0-1])
    - FK: idSerie REFERENCES **Serie_Letteraria**

- Indirizzo(***codInd***, città, via, civico, CAP)
- Ente(***P.IVA_ENTE***,nome,codInd)
    - FK: codInd REFERENCES **Indirizzo**
- Distribuzione(***P.IVA***, ***codF***)
    - FK: P.IVA_ENTE REFERENCES **Ente**
    - FK: codF REFERENCES **Film**
- Sedi_Territoriali(***codInd***, P.IVA_ENTE)
    - FK: codInd REFERENCES **Indirizzo**
    - FK: P.IVA_ENTE REFERENCES **Ente**
- Incasso(dataInizio, dataFine, incasso, ***codF***, ***codInd***)
    - FK: codF REFERENCES **Film**
    - FK: codInd REFERENCES **Indirizzo**
- Fondo(***codFondo***, dataAccredito, patrimonio, P_IVA_SPONSOR[0, 1], P.IVA_FINANZIATORE[0, 1], CF, codF)
    - FK: P.IVA_SPONSOR REFERENCES **Sponsor**
    - FK: P.IVA_FINANZIATORE REFERENCES **Finanziatore**
    - FK: CF REFERENCES **Membro_troupe**
    - FK: codF REFERENCES **Film**
- Sponsor(***P.IVA_SPONSOR***, nome)
- Finanziatore(***P.IVA_FINANZIATORE***, nome, codInd, percentuale_guadagno)
    - FK: codInd REFERENCES **Indirizzo**
- Serie_Letteraria(***idSerie***, titolo, genere, volumi, CF)
    - FK: CF REFERENCES **Membro_troupe**
- Membro_Troupe(***CF***, nome, cognome, IBAN, dataNascita, telefono, codInd, percentuale_contributo[0-1], ruolo, tipoOperatore[0-1])
    - FK: codInd REFERENCES **Indirizzo**
- Ruolo(***nomeRuolo***)
- RuoloMembroTroupe(***CF***, ***nomeRuolo***)
    - FK: CF REFERENCES **Membro_troupe**
    - FK: nomeRuolo REFERENCES **Ruolo**
- BustaPaga(***codB***, retribuzioneOraria, oreLavorate, mese)
- Retribuzione(***CF***, ***codB***)

    - FK: CF REFERENCES **Membro_Troupe**
    - FK: codB REFERENCES **BustaPaga**

- Film_Membro_Troupe(***codF***, ***CF***)

    - FK: codF REFERENCES **Film**
    - FK: CF REFERENCES **Membro_troupe**

- Supervisione(Supervisore, ***Subalterno***)

    - FK: Supervisore REFERENCES **Membro_troupe**
    - FK: Subalterno REFERENCES **Membro_troupe**

- ScenaCiak(***codScena***, note_di_produzione, rullo, numRiprese, durataOre, costo_affitto_giornaliero[0-1], codInd, codF)

    - FK: codInd REFERENCES **Indirizzo**
    - FK: codF REFERENCES **Film**

- MembroTroupeScena(***codScena***, ***CF***)

    - FK: codScena REFERENCES **Scena-Ciak**
    - FK: CF REFERENCES **Membro-Troupe**

- Magazzino(***numMagazzino***, codInd)

    - FK: codInd REFERENCES **Indirizzo**

- PosizioneMagazzino(***codP***, ***numMagazzino***, scaffale, percorso)

    - FK: numMagazzino REFERENCES **Magazzino**

- Costume(***codC***, tipo, descrizione, CF, codP)

    - FK: CF REFERENCES **Membro_troupe**
    - FK: codP REFERENCES **PosizioneMagazzino**

- Stilista_Costume(***CF***, ***codC***)

    - FK: CF REFERENCES **Membro_troupe**
    - FK: codC REFERENCES **Costume**

- Costume_Scena(***codC***, ***codScena***)

    - FK: codC REFERENCES **Costume**
    - FK: codScena REFERENCES **Scena-Ciak**

- Oggetti_di_scena(***codO***, tipo, descrizione, codP)

    - FK: numMagazzino REFERENCES **PosizioneMagazzino**

- Oggetto_Scena(***codO***, ***codScena***)

    - FK: codO REFERENCES **Oggetto_di_scena**

    - FK: codScena REFERENCES **ScenaCiak**

- Acquisto(***idAcquisto***, data, prezzoTotale, P.IVA_DITTA)
    - FK: P.IVA_DITTA REFERENCES **Ditta**
- Acquisto_Costume(***codC***, ***idAcquisto***, prezzo)
    - FK: codC REFERENCES **Costume**
    - FK: idAcquisto REFERENCES **Acquisto**
- Acquisto_Oggetto(***codO***, ***idAcquisto***, prezzo)
    - FK: codO REFERENCES **Oggetto_di_scena**
    - FK: idAcquisto REFERENCES **Acquisto**
- Ditta(***P.IVA_DITTA***, nome,codInd)
    - FK: codInd REFERENCES **Indirizzo**

6.5 Creazione delle tables

```
CREATE TABLE if not exists Indirizzo(
        codInd int primary key,
        citta varchar(21) NOT NULL,
    via varchar(40) NOT NULL,
    civico int NOT NULL,
    CAP int NOT NULL check(length(CAP) = 5)
    );

CREATE TABLE IF NOT EXISTS  Enti(
  P_IVA varchar(11) NOT NULL,
  nome varchar(40) NOT NULL,
  codInd INT NOT NULL,
  FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
  ON DELETE CASCADE
  ON UPDATE NO ACTION,
  PRIMARY KEY(P_IVA)
) ;

CREATE TABLE if not exists MembroTroupe(
        CF varchar(12) primary key,
    nome varchar(20) NOT NULL,
    cognome varchar(20) NOT NULL,
    IBAN varchar(30) NOT NULL,
    dataNascita date NOT NULL,
    telefono varchar(15) NOT NULL,
    codInd INT NOT NULL,
    FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    percentualeContributo float(3) check(percentualeContributo between 0  and
```

```
100),
    ruolo varchar(41) NOT NULL check (ruolo in ('sceneggiatore', 'produttore',
    'produttore esecutivo','aiuto regista', 'capo regista',
    'regista', 'attore', 'stilista', 'operatore')),
    tipoOperatore varchar(31) check (tipoOperatore in ('fonico', 'fotografico'))
);


CREATE TABLE IF NOT EXISTS SerieLetteraria(
    idSerie varchar(11) primary key,
    titolo varchar(51) NOT NULL,
    genere varchar(40) NOT NULL,
    CF  varchar(12) NOT NULL,
        FOREIGN KEY (CF) REFERENCES MembroTroupe(CF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );

CREATE TABLE IF NOT EXISTS Film (
  codF int NOT NULL AUTO_INCREMENT,
  titolo varchar(51) NOT NULL,
  genere varchar(40) NOT NULL CHECK (genere = 'Animazione' or genere = 'Avventura'
or genere = 'Azione' or genere = 'Biografico' or genere = 'Commedia' or genere =
'Documentario' or genere = 'Drammatico' or genere = 'Pornografico' or genere =
'Fantascienza' or genere = 'Fantasy' or genere = 'Guerra' or genere = 'Horror' or
genere = 'Musical' or genere = 'Storico' or genere = 'Thriller' or genere =
'Western'),
  durata int NOT NULL CHECK (durata > 0),
  dataUscita date DEFAULT NULL,
  idSerie varchar(11),
  FOREIGN KEY (idSerie) REFERENCES SerieLetteraria(idSerie)
  ON DELETE CASCADE
  ON UPDATE NO ACTION,
  PRIMARY KEY (codF)
);


CREATE TABLE IF NOT EXISTS SediTerritoriali (
  P_IVA varchar(11) NOT NULL,
  FOREIGN KEY (P_IVA) REFERENCES Enti (P_IVA)
  ON DELETE CASCADE
  ON UPDATE NO ACTION,
  codInd INT NOT NULL,
  FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
  ON DELETE CASCADE
  ON UPDATE NO ACTION,
  PRIMARY KEY (codInd)
);


CREATE TABLE if not exists Distribuzione(
        P_IVA varchar(11) ,
        FOREIGN KEY (P_IVA) REFERENCES Enti(P_IVA)
        ON DELETE CASCADE
```

```
        ON UPDATE NO ACTION,
        codF INT NOT NULL,
        FOREIGN KEY (codF) REFERENCES Film(codF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    PRIMARY KEY(P_IVA, codF)
    );


CREATE TABLE if not exists Incasso(
        percentualeTrattenute float(3) NOT NULL,
    codF INT NOT NULL,
        FOREIGN KEY (codF) REFERENCES Film(codF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    codInd INT NOT NULL,
    FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    primary key(codF, codInd)
    );

-- incasso settimanale non ci piace....

CREATE TABLE if not exists Sponsor(
        P_IVA_SPONSOR varchar(11) primary key,
        nome varchar(41) NOT NULL
    );


CREATE TABLE if not exists Finanziatore(
        P_IVA_FINANZIATORE varchar(11) primary key,
    nome varchar(41) NOT NULL,
     codInd INT NOT NULL,
    FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    percentualeGuadagno float(3) NOT NULL CHECK(percentualeGuadagno between 0 and
100)
    );




CREATE TABLE if not exists Fondo(
        codFondo varchar(13) primary key,
    dataAccredito date NOT NULL,
    patrimonio float(14) NOT NULL CHECK(patrimonio >= 0),
    P_IVA_SPONSOR varchar(11) NOT NULL,
    FOREIGN KEY (P_IVA_SPONSOR) REFERENCES Sponsor(P_IVA_SPONSOR)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    P_IVA_FINANZIATORE varchar(11) NOT NULL,
    FOREIGN KEY (P_IVA_FINANZIATORE) REFERENCES Finanziatore(P_IVA_FINANZIATORE)
```

```
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    CF varchar(12) NOT NULL,
        FOREIGN KEY (CF) REFERENCES MembroTroupe(CF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    codF INT NOT NULL,
        FOREIGN KEY (codF) REFERENCES Film(codF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
    );



CREATE TABLE if not exists Film_Membro_Troupe(
        codF INT NOT NULL,
        FOREIGN KEY (codF) REFERENCES Film(codF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        CF varchar(12) NOT NULL,
        FOREIGN KEY (CF) REFERENCES MembroTroupe(CF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        PRIMARY KEY(codF, CF)
        );

-- CREATE TABLE if not exists Supervisione(
-- supervisore INT NOT NULL,
-- FOREIGN KEY (supervisore) REFERENCES Supervisione(supervisore)
-- ON DELETE CASCADE
-- ON UPDATE NO ACTION,
-- subalterno INT NOT NULL,
--  FOREIGN KEY (subalterno) REFERENCES Supervisione(subalterno)
-- ON DELETE CASCADE
-- ON UPDATE NO ACTION,
-- PRIMARY KEY(subalterno)
-- );

CREATE TABLE if not exists ScenaCiak(
        -- TODO HO CAMBIATO GLI ATTRIBUTI dataInizio e Fine, trovare un sostituto
valido e cambiare il resto nell ER e relazione....
        --scommentando le date non da errore la query, mahhh .....
        codScena int primary key,
        noteDiProduzione varchar(255),
        rullo int NOT NULL,
        numRiprese int NOT NULL,
        -- dataInizio date,
        -- dataFine date, -- CHECK(dataFine >= dataInizio),

        -- sostituiro le date con il numero di ore
        durataOre float NOT NULL,
        costoAffittoGiornaliero float(5),
        codInd INT NOT NULL,
        FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
```

```
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        codF INT NOT NULL,
        FOREIGN KEY (codF) REFERENCES Film(codF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );

CREATE TABLE if not exists MembroTroupeScena(
        codScena INT NOT NULL,
        FOREIGN KEY (codScena) REFERENCES ScenaCiak(codScena)
        ON DELETE CASCADE
        ON UPDATE NO ACTION ,
        CF varchar(12) NOT NULL,
        FOREIGN KEY (CF) REFERENCES MembroTroupe(CF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        PRIMARY KEY(codScena, CF)
        );

CREATE TABLE if not exists Magazzino(
        numMagazzino int Primary Key,
        codInd INT NOT NULL,
        FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );

CREATE TABLE if not exists PosizioneMagazzino(
        codP int,
        numMagazzino INT NOT NULL,
        scaffale int NOT NULL,
        percorso varchar(1) NOT NULL,
        PRIMARY KEY (codP, numMagazzino),
        FOREIGN KEY (numMagazzino) REFERENCES Magazzino(numMagazzino)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );

CREATE TABLE if not exists Costume(
        codC int Primary Key,
        tipo varchar(12) NOT NULL CHECK(tipo='epoca' OR tipo='contemporaneo' OR
tipo='fantasia'),
        descrizione varchar(255) NOT NULL,
        CF varchar(12) NOT NULL,
        FOREIGN KEY (CF) REFERENCES MembroTroupe(CF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        codP INT NOT NULL,
--        FOREIGN KEY (codP) REFERENCES PosizioneMagazzino(codP)
        FOREIGN KEY (codP) REFERENCES PosizioneMagazzino(codP)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );
```

```
CREATE TABLE if not exists StilistaCostume(
        CF varchar(12) NOT NULL,
        FOREIGN KEY (CF) REFERENCES MembroTroupe(CF)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        codC INT NOT NULL,
        FOREIGN KEY (codC) REFERENCES Costume(codC)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        PRIMARY KEY(CF, codC)
        );


CREATE TABLE if not exists CostumeScena(
        codC INT NOT NULL,
        FOREIGN KEY (codC) REFERENCES Costume(codC)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        codScena INT NOT NULL,
        FOREIGN KEY (codScena) REFERENCES ScenaCiak(codScena)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        PRIMARY KEY(codC, codScena)
        );


CREATE TABLE if not exists OggettiDiScena(
        codO int Primary key,
        tipo varchar(21) CHECK (tipo in
('arredo','maschere','armi','mobili','strumentoMusicale','motori')),
        descrizione varchar(255) NOT NULL,
        codP INT NOT NULL,
        FOREIGN KEY (codP) REFERENCES PosizioneMagazzino(codP)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );


CREATE TABLE if not exists OggettoScena(
        codO INT NOT NULL,
        FOREIGN KEY (codO) REFERENCES OggettiDiScena(codO)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        codScena  INT NOT NULL,
        FOREIGN KEY (codScena) REFERENCES ScenaCiak(codScena)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        PRIMARY KEY(codO, codScena)
        );

CREATE TABLE if not exists Ditta(
        P_IVA_DITTA varchar(11) Primary Key,
        nome varchar(41) NOT NULL,
```

```
        codInd INT NOT NULL,
        FOREIGN KEY (codInd) REFERENCES Indirizzo(codInd)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );

CREATE TABLE if not exists Acquisto(
        idAcquisto int Primary Key,
        data date NOT NULL,
        prezzoTotale float(8) NOT NULL,
        P_IVA_DITTA varchar(11) NOT NULL,
        FOREIGN KEY (P_IVA_DITTA) REFERENCES Ditta(P_IVA_DITTA)
        ON DELETE CASCADE
        ON UPDATE NO ACTION
        );

CREATE TABLE if not exists AcquistoCostume(
        codC INT NOT NULL,
        FOREIGN KEY (codC) REFERENCES Costume(codC)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        idAcquisto INT NOT NULL,
        FOREIGN KEY (idAcquisto) REFERENCES Acquisto(idAcquisto)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
        prezzo float(8) NOT NULL,
        PRIMARY KEY(codC, idAcquisto)
        );
```

Special thanks

---

libreria python per generare tutto

generatore di indirizzi random

generatore di IBAN random

generatore di numeri telefono random

generatore di P.IVA random

indirizzo e persona

possibili query per noi

- calcolo percentuale contribuito caporegista e regista (ADDED)

```
select @Denaro := sum(incasso) as money FROM Incasso;
select distinct M.nome, M.cognome,M.percentualeContributo,
```

```
(M.percentualeContributo / 100 * @Denaro ) as Guadagno
from Incasso I, MembroTroupe M
where M.percentualeContributo is not null;
```

TODO !!!!!!!!

- la query sopra va bene, ma potrebbe essere meglio facendo vedere anche i ruoli
  che si possono in teoria vedere cosi, ma george lucas appare 3 volte....
  o mettiamo 1% per ogni ruolo -> 3% totale oppure non mettiamo i ruoli

```
select @Denaro := sum(incasso) as money FROM Incasso;
select distinct M.nome, M.cognome,M.percentualeContributo,
(M.percentualeContributo / 100 * @Denaro ) as guadagno, Rm.nomeRuolo
from Incasso I, MembroTroupe M, RuoloMembroTroupe Rm
where (M.CF = Rm.CF) and M.percentualeContributo is not null;
```

- calcolo trattenute sede territoriale
- Spese mensili totali
- Fatturato Annuo (qui vanno aggiunte tutte le spese, gli stipendi etc etc, mancano le query) lo componiamo di altre query utili a calcolare un eventuale fatturato

```
-- detrazioni

-- query sulle spese per location per ogni scena del film
select @spesa := sum(costoAffittoGiornaliero * durataOre) as Spesa from
ScenaCiak
where codF = 2;
```

-- query su stipendio nei mesi in cui e stato girato select @stipendi := sum(retribuzioneOraria * oreLavorate) as CostoTroupe_Mese from BustaPaga b join Retribuzione r on (b.codB = r.codB) join Film_Membro_Troupe flm on (r.CF = flm.CF) and codF = 2;

-- sottrarre il guadagno di finanziatori con query specifiche select @guadagno := sum(F.percentualeGuadagno / 100 * @fatturato) as guadagno from Finanziatore F join Fondo ff on (F.P_IVA_FINANZIATORE = ff.P_IVA_FINANZIATORE) where F.percentualeGuadagno is not null and codF = 2; select @detrazioni := (@stipendi + @spesa + @guadagno) as Det;

-- Entrate select @fatturato := sum(incasso) as Fatturato from Incasso where codF = 2;

-- Ricavo select @ricavo := @fatturato - @detrazioni as Ricavo;

```
```

- Controllo scena da riprendere in giornata
- /* l'applicativo deve quindi anche poter ordinare gli oggetti e i costumi temporalmente utilizzati nelle riprese*/