Videonoleggio

Fausto Ginepri Alex Ardelean

November 2021

Contents

1 Progettazione concettuale

1.1 Raccolta dei requisiti

Si vuole progettare una base di dati finalizzata a modellare l'organizzazione di un negozio che svolge l'attività di videonoleggio.

Il negozio possiede una collezioni di copie film nei supporti: DVD, BlueRay e VHS. Di ogni film di cui si dispone almeno una copia si vogliono memorizzare alcune informazioni di base: titolo, trama, anno, durata, paese e il numero di copie possedute. Inoltre si vogliono memorizzare il linguaggio: originale, dei doppiaggi e dei sottotitoli disponibili; si vuole sapere il genere di appartenenza del film e gli attori, i registi e i produttori che hanno contribuito alla realizzazione del film. Si vuole memorizzare anche il fatto che un film sia sequel di un'altro film (se il negozio possiede entrambe le copie dei film collegati).

I film vengono noleggiati ai clienti che sono iscritti al negozio e possiedono una tessera. Dei clienti si memorizza quindi: nome, cognome, codice fiscale, residenza, email, telefono.

Ad ogni tessera è associato un saldo utilizzato per effettuare i noleggi. La tessera può essere ricaricata effettuando un pagamento direttamente alla cassa del negozio con annessa una ricevuta fiscale. Ogni cliente deve possedere una tessere e ne può possedere al massimo una.

Il noleggio avviene nel seguente modo: lettura della tessera, scelta del film, conferma del noleggio, consegna della copia fisica e restituzione della copia. Si possono noleggiare più film contemporaneamente ma la procedura di noleggio va effettuata separatamente per ogni film. Il costo del noleggio è di 1€ a cui va aggiunto 1€ per ogni 24h di possesso della copia del film (il costo di 1€/24h viene aggiunto alla scadenza esatta delle 24h). Non si può effettuare un noleggio se il saldo corrente è inferiore al costo minimo di noleggio (ovvero 1€). Al momento della restituzione del film il costo viene detratto dal saldo associato alla tessera, nel caso in cui il saldo risulti negativo non si potrà effettuare un nuovo noleggio fino a che non si ricaricherà la tessera ad un saldo positivo. Dei noleggi si vuole memorizzare la data (e ora) di inizio per i noleggi in corso, mentre per i noleggi conclusi si vuole memorizzare anche la data (e ora) di fine del noleggio.

Nel negozio lavorano due tipologie di personale: il cassiere che si occupa della gestione dei noleggi e della ricarica delle tessere, il capo che si occupa della conduzione del negozio e della fornitura delle copie dei film. Le copie del film vengono fornite al capo tramite un distributore che distribuisce film.

1.2 Analisi delle specifiche

1.2.1 Frasi di carattere generale

Si vuole progettare una base di dati finalizzata a modellare l'organizzazione di un negozio che svolge l'attività di videonoleggio.

1.2.2 Frasi relative ai clienti

I clienti sono iscritti al negozio e possiedono una tessera. Dei clienti si memorizza: nome, cognome, codice fiscale (identifica i clienti), residenza (provincia, città, via, civico), email e telefono.

1.2.3 Frasi relative alle tessere

Ogni cliente possiede una tessera a cui è associato un saldo utilizzato per effettuare i noleggi. La tessera può essere ricaricata effettuando un pagamento direttamente alla cassa del negozio con annessa una ricevuta fiscale. Ogni cliente deve possedere una tessere e ne può possedere al massimo una. Ogni tessera è identificata da un ID.

1.2.4 Frasi relative ai noleggi

Il negozio noleggia le copie dei film ai clienti che possiedono una tessera. Si possono noleggiare più film contemporaneamente ma la procedura di noleggio va effetuata separatamente per ogni film. Il costo del noleggio è di 1€ a cui va aggiunto 1€ per ogni 24h di possesso della copia del film (il costo di 1€/24h viene aggiunto alla scadenza esatta delle 24h). Non si può effettuare un noleggio se il saldo corrente è inferiore al costo minimo di noleggio (ovvero 1€). Al momento della restituzione del film il costo viene detratto dal saldo associato alla tessera. Non si può effettuare un noleggio con un saldo negativo. Si memorizza la data (e l'ora) di inizio del noleggio e la data (e l'ora) di fine del noleggio, in caso di noleggio già concluso.

1.2.5 Frasi relative ai film

Il negozio possiede copie di film di cui si memorizza: titolo, trama, anno, durata, paese e il numero di copie possedute. Si vuole memorizzare il personale cinematografico che ha contribuito alla realizzazione del film. Si vuole memorizzare la lingua: originale, del doppiaggio e dei sottotitoli e l'appartenenza a uno o più generi. Si vuole inoltre memorizzare se un film è sequel di un'altro film posseduto dal negozio. Per ogni film si memorizza anche il numero di copie possedute e per ogni copia si memorizza il supporto e il numero della copia. Ogni film è identificato da un ID. Le copie dei film sono identificate dal numero della copia e dall'ID del film di cui è copia.

1.2.6 Frasi relative al personale cinematografico

Il personale cinematografico si distingue in tre categorie: regista, produttore e attore. Un film è diretto da almeno un regista. Un film è prodotto da almeno un produttore. In un film ha recitato almeno un attore. Del personale cinematografico si memorizzano solo nome e cognome. Il personale cinematografico è identificato da un ID.

1.2.7 Frasi relative al personale

Nel negozio lavorano due tipologie di personale: il cassiere che si occupa della gestione dei noleggi e della ricarica delle tessere, il capo che si occupa della conduzione del negozio e della fornitura delle copie dei film. Il negozio può avere più cassieri ma un unico capo. Del personale si memorizzano nome e cognome e il codice fiscale (come identificatore). Dei cassieri si vuole memorizzare anche email e telefono.

1.2.8 Frasi relative al distributore

Il distributore distribuisce film e fornisce copie fisiche al negozio tramite il capo.

1.3 Glossario termini principali

Termine	Descrizione	Sinonimi	Termini collegati
Cliente	Persona che effettua un noleggio	Iscritto	Tessera, Noleggio, Iscrizione
Tessera	Strumento utile ad effettuare un noleggio		Cliente, Saldo, Noleggio
Noleggio	Prestito di un film su pagamento		Cliente, Film, Copia film, Tessera
Film	Produzione cinematografica		Copia film, Regista, Attore, Produttore
Copia flim	Copia fisica di un film che viene noleggiata ai clienti		Film
Linguaggio	Lingua in cui è disponibile un film		Film
Genere	Genere in cui è disponibile un film		Film
Personale Cinematografico	Persone coinvolte nella realizzazione di un film		Film, Regista, Produttore, Attore
Regista	Persona che dirige un film		Film, Personale Cinematografico
Produttore	Persona che produce un film		Film, Personale Cinematografico
Attore	Persona che recita in un film		Film, Personale Cinematografico
Cassiere	Persona impiegata nel negozio		Tessera, Negozio, Personale
Capo	Persona coinvolta nella conduzione del negozio e nella fornitura di film		Flm, Distributore, Negozio, Personale
Negozio	Luogo in cui si svolgono i noleggi		Capo, Cassiere, Film, Cliente
Distributore	Azienda che distribuisce copie di film		Film, Capo, CopiaFilm

1.4 Elenco delle operazioni

OP1 Aggiunta di un negozio (1 volta).

OP2 Stampa i dati relativi al negozio (10 volte al mese).

OP3 Aggiunta di un cliente (10 volte al mese).

OP4 Visualizzazione del saldo di un cliente (100 volte al giorno).

OP5 Aggiunta di una ricarica (10 volte al giorno).

OP6 Avvio di un noleggio (100 volte al giorno).

OP7 Chiusura di un noleggio (100 volte al giorno).

OP8 Stampa lo storico dei noleggio relativi ad un cliente (10 volte al giorno).

OP9 Stampa la lista dei film disponibili per il noleggio (50 volte al giorno).

OP10 Stampa la lista dei film disponibili per il noleggio relativi ad un attore (30 volte al giorno).

OP11 Stampa la lista dei film disponibili per il noleggio relativi ad un regista (20 volte al giorno).

OP12 Stampa la lista dei film disponibili per il noleggio relativi ad un produttore (10 volte al giorno).

OP13 Aggiunta di un cassiere (1 volta all'anno).

OP14 Aggiunta di un capo (1 volta).

OP15 Aggiunta di una copia film (80 volte al mese).

OP16 Aggiunta di un film (20 volte al mese).

OP17 Aggiunta di un linguaggio (1 volta all'anno).

OP18 Aggiunta di un genere (1 volta all'anno).

OP19 Aggiunta di un regista (2 volta al mese).

OP20 Aggiunta di un attore (5 volte al mese).

OP21 Aggiunta di un produttore (2 volte al mese).

OP22 Aggiunta di un distributore (1 volta al mese).

OP23 Stampa i dati relativi ad un film (200 volte al giorno).

1.5 Schema Entità-Relazione

Il punto di partenza per la realizzazoine del modello concettuale sono i due concetti chiave di CLIENTE, Noleggio e FILM che però vengono realizzati separando il concetto di FILM in senso astratto e copia fisica di un film:

- CLIENTE identifica coloro che noleggiano film.
- COPIAFILM identifica la copia fisica di un film.
- FILM indica il film inteso in senso astratto.

Inoltre si vuole distinguere tra un noleggio già concluso e uno in corso, per fare ciò si fa uso delle relazioni NoleggioCorrente tra CLIENTE e COPIAFILM per indicare i noleggi in corso, e l'entità NOLEGGIOPASSATO per indicare quelli già conclusi. E' necessario usare un'entità per i noleggi passati perché altrimenti un cliente non potrebbe effettuare un noleggio della stessa copia (anche in due istanti temporali diversi).

A partire da questi concetti si può realizzare uno schema scheletro composto dalle entità CLIENTE, COPIAFILM, FILM e NOLEGGIOPASSATO. CLIENTE e COPIAFILM sono legati dalla relazione NoleggioCorrente, COPIAFILM e FILM sono legate dalla relazione CopiaDi, la relazione Compimento lega CLIENTE a NOLEGGIOPASSATO mentre la relazione Di lega NOLEGGIOPASSATO a COPIAFILM.

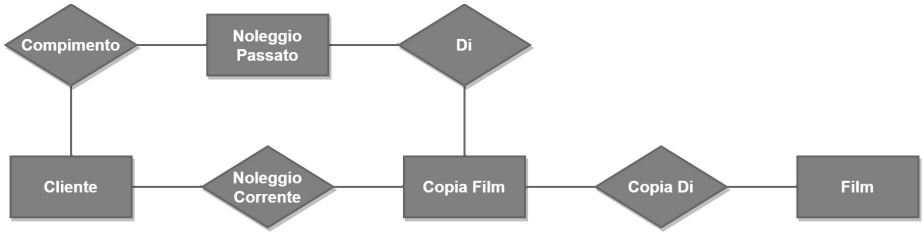

Figure 1: Primo livello

Bisogna ora rappresentare l'iscrizione dei clienti nel NEGOZIO, il loro possesso di una TESSERA e il possesso delle copie dei film da parte del NEGOZIO. Per fare ciò si usano le entità:

- TESSERA identifica la tessera posseduta dai clienti iscritti al negozio.
- NEGOZIO identifica il negozio di videonoleggio.

CLIENTE e TESSERA sono legati dalla relazione Possesso, CLIENTE e NEGOZIO sono legati dalla relazione Noleggio mentre NEGOZIO e COPIAFILM sono legati dalla relazione Possesso2.

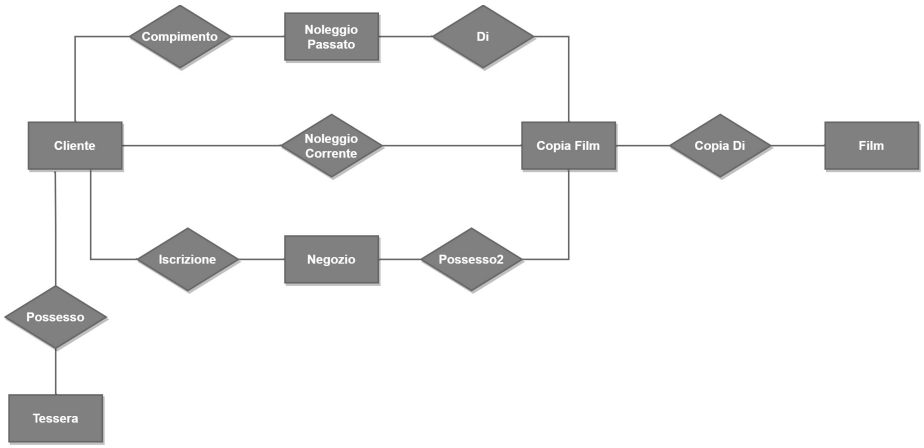

Figure 2: Secondo livello

Si vuole ora rappresentare la parte relativa al PERSONALE che lavora nel negozio. Il PERSONALE può essere di due tipi: CASSIERE e CAPO che svolgono rispettivamente il ruolo di cassiere impiegato nel negozio che effettua le ricariche delle tessere e il ruolo di capo dedito alla conduzione del negozio che fornisce le copie dei film. Tali ruoli possono essere rappresentati dalle relazioni Impiego e Conduzione con NEGOZIO. Il PERSONALE può quindi essere rappresentato come generalizzazione di CASSIERE e CAPO. Il lavoro di ricarica delle tessere può essere rappresentato dalla relazione ternaria Ricarica, che coinvolge le entità TESSERA, RICEVUTAFISCALE e CASSIERE. Mentre il lavoro di fornitura delle copie dei film può essere rappresentato dalla relazione Fornitura tra CAPO e COPIAFILM.

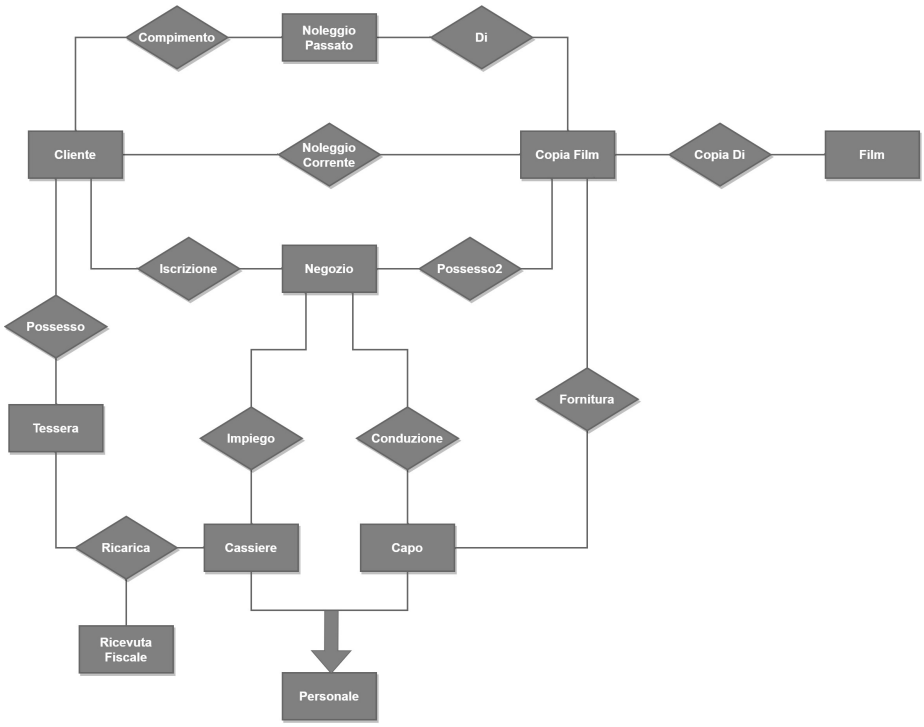

Figure 3: Terzo livello

Si vuole ora espandere il concetto di FILM, dato che di un film si vogliono rappresentare le seguenti proprietà:

- il LINGUAGGIO Originale di un FILM.
- il LINGUAGGIO del Doppiaggio di un FILM.
- il LINGUAGGIO dei Sottotitoli di un FILM.
- l'Appartenenza di un FILM a un GENERE.
- ATTORE che compie una Recitazione in un FILM.
- PRODUTTORE che partecipa alla Produzione di un FILM.
- REGISTA che partecipa alla Direzione di un FILM.
- DISTRIBUTORE che partecipa alla Distribuzione di un FILM e alla Fornitura della COPIAFILM tramite il CAPO.
- essere Sequel di un'altro FILM.

REGISTA, PRODUTTORE e ATTORE possono essere generalizzati dalla generalizzazione PERSONALECINEMATOGRAFICO.

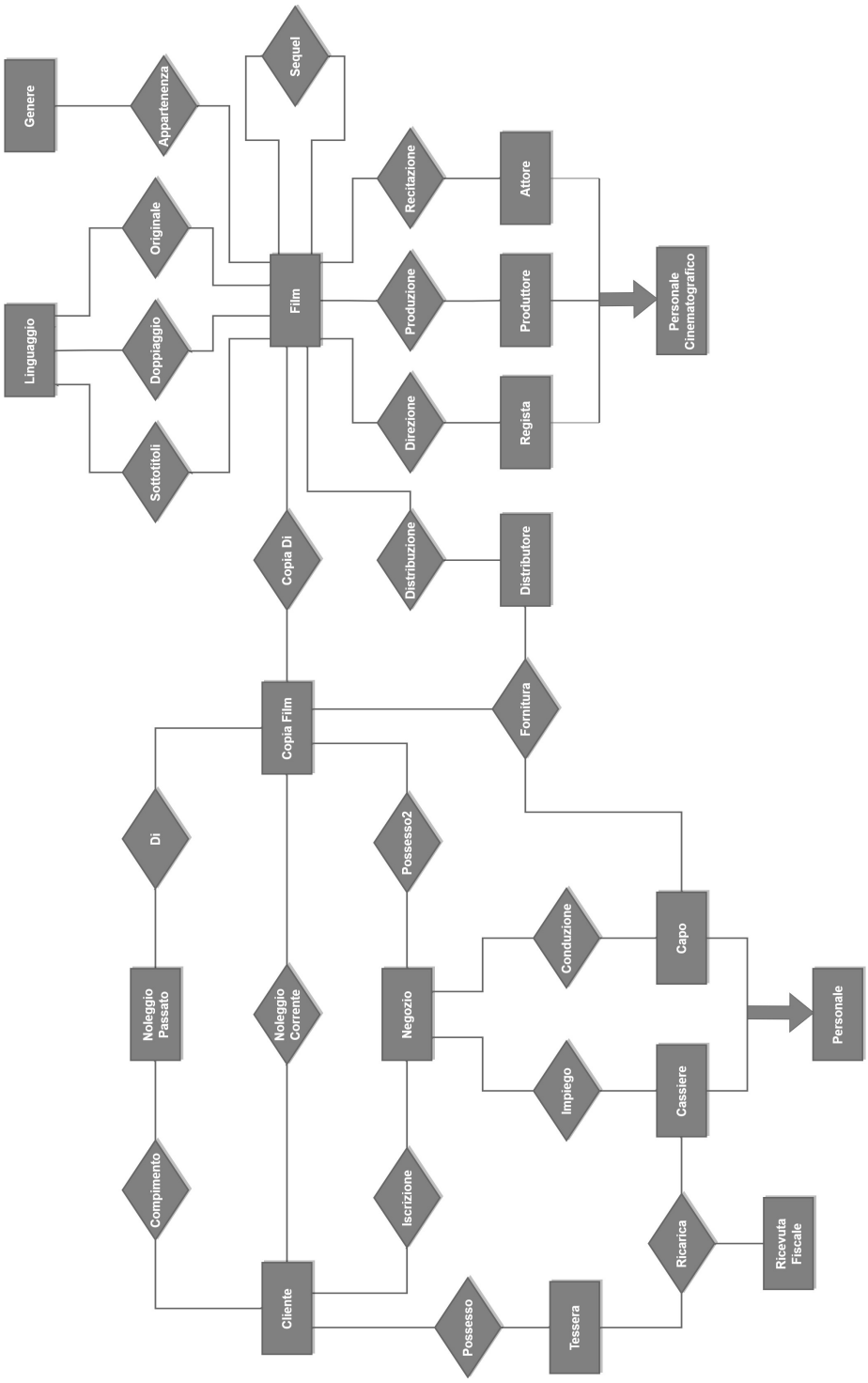

Figure 4: Quarto livello

Per terminare la fase di progettazione concettuale è necessario completare lo schema con alcune informazioni qualitative e quantitative, ovvero con gli attributi e le cardinalità delle relazioni. Le informazioni sono state ricavate dall'analisi delle specifiche.

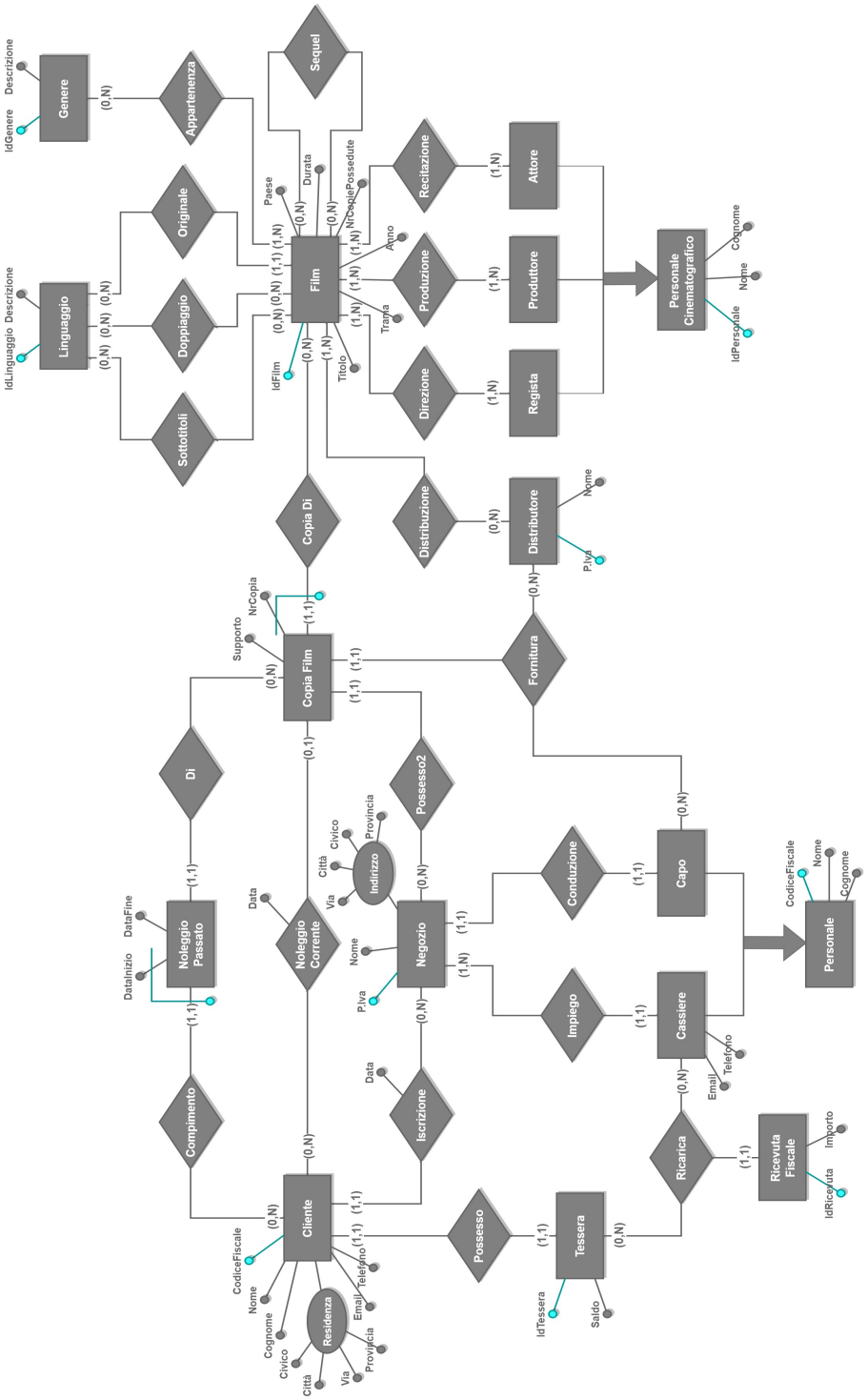

Figure 5: Quinto livello

1.6 Vincoli non esprimibili

- Per effettutare un noleggio il saldo nella tessera deve essere superiore alla soglia minimia di 1€.

1.7 Dizionario dei dati(Entità)

Entità	Identificatore	Attributi	Descrizione
CLIENTE	CodiceFiscale	Nome, Cognome, Provincia, Città, Civico, Via, Email, Telefono	Cliente del negozio
NEGOZIO	P.Iva	Nome, Provincia, Via, Civico, Città	Dati relativi al negozio
COPIAFILM	NrCopia, IdFilm	Supporto	Copia fisica di un film astratto
NOLEGGIO PASSATO	DataInizio, CodiceFiscale	DataFine	Noleggio concluso
TESSERA	IdTessera	Saldo	Tessera posseduta dai clienti usata per noleggiare
RIVEVUTA FISCALE	IdRicevuta	Importo	Ricevuta emessa nell'operazione di ricarica della tessera
CASSIERE	CodiceFiscale	Email, Telefono	Personale impiegato nel negozio
CAPO	CodiceFiscale		Personale alla conduzione del negozio
PERSON-ALE	CodiceFiscale	Nome, Cognome	Persona che lavora nel negozio
FILM	IdFilm	Titolo, Anno, Trama, NrCopiePossedute, Durata, Paese	Film inteso in senso astratto
LINGUAG-GIO	IdLingua	Descrizione	Lingua in cui un film può essere disponibile
GENERE	IdGenere	Descrizione	Genere a cui un film può appartenere
ATTORE	IdPersonale		Persona che recita in un film
PRODUT-TORE	IdPersonale		Persona che produce un film
REGISTA	IdPersonale		Persona che dirige un film
PERSON-ALE CINE-MATOGRAFICO	IdPersonale	Nome, Cognome	Persona che realizza un film
DISTRIBU-TORE	P.Iva	Nome	Ente che distribuisce film

1.8 Dizionario dei dati(Relazioni)

Relazione	Entità	Attributi	Descrizione
NOLEGGIO CORRENTE	Cliente, CopiaFilm	Data	Noleggio in corso delle copie fisiche dei film da parte dei clienti
COMPIMENTO	Cliente, NoleggioPassato		Noleggio concluso da parte di un cliente
DI	NoleggioPassato, CopiaFilm		Copia fisica relativa ad un noleggio concluso
ISCRIZIONE	Cliente, Negozio	Data	Iscrizione di un cliente al negozio
COPIADI	CopiaFilm, Film		Copia fisica di un film
POSSESSO	Cliente, Tessera		Tessera relativa ad un cliente
POSSESSO2	Negozio, CopiaFilm		Copie fisiche dei film che possiede il negozio
IMPIEGO	Cassiere, Negozio		Ruolo del cassiere all'interno del negozio
CONDUZIONE	Capo, Negozio		Ruolo del capo all'interno del negozio
RICARICA	Tessera, Cassiere, RicevutaFiscale		Ricarica del saldo della tessere effettuata dal cassiere
FORNITURA	Capo, Dostributore, CopiaFilm		Acquisizione delle copie dei film dall'azienda di distribuzione
DISTRIBUZIONE	Distributore, Film		Film distribuiti da un distributore
ORIGINALW	Film, Linguaggio		Linguaggio originale disponibile per un film
DOPPIAGGIO	Film, Linguaggio		Doppiaggio disponibile per un film
SOTTOTITOLI	Film, Linguaggio		Sottotitoli disponibili per un film
APPARTENENZA	Film, Genere		Genere di appartenenza per un film
SEQUEL	Film, Film		Capitolo successivo di un film
DIREZIONE	Film, Regista		Direzione cinematografica di un film
PRODUZIONE	Film, Produttore		Produzione cinematografica di un film
RECITAZIONE	Film, Attore		Attori partecipanti al cast di un film

2 Progettazione logica

2.1 Tavola dei volumi

Concetto	Tipo	Volume
CLIENTE	E	500
NEGOZIO	E	1
FILM	E	10000
COPIAFILM	E	4(copie film/film)×10000(film)=40000
NOLEGGIOPASSATO	E	50(noleggi/cliente)×500(clienti)=25000
TESSERA	E	500
RICEVUTAFISCALE	E	50(ricarica/cliente)×500(clienti)=25000
CASSIERE	E	3
CAPO	E	1
PERSONALE	E	4(cassiere+capo)
LINGUAGGIO	E	15
GENERE	E	15
ATTORE	E	10000
PRODUTTORE	E	500
REGISTA	E	500
PERSONALECINEMATOGRAFICO	E	11000(attore+produttore+regista)
DISTRIBUTORE	E	50
NOLEGGIOCORRENTE	R	0.2(noleggio/cliente)×500(clienti)=100
COMPIMENTO	R	50(noleggi/cliente)×500(clienti)=25000
DI	R	50(noleggi/cliente)×500(clienti)=25000
ISCRIZIONE	R	500
COPIADI	R	4(copie film/film)×10000(film)=40000
POSSESSO	R	500
POSSESSO2	R	4(copie film/film)×10000(film)=40000
IMPIEGO	R	3
CONDUZIONE	R	1
RICARICA	R	25000
FORNITURA	R	4(copie film/film)×10000(film)=40000
DISTRIBUZIONE	R	10000
ORIGINALE	R	1(linguaggi/film)×10000(film)=10000
DOPPIAGGIO	R	5(linguaggi/film)×10000(film)=50000
SOTTOTITOLI	R	10(linguaggi/film)×10000(film)=100000
APPARTENENZA	R	2(generi/film)×10000(film)=20000
SEQUEL	R	0.01(sequel/film)×10000(film)=100
DIREZIONE	R	1.1(registi/film)×10000(film)=11000
PRODUZIONE	R	2.5(produttori/film)×10000(film)=25000
RECITAZIONE	R	5(attori/film)×10000(film)=50000

Table 1: Tavola dei volumi

2.2 Tavola delle operazioni

Operazioni	Tipo	Frequenza
OP1 Aggiunta di un negozio	B	1 volta
OP2 Stampa i dati relativi al negozio	B	10 volte al mese
OP3 Aggiunta di un cliente	B	10 volte al mese
OP4 Visualizzazione del saldo di un cliente	I	100 volte al giorno
OP5 Aggiunta di una ricarica	I	10 volte al giorno
OP6 Avvio di un noleggio	I	100 volte al giorno
OP7 Chiusura di un noleggio	I	100 volte al giorno
OP8 Stampa lo storico dei noleggio relativi ad un cliente	I	10 volte al giorno
OP9 Stampa la lista dei film disponibili per il noleggio	I	50 volte al giorno
OP10 Stampa la lista dei film disponibili per il noleggio relativi ad un attore	I	30 volte al giorno
OP11 Stampa la lista dei film disponibili per il noleggio relativi ad un regista	I	20 volte al giorno
OP12 Stampa la lista dei film disponibili per il noleggio relativi ad un produttore	I	10 volte al giorno
OP13 Aggiunta di un cassiere	B	1 volta all'anno
OP14 Aggiunta di un capo	B	1 volta
OP15 Aggiunta di una copia film	B	80 volte al mese
OP16 Aggiunta di un film	B	20 volte al mese
OP17 Aggiunta di un linguaggio	B	1 volta all'anno
OP18 Aggiunta di un genere	B	1 volta all'anno
OP19 Aggiunta di un regista	B	2 volta al mese
OP20 Aggiunta di un attore	B	5 volte al mese
OP21 Aggiunta di un produttore	B	2 volte al mese
OP22 Aggiutna di un distributore	B	1 volta al mese
OP23 Stampa i dati relativi ad un film	I	200 volte al giorno

2.3 Analisi delle ridondanze

2.3.1 Ridondanza 1

Una possibile ridondanza è quella riguardante il numero di copie possedute di un film. Questo valore può essere ricavato conteggiando le occorrenze della relazione CopiaDi.

Le operazioni che coinvolgono tale attributo sono:

- Op15 Aggiunta di una copia film
- Op23 Stampa i dati relativi ad un film

In presenza di ridondanza si ha:

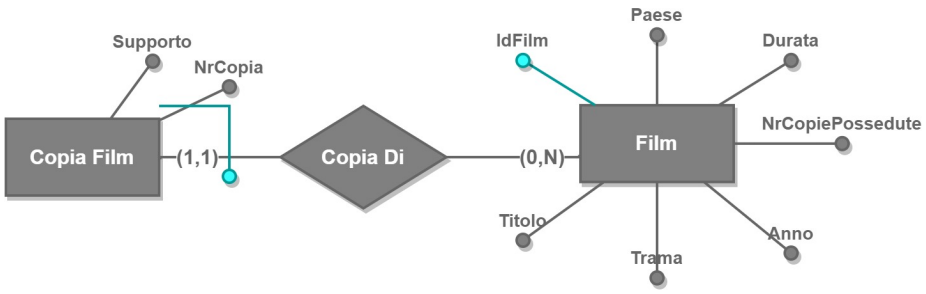

Figure 6: Schema con ridondanza

Tavola degli accessi per l'operazione 15:

Concetto	Costrutto	Accessi
Film	Entità	1 Lettura
Film	Entità	1 Scrittura
CopiaDi	Relazione	1 Scrittura
CopiaFilm	Entità	1 Scrittura

Table 2: Tavola degli accessi per l'operazione 15

Tavola degli accessi per l'operazione 23:

Concetto	Costrutto	Accessi
Film	Entità	1 Lettura

Table 3: Tavola degli accessi per l'operazione 23

Considerando il peso di una scrittura pari a 2 letture, si ha un costo totale di accessi al mese di:

$$(3S + 1L) \times 80 + (1L) \times 200 \times 30 =$$
$$(7L) \times 80 + (1L) \times 200 \times 30 = 6560L$$

Poiché il numero medio di copie possedute per ogni film è 4 è sufficiente 1 byte per rappresentare tale informazione. E dato che il numero di occorrenze di film è 10000 si otterrebbe uno spreco di $10000 \times 1byte = 10000$ bytes. Si vuole ora valutare il costo in assenza di ridondanze:

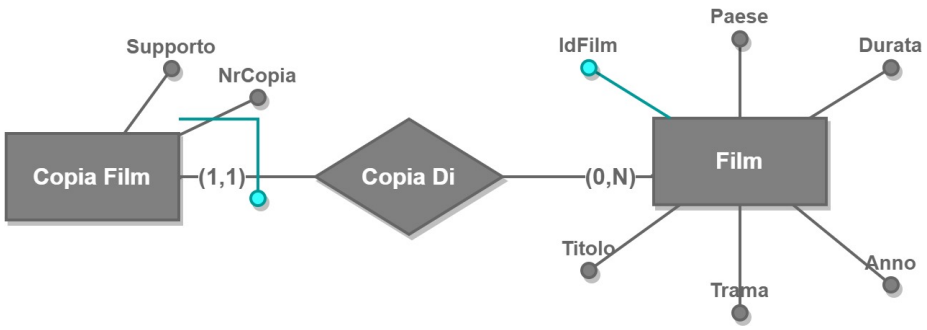

Figure 7: Schema senza ridondanza

Tavola degli accessi per l'operazione 15:

Concetto	Costrutto	Accessi
CopiaFilm	Entità	1 Scrittura
CopiaDi	Rekazione	1 Scrittura

Table 4: Tavola degli accessi per l'operazione 15

Tavola degli accessi per l'operazione 23:

Concetto	Costrutto	Accessi
Film	Entità	1 Lettura
CopiaDi	Relazione	4 Lettura

Table 5: Tavola degli accessi per l'operazione 26

Costo:

$$(2S) \times 80 + (5L) \times 200 \times 30 =$$
$$(4L) \times 80 + (5L) \times 200 \times 30 = 30320L$$

Il numero di accessi in presenza di ridondanza(6560L) sono molto inferiori rispetto a quelli in assenza di ridondanza(30320L), e nonostante il fatto che in presenza di ridondanza si ha uno spreco di 10000byte, si sceglie comunque di mantendere la ridondanza dato l'evidente vantaggio in termini di accessi.

2.3.2 Ridondanza 2

Un'altra ridondanza individuata è quella riguardante il saldo della tessera di un cliente. Questo valore può essere ricavato tenendo conto delle ricariche eseguite

e del costo dei noleggi passati(che si calcola sulla base della data di inizio e fine). Questo valore si può ricavare dall'attributo Importo dell'entità RICEVUTAFISCALE e dagli attributi DataInizio e DataFine di NOLEGGIOPASSATO.

Le operazioni che coinvolgono tale attributo sono:

- Op4 Visualizzazione del saldo di una tessera.
- Op5 Aggiunta di una ricarica.
- Op7 Chiusura di un noleggio.

In presenza di ridondanza si ha:

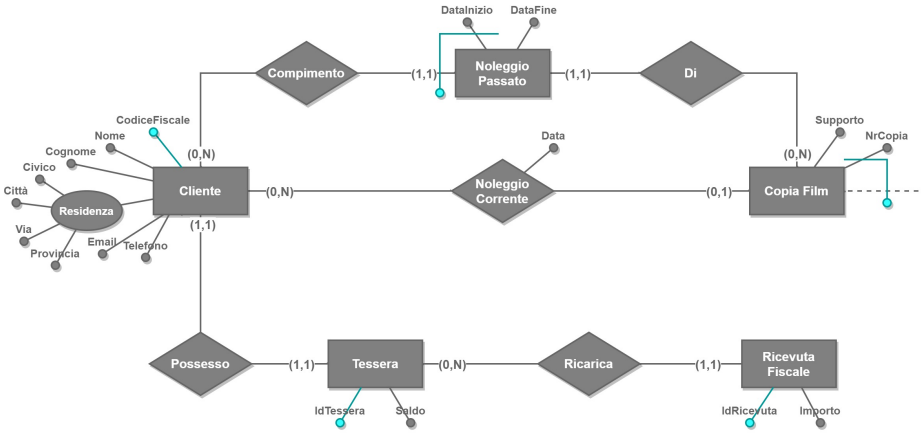

Figure 8: Schema con ridondanza

Tavola degli accessi per l'operazione 4:

Concetto	Costrutto	Accessi
Tessera	Entità	1 Lettura

Table 6: Tavola degli accessi per l'operazione 4

Tavola degli accessi per l'operazione 5:

Concetto	Costrutto	Accessi
Tessera	Entità	1 Lettura
Tessera	Entità	1 Scrittura
Ricarica	Relazione	1 Scrittura
RicevutaFiscale	Entità	1 Scrittura

Table 7: Tavola degli accessi per l'operazione 5

Tavola degli accessi per l'operazione 7:

Concetto	Costrutto	Accessi
Cliente	Entità	1 Lettura
NoleggioCorrente	Relazione	1 Lettura
CopiaFilm	Entità	1 Lettura
NoleggioCorrente	Relazione	1 Scrittura
Compimento	Relazione	1 Scrittura
NoleggioPassato	Entità	1 Scrittura
Di	Relazione	1 Scrittura
Possesso	Relazione	1 Lettura
Tessera	Entità	1 Lettura
Tessera	Entità	1 Scrittura

Table 8: Tavola degli accessi per l'operazione 7

Considerando il peso di una scrittura pari a 2 letture, si ha un costo totale di accessi al giorno di:

$$(1L) \times 100 + (1L + 3S) \times 10 + (5L + 5S) \times 100 =$$
$$(1L) \times 100 + (7L) \times 10 + (15L) \times 100 = 1670L$$

Supponendo di poter rappresentare il saldo con 2 byte si ha uno spreco di memoria pari a $2(byte) \times 500(tessere) = 1000byte$.

In assenza di ridondanza si ha:

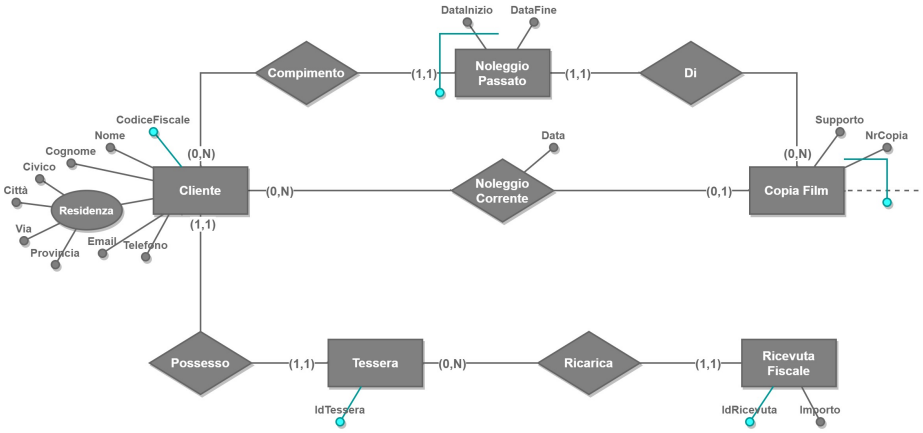

Figure 9: Schema senza ridondanza

Tavola degli accessi per l'operazione 4:

Concetto	Costrutto	Accessi
Tessera	Entità	1 Lettura
Possesso	Relazione	1 Lettura
Cliente	Entità	1 Lettura
Compimento	Relazione	50 Letture
NoleggioPassato	Entità	50 Letture
Ricarica	Relazione	50 Lettura
RicevutaFiscale	Entità	50 Letture

Table 9: Tavola degli accessi per l'operazione 4

Tavola degli accessi per l'operazione 5:

Concetto	Costrutto	Accessi
Ricarica	Relazione	1 Scrittura
RicevutaFiscale	Entità	1 Scrittura

Table 10: Tavola degli accessi per l'operazione 5

Tavola degli accessi per l'operazione 7:

Concetto	Costrutto	Accessi
NoleggioCorrente	Relazione	1 Lettura
NoleggioCorrente	Relazione	1 Scrittura
Cliente	Entità	1 Lettura
NoleggioPassato	Entità	1 Scrittura
Compimento	Relazione	1 Scrittura
Di	Relazione	1 Scrittura

Table 11: Tavola degli accessi per l'operazione 7

Considerando il peso di una scrittura pari a 2 letture, si ha un costo totale di accessi al giorno di:

$$(203L) \times 100 + (2S) \times 10 + (2L + 4S) \times 100 =$$
$$(203L) \times 100 + (4L) \times 10 + (10L) \times 100 = 21493L$$

Nonostante lo spreco di memoria di 1000 byte, conviene mantenere la ridondanza al fine di favorire l'efficienza. In presenza di ridondanza si hanno infatti solo 1670L accessi a fronte di 21493L accessi in assenza di ridondanza.

2.4 Eliminazione delle generalizzazioni

Lo schema presenta due generalizzazioni:

- PERSONALE che generalizza CASSIERE e CAPO.
- PERSONALECINEMATOGRAFICO che generalizza REGISTA, PRODUTTORE e ATTORE.

Dato che si tratta in entrambi i casi di una generalizzazione totale e le operazioni si riferiscono solo a singole entità, si possono eliminare entrambe tramite accorpamento del genitore della generalizzazione nelle figlie.

Il nuovo schema senza generalizzazioni risulta essere:

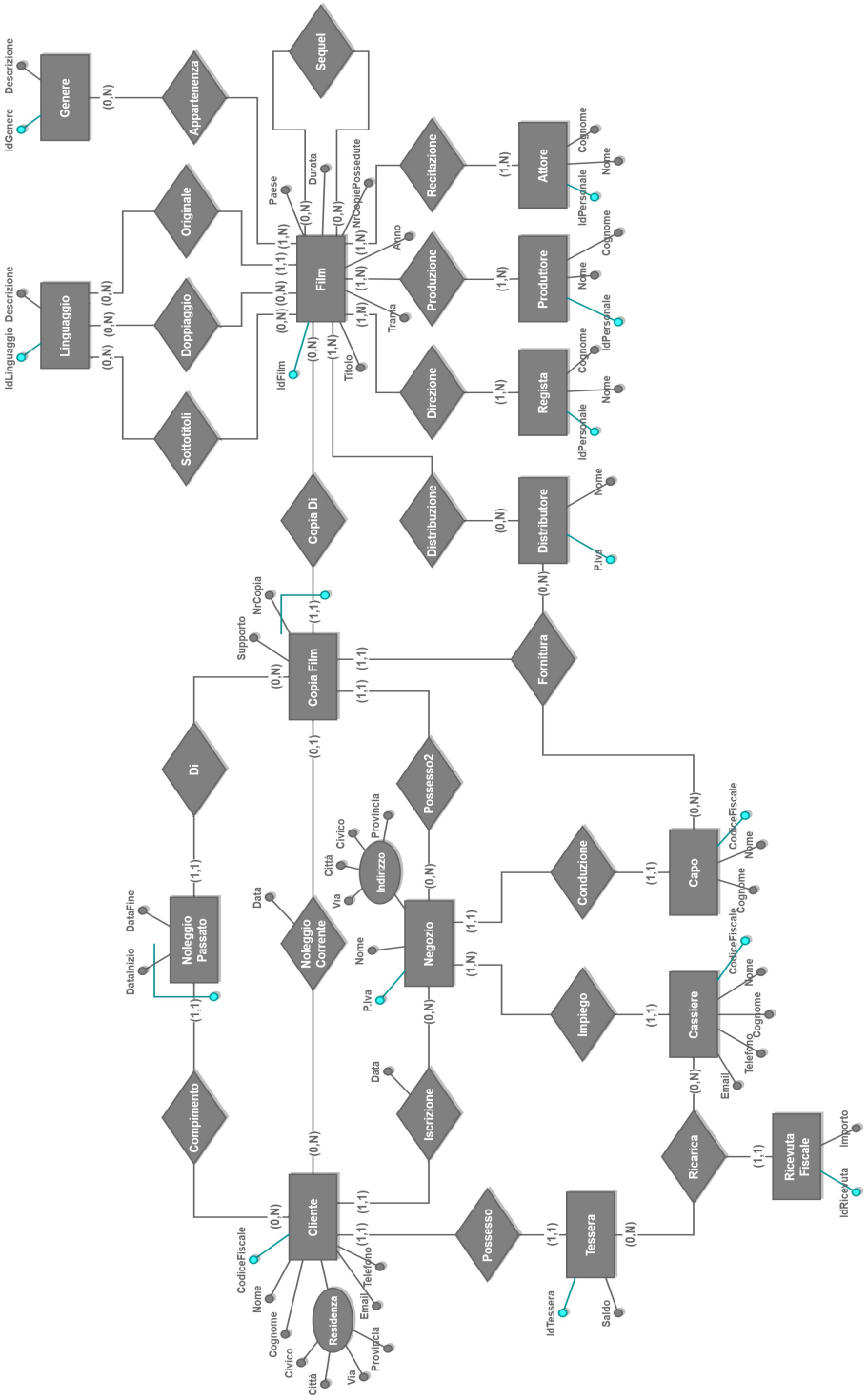

Figure 10: Schema senza generalizzazioni

2.5 Partizionamento/accorpamento di concetti

Non sono state individuate possibili relationship da accorpare/partizionare.

Dato che l'entità CLIENTE e l'entità TESSERA vengono accedute insieme è possibile accorparle nell'entità CLIENTE.

2.6 Scelta degli identificatori primari

Tutte le entità possiedono un unico identificatore che può essere usato come identificatore principale.

Perciò lo schema finale ristrutturato risulta essere:

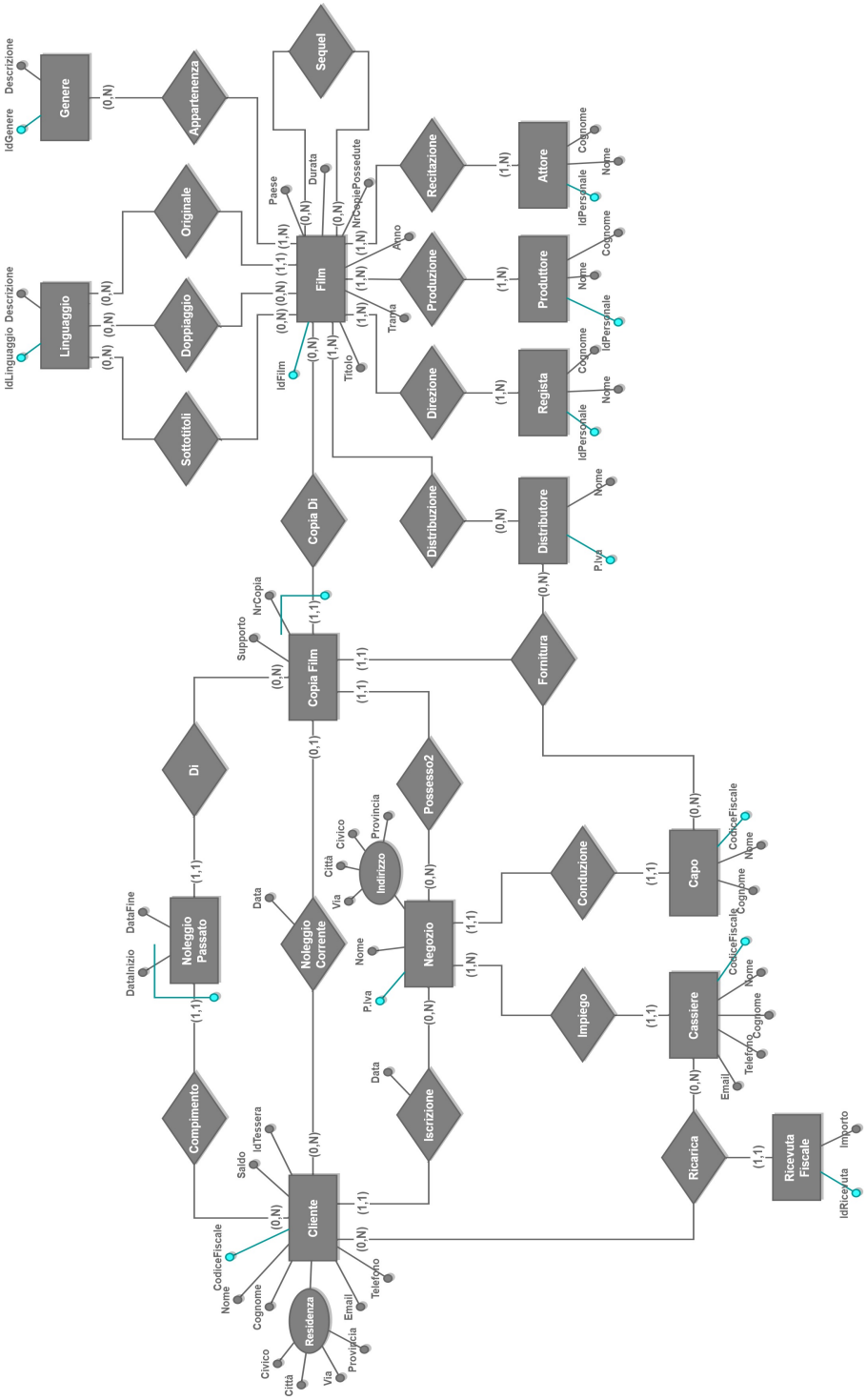

Figure 11: Schema ristrutturato

2.7 Modello relazionale

2.7.1 Traduzione di entità

CLIENTE(CodiceFiscale, Nome, Cognome, Email, Telefono, Provincia, Città, Via, Civico, Saldo, IdTessera)

COPIAFILM(NrCopia, Film, Supporto)

Con il vincolo di integrità referenziale tra Film e la chiave della relazione FILM.

NOLEGGIOPASSATO(Cliente, DataInizio, DataFine, NrCopia, Film)

Con il vincolo di integrità referenziale tra Cliente e la chiave della relazione CLIENTE e il vincolo di integrità referenziale tra NrCopia e Film e la chiave di COPIAFILM.

NEGOZIO(P.Iva, Nome, Provincia, Città, Via, Civico, Capo)

Con il vincolo di integrità referenziale tra Capo e la chiave della relazione CAPO.

RICEVUTAFISCALE(IdRicevuta, Importo)

CASSIERE(CodiceFiscale, Nome, Cognome, Telefono, Email)

CAPO(CodiceFiscale, Nome, Cognome)

LINGUAGGIO(IdLinguaggio, Descrizione)

GENERE(IdGenere, Descrizione)

ATTORE(IdPersonale, Nome, Cognome)

PRODUTTORE(IdPersonale, Nome, Cognome)

REGISTA(IdPersonale, Nome, Cognome)

FILM(IdFilm, Titolo, Trama, Anno, Paese, Durata, NrCopiePossedute)

DISTRIBUTORE(P.Iva, Nome)

2.7.2 Traduzione di relazioni

NOLEGGIOCORRENTE(NrCopia, Film, Cliente, Data)

Con vincoli di integrità referenziale tra:

- Cliente in NOLEGGIOCORRENTE e (la chiave di) CLIENTE.
- NrCopia e Film in NOLEGGIOCORRENTE e (la chiave di) COPIAFILM.

ISCRIZIONE(Cliente, Negozio, Data)

Con vincoli di integrità referenziale tra:

- Cliente in ISCRIZIONE e (la chiave di) CLIENTE.

- Negozio in ISCRIZIONE e (la chiave di) NEGOZIO.

POSSESSO2(NrCopia, Film, Negozio)

Con vincoli di integrità referenziale tra:

- NrCopia e Film in POSSESSO2 e (la chiave di) COPIAFILM.
- Negozio in POSSESSO2 e (la chiave di) NEGOZIO.

RICARICA(RicevutaFiscale, Cliente, Cassiere)

Con vincoli di integrità referenziale tra:

- Cliente in RICARICA e (la chiave di) CLIENTE.
- Cassiere in RICARICA e (la chiave di) CASSIERE.
- RicevutaFiscale in RICARICA e (la chiave di) RICEVUTAFISCALE.

IMPIEGO(Cassiere, Negozio)

Con vincoli di integrità referenziale tra:

- Cassiere in IMPIEGO e (la chiave di) CASSIERE.
- Negozio in IMPIEGO e (la chiave di) NEGOZIO.

SOTTOTITOLI(Film, Linguaggio)

Con vincoli di integrità referenziale tra:

- Film in SOTTOTITOLI e (la chiave di) FILM.
- Linguaggio in SOTTOTITOLI e (la chiave di) LINGUAGGIO.

DOPPIAGGIO(Film, Linguaggio)

Con vincoli di integrità referenziale tra:

- Film in DOPPIAGGIO e (la chiave di) FILM.
- Linguaggio in DOPPIAGGIO e (la chiave di) LINGUAGGIO.

ORIGINALE(Film, Linguaggio)

Con vincoli di integrità referenziale tra:

- Film in ORIGINALE e (la chiave di) FILM.
- Linguaggio in ORIGINALE e (la chiave di) LINGUAGGIO.

APPARTENENZA(Film, Genere)

Con vincoli di integrità referenziale tra:

- Film in APPARTENENZA e (la chiave di) FILM.
- Genere in APPARTENENZA e (la chiave di) GENERE.

SEQUEL(Precedente, Successivo)

Con vincoli di integrità referenziale tra:

- Precedente in SEQUEL e (la chiave di) FILM.
- Successivo in SEQUEL e (la chiave di) FILM.

RECITAZIONE(Film, Attore)

Con vincoli di integrità referenziale tra:

- Film in RECITAZIONE e (la chiave di) FILM.
- Attore in RECITAZIONE e (la chiave di) ATTORE.

PRODUZIONE(Film, Produttore)

Con vincoli di integrità referenziale tra:

- Film in PRODUZIONE e (la chiave di) FILM.
- Produttore in PRODUZIONE e (la chiave di) PRODUTTORE.

DIREZIONE(Film, Regista)

Con vincoli di integrità referenziale tra:

- Film in DIREZIONE e (la chiave di) FILM.
- Regista in DIREZIONE e (la chiave di) REGISTA.

DISTRIBUZIONE(Film, Distributore)

Con vincoli di integrità referenziale tra:

- Film in DISTRIBUZIONE e (la chiave di) FILM.
- Distributore in DISTRIBUZIONE e (la chiave di) DISTRIBUTORE.

FORNITURA(CopiaFilm, Film, Capo, Distributore)

Con vincoli di integrità referenziale tra:

- CopiaFilm e Film in FORNITURA e (la chiave di) COPIAFILM.
- Capo in FORNITURA e (la chiave di) CAPO.
- Distributore in FORNITURA e (la chiave di) DISTRIBUTORE.

3 Implementazione delle operazioni

3.1 Creazione delle tabelle

```
CREATE TABLE "Cliente"
(
    codice_fiscale character(16) NOT NULL,
    nome character varying(30) NOT NULL,
    cognome character varying(30) NOT NULL,
    email character varying(30),
    telefono integer,
    provincia character(2),
    citta character varying(30),
    via character varying(30),
    civico integer,
    saldo numeric(19,2),
    id_tessera integer,
    CONSTRAINT "Cliente_pkey" PRIMARY KEY (codice_fiscale)
)

CREATE TABLE "Film"
(
    id_film integer NOT NULL,
    titolo character varying(30) NOT NULL,
    trama character varying(300),
    anno integer,
    paese character varying(30),
    durata integer,
    nr_copie_possedute integer,
    CONSTRAINT "Film_pkey" PRIMARY KEY (id_film)
)

CREATE TABLE "CopiaFilm"
(
    nr_copia integer NOT NULL,
    film integer NOT NULL,
    supporto character varying(30),
    CONSTRAINT "CopiaFilm_pkey" PRIMARY KEY (nr_copia, film),
    CONSTRAINT "CopiaFilm_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "NoleggioPassato"
(
    cliente character varying(16) NOT NULL,
    data_inizio timestamp with time zone NOT NULL,
    data_fine timestamp with time zone NOT NULL,
    nr_copia integer NOT NULL,
    film integer NOT NULL,
    CONSTRAINT "NoleggioPassato_pkey" PRIMARY KEY (cliente, data_inizio),
    CONSTRAINT "NoleggioPassato_cliente_fkey" FOREIGN KEY (cliente)
        REFERENCES public."Cliente" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "NoleggioPassato_film_nr_copia_fkey"
    FOREIGN KEY (film, nr_copia)
        REFERENCES public."CopiaFilm" (film, nr_copia) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

CREATE TABLE public."Capo"
(
    codice_fiscale character(16) NOT NULL,
    nome character varying(30) NOT NULL,
    cognome character varying(30) NOT NULL,
    CONSTRAINT "Capo_pkey" PRIMARY KEY (codice_fiscale)
)

CREATE TABLE "Negozio"
(
    p_iva character(11) NOT NULL,
    nome character varying(30) NOT NULL,
    provincia character(2) NOT NULL,
    citta character varying(30) NOT NULL,
    via character varying(30) NOT NULL,
    civico integer NOT NULL,
    capo character(16) NOT NULL,
    CONSTRAINT "Negozio_pkey" PRIMARY KEY (p_iva),
    CONSTRAINT "Negozio_capo_fkey" FOREIGN KEY (capo)
        REFERENCES public."Capo" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

```
CREATE TABLE "RicevutaFiscale"
(
    id_ricevuta integer NOT NULL,
    importo numeric(19,2) NOT NULL,
    CONSTRAINT "RicevutaFiscale_pkey" PRIMARY KEY (id_ricevuta)
)

CREATE TABLE "Cassiere"
(
    codice_fiscale character(16) NOT NULL,
    nome character varying(30) NOT NULL,
    cogonome character varying(30) NOT NULL,
    telefono integer,
    email character varying(30),
    CONSTRAINT "Cassiere_pkey" PRIMARY KEY (codice_fiscale)
)

CREATE TABLE "Linguaggio"
(
    id_linguaggio integer NOT NULL,
    descrizione character varying(30) NOT NULL,
    CONSTRAINT "Linguaggio_pkey" PRIMARY KEY (id_linguaggio)
)

CREATE TABLE "Genere"
(
    id_genere integer NOT NULL,
    descrizione character varying(30) NOT NULL,
    CONSTRAINT "Genere_pkey" PRIMARY KEY (id_genere)
)

CREATE TABLE "Attore"
(
    id_personale integer NOT NULL,
    nome character varying(30) NOT NULL,
    cognome character varying(30) NOT NULL,
    CONSTRAINT "Attore_pkey" PRIMARY KEY (id_personale)
)

CREATE TABLE "Produttore"
(
    id_personale integer NOT NULL,
    nome character varying(30) NOT NULL,
    cognome character varying(30) NOT NULL,
    CONSTRAINT "Produttore_pkey" PRIMARY KEY (id_personale)
)
```

```
CREATE TABLE "Regista"
(
    id_personale integer NOT NULL,
    nome character varying(30) NOT NULL,
    cognome character varying(30) NOT NULL,
    CONSTRAINT "Regista_pkey" PRIMARY KEY (id_regista)
)

CREATE TABLE "Distributore"
(
    p_iva character(11) NOT NULL,
    nome character varying(30) NOT NULL,
    CONSTRAINT "Distributore_pkey" PRIMARY KEY (p_iva)
)

CREATE TABLE "NoleggioCorrente"
(
    nr_copia integer NOT NULL,
    film integer NOT NULL,
    cliente character(16) NOT NULL,
    data timestamp with time zone NOT NULL,
    CONSTRAINT "NoleggioCorrente_pkey" PRIMARY KEY (nr_copia, film),
    CONSTRAINT "NoleggioCorrente_cliente_fkey" FOREIGN KEY (cliente)
        REFERENCES public."Cliente" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "NoleggioCorrente_nr_copia_film_fkey"
    FOREIGN KEY (film, nr_copia)
        REFERENCES public."CopiaFilm" (film, nr_copia) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Iscrizione"
(
    cliente character(16) NOT NULL,
    negozio character(11) NOT NULL,
    data time with time zone NOT NULL,
    CONSTRAINT "Iscrizione_pkey" PRIMARY KEY (cliente, negozio),
    CONSTRAINT "Iscrizione_cliente_fkey" FOREIGN KEY (cliente)
        REFERENCES public."Cliente" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Iscrizione_negozio_fkey" FOREIGN KEY (negozio)
        REFERENCES public."Negozio" (p_iva) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
CREATE TABLE "Possesso2"
(
    nr_copia integer NOT NULL,
    film integer NOT NULL,
    negozio character(11) NOT NULL,
    CONSTRAINT "Possesso2_pkey" PRIMARY KEY (nr_copia, film),
    CONSTRAINT "Possesso2_negozio_fkey" FOREIGN KEY (negozio)
        REFERENCES public."Negozio" (p_iva) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Possesso2_nr_copia_film_fkey" FOREIGN KEY (film, nr_copia)
        REFERENCES public."CopiaFilm" (film, nr_copia) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Ricarica"
(
    ricevuta_fiscale integer NOT NULL,
    cliente character(16) NOT NULL,
    cassiere character(16) NOT NULL,
    CONSTRAINT "Ricarica_pkey" PRIMARY KEY (ricevuta_fiscale),
    CONSTRAINT "Ricarica_cassiere_fkey" FOREIGN KEY (cassiere)
        REFERENCES public."Cassiere" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Ricarica_cliente_fkey" FOREIGN KEY (cliente)
        REFERENCES public."Cliente" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Ricarica_ricevuta_fiscale_fkey" FOREIGN KEY (ricevuta_fiscale)
        REFERENCES public."RicevutaFiscale" (id_ricevuta) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
CREATE TABLE "Impiego"
(
    cassiere character(16) NOT NULL,
    negozio character(11) NOT NULL,
    CONSTRAINT "Impiego_pkey" PRIMARY KEY (cassiere),
    CONSTRAINT "Impiego_cassiere_fkey" FOREIGN KEY (cassiere)
        REFERENCES public."Cassiere" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Impiego_negozio_fkey" FOREIGN KEY (negozio)
        REFERENCES public."Negozio" (p_iva) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Sottotitoli"
(
    film integer NOT NULL,
    linguaggio integer NOT NULL,
    CONSTRAINT "Sottotitoli_pkey" PRIMARY KEY (film, linguaggio),
    CONSTRAINT "Sottotitoli_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Sottotitoli_linguaggio_fkey" FOREIGN KEY (linguaggio)
        REFERENCES public."Linguaggio" (id_linguaggio) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

CREATE TABLE "Doppiaggio"
(
    film integer NOT NULL,
    linguaggio integer NOT NULL,
    CONSTRAINT "Doppiaggio_pkey" PRIMARY KEY (film, linguaggio),
    CONSTRAINT "Doppiaggio_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Doppiaggio_linguaggio_fkey" FOREIGN KEY (linguaggio)
        REFERENCES public."Linguaggio" (id_linguaggio) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

CREATE TABLE "Originale"
(
    film integer NOT NULL,
    linguaggio integer NOT NULL,
    CONSTRAINT "Originale_pkey" PRIMARY KEY (film),
    CONSTRAINT "Originale_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Appartenenza"
(
    film integer NOT NULL,
    genere integer NOT NULL,
    CONSTRAINT "Appartenenza_pkey" PRIMARY KEY (film, genere),
    CONSTRAINT "Appartenenza_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Appartenenza_genere_fkey" FOREIGN KEY (genere)
        REFERENCES public."Genere" (id_genere) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Sequel"
(
    precedente integer NOT NULL,
    successivo integer NOT NULL,
    CONSTRAINT "Sequel_pkey" PRIMARY KEY (precedente, successivo),
    CONSTRAINT "Sequel_precedente_fkey" FOREIGN KEY (precedente)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Sequel_successivo_fkey" FOREIGN KEY (successivo)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Recitazione"
(
    film integer NOT NULL,
    attore integer NOT NULL,
    CONSTRAINT "Recitazione_pkey" PRIMARY KEY (film, attore),
    CONSTRAINT "Recitazione_attore_fkey" FOREIGN KEY (attore)
        REFERENCES public."Attore" (id_personale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Recitazione_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Produzione"
(
    film integer NOT NULL,
    produttore integer NOT NULL,
    CONSTRAINT "Produzione_pkey" PRIMARY KEY (film, produttore),
    CONSTRAINT "Produzione_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Produzione_produttore_fkey" FOREIGN KEY (produttore)
        REFERENCES public."Produttore" (id_personale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Direzione"
(
    film integer NOT NULL,
    regista integer NOT NULL,
    CONSTRAINT "Direzione_pkey" PRIMARY KEY (film, regista),
    CONSTRAINT "Direzione_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Direzione_regista_fkey" FOREIGN KEY (regista)
        REFERENCES public."Regista" (id_personale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Distribuzione"
CREATE TABLE IF NOT EXISTS public."Distribuzione"
(
    film integer NOT NULL,
    distributore character(11) NOT NULL,
    CONSTRAINT "Distribuzione_pkey" PRIMARY KEY (film, distributore),
    CONSTRAINT "Distribuzione_film_fkey" FOREIGN KEY (film)
        REFERENCES public."Film" (id_film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Distribuzione_distributore_fkey" FOREIGN KEY (distributore)
        REFERENCES public."Distributore" (p_iva) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE "Fornitura"
(
    copia_film integer NOT NULL,
    film integer NOT NULL,
    capo character(16) NOT NULL,
    distributore character(11) NOT NULL,
    CONSTRAINT "Fornitura_pkey" PRIMARY KEY (copia_film, film),
    CONSTRAINT "Fornitura_capo_fkey" FOREIGN KEY (capo)
        REFERENCES public."Capo" (codice_fiscale) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Fornitura_copia_film_film_fkey" FOREIGN KEY (copia_film, film)
        REFERENCES public."CopiaFilm" (nr_copia, film) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Fornitura_distributore_fkey" FOREIGN KEY (distributore)
        REFERENCES public."Distributore" (p_iva) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

3.2 Creazione delle query

Op1: aggiunta di un negozio

```
CREATE OR REPLACE FUNCTION public.aggiungi_negozio(
        p_iva character,
        nome character varying,
        provincia character varying,
        citta character varying,
        via character varying,
        civico integer,
        capo character)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE

BEGIN
        INSERT INTO "Negozio" VALUES (p_iva, nome,
            provincia, citta, via, civico, capo);
END;
$BODY$;
```

Op2 stampa i dati relativi al negozio

```
CREATE OR REPLACE FUNCTION public.stampa_dati_negozio()
    RETURNS TABLE(_iva character,
    nome character varying,
    provincia character,
    citta character varying,
    via character varying,
    civico integer,
    capo character)
AS $BODY$
        SELECT * FROM "Negozio";
$BODY$
LANGUAGE sql VOLATILE
```

Op3: aggiunta di un cliente

```
CREATE OR REPLACE FUNCTION public.aggiungi_cliente(
        codice_fiscale character,
        nome character varying,
        cognome character varying,
        email character varying,
        telefono integer,
        provincia character,
        citta character varying,
        via character varying,
        civico integer)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_tessera integer;
BEGIN
        IF (SELECT count(*) from "Cliente")=0 THEN
                id_tessera:=1;
                ELSE id_tessera:=(SELECT max("Cliente".id_tessera)
                FROM "Cliente")+1;
        END IF;
        INSERT INTO "Cliente" VALUES(codice_fiscale, nome, cognome,
        email, telefono, provincia, citta, via, civico, 0, id_tessera);
        INSERT INTO "Iscrizione" VALUES(codice_fiscale, (SELECT p_iva
        FROM "Negozio"), CURRENT_TIMESTAMP);
END;
$BODY$;
```

Op4 visualizza saldo di un cliente

```
CREATE OR REPLACE FUNCTION public.visualizza_saldo(
        cliente character
        )
    RETURNS TABLE(
        saldo numeric
        )
AS $BODY$
        SELECT saldo FROM "Cliente" WHERE codice_fiscale=cliente;
$BODY$
LANGUAGE sql VOLATILE
```

Op5 aggiunta di una ricarica

```
CREATE OR REPLACE FUNCTION public.aggiungi_ricarica(
        cliente character,
        cassiere character,
        importo numeric
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_ric integer;
BEGIN
        IF (SELECT count(*) from "RicevutaFiscale")=0 THEN
                id_ric:=1;
                ELSE id_ric:=(SELECT max("RicevutaFiscale".id_ricevuta)
                from "RicevutaFiscale")+1;
        END IF;
        INSERT INTO "RicevutaFiscale" VALUES(id_ric, importo);
        INSERT INTO "Ricarica" VALUES(id_ric, cliente, cassiere);
        UPDATE "Cliente" SET saldo=saldo+importo WHERE codice_fiscale=cliente;
END;
$BODY$;
```

Op6 avvio di un noleggio

```
CREATE OR REPLACE FUNCTION public.aggiungi_noleggio(
        codice_fiscale character,
        film integer,
        supporto character varying
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        r record;
        nuovo_saldo numeric;
BEGIN
IF((SELECT saldo FROM "Cliente" WHERE
"Cliente".codice_fiscale=aggiungi_noleggio.codice_fiscale)>=1) THEN
 FOR r IN SELECT * FROM "CopiaFilm" LOOP
  IF ((SELECT COUNT(*) FROM "CopiaFilm"
   WHERE r.film=aggiungi_noleggio.film
        AND r.supporto=aggiungi_noleggio.supporto)>=1) THEN
        --RAISE NOTICE 'esiste una copia del film % con supporto
         %', r.film, r.supporto;
        if((SELECT COUNT(*) FROM "NoleggioCorrente"
         WHERE r.film="NoleggioCorrente".film
         AND r.nr_copia="NoleggioCorrente".nr_copia)=0) THEN
         --RAISE NOTICE 'non esiste un noleggio corrente della
          copia del film % numero % con supporto %', r.film,
            r.nr_copia, r.supporto;
         INSERT INTO "NoleggioCorrente" VALUES(r.nr_copia,
          r.film, codice_fiscale,
         CURRENT_TIMESTAMP);
         nuovo_saldo:=(SELECT saldo FROM "Cliente" WHERE
         "Cliente".codice_fiscale=aggiungi_noleggio.codice_fiscale);
         UPDATE "Cliente" SET saldo=(nuovo_saldo-1) WHERE
         "Cliente".codice_fiscale=aggiungi_noleggio.codice_fiscale;
         EXIT;
        END IF;
  END IF;
 END LOOP;
ELSE RAISE NOTICE 'saldo insufficiente';
END IF;
END;
$BODY$;
```

Op7 chiusura di un noleggio

```
CREATE OR REPLACE FUNCTION public.chiudi_noleggio(
        codice_fiscale character,
        film integer,
        nr_copia integer
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        noleggio record;
        costo integer;
BEGIN
        SELECT * FROM "NoleggioCorrente" INTO noleggio
                        WHERE "NoleggioCorrente".cliente=codice_fiscale
                        AND "NoleggioCorrente".film=chiudi_noleggio.film
                        AND "NoleggioCorrente".nr_copia=
                        chiudi_noleggio.nr_copia;
        RAISE NOTICE '%', noleggio;
        IF (noleggio IS NOT NULL) THEN
                RAISE NOTICE 'trovato noleggio';
                INSERT INTO "NoleggioPassato"
                VALUES(noleggio.cliente, noleggio.data,
                CURRENT_TIMESTAMP, noleggio.nr_copia, noleggio.film);
                costo=FLOOR(EXTRACT(EPOCH
                FROM CURRENT_TIMESTAMP-noleggio.data)/60/60/24);
                RAISE NOTICE 'costo: %', costo;
                UPDATE "Cliente" SET saldo=saldo-costo
                WHERE "Cliente".codice_fiscale=noleggio.cliente;
                DELETE FROM "NoleggioCorrente"
                WHERE "NoleggioCorrente".cliente=codice_fiscale
                        AND "NoleggioCorrente".film=chiudi_noleggio.film
                        AND "NoleggioCorrente".nr_copia=
                        chiudi_noleggio.nr_copia;
        END IF;
END;
$BODY$;
```

Op8 stampa storico noleggi relatico ad un cliente

```
CREATE OR REPLACE FUNCTION public.stampa_noleggi_passati(
        cliente character
        )
    RETURNS TABLE(film character varying)
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
        SELECT titolo FROM "NoleggioPassato"
        JOIN "Film" ON film=id_film WHERE
        "NoleggioPassato".cliente=stampa_noleggi_passati.cliente;
$BODY$;
```

Op9 stampa film disponibili per il noleggio

```
CREATE OR REPLACE FUNCTION public.stampa_film_disponibili()
    RETURNS TABLE(film character varying, id_film integer)
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
        SELECT titolo, id_film FROM "Film"
        WHERE id_film IN(SELECT film FROM "CopiaFilm"
        WHERE (film, nr_copia) NOT IN
        (SELECT film, nr_copia FROM "NoleggioCorrente"));
$BODY$;
```

Op10 stampa film disponibili per il noleggio relativi ad un attore

```
CREATE OR REPLACE FUNCTION public.stampa_film_disponibili_con_attore(
att character varying[])
    RETURNS TABLE(film1 character varying, id_film1 integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE

AS $BODY$
BEGIN
        RAISE NOTICE 'n % c %', att[1], att[2];
        RETURN QUERY
        SELECT titolo, id_film
        FROM "Film" JOIN "Recitazione" ON film=id_film
        JOIN "Attore" ON attore=id_personale
        WHERE att[1]=nome AND att[2]=cognome
        AND id_film IN(SELECT film
                                FROM "CopiaFilm"
                                WHERE (film, nr_copia) NOT IN
                                (SELECT film, nr_copia
                                FROM "NoleggioCorrente"));
END;
$BODY$;
```

Op11 stampa i film disponibili per il noleggio relativi ad un regista

```
CREATE OR REPLACE FUNCTION
public.stampa_film_disponibili_con_regista(
reg character varying[])
    RETURNS TABLE(film1 character varying, id_film1 integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE

AS $BODY$
BEGIN
        RETURN QUERY
        SELECT titolo, id_film
        FROM "Film" JOIN "Direzione" ON film=id_film
        JOIN "Regista" ON regista=id_personale
        WHERE reg[1]=nome AND reg[2]=cognome
        AND id_film IN(SELECT film
                                FROM "CopiaFilm"
                                WHERE (film, nr_copia)
                                NOT IN (SELECT film,
                                nr_copia FROM "NoleggioCorrente"));
END;
$BODY$;
```

Op12 stampa i film disponibili per il noleggio relativi ad un produttore

```
CREATE OR REPLACE FUNCTION public.stampa_film_disponibili_con_produttore(
pro character varying[])
    RETURNS TABLE(film1 character varying, id_film1 integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE

AS $BODY$
BEGIN
        RETURN QUERY
        SELECT titolo, id_film
        FROM "Film" JOIN "Produzione" ON film=id_film
        JOIN "Produttore" ON produttore=id_personale
        WHERE pro[1]=nome AND pro[2]=cognome
        AND id_film IN(SELECT film
                                FROM "CopiaFilm"
                                WHERE (film, nr_copia)
                                NOT IN (SELECT film,
                                nr_copia FROM "NoleggioCorrente"));
END;
$BODY$;
```

Op13 aggiunta di un cassiere

```
CREATE OR REPLACE FUNCTION public.aggiungi_cassiere(
        codice_fiscale character,
        nome character varying,
        cognome character varying,
        telefono integer,
        email character varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE

BEGIN
        INSERT INTO "Cassiere" VALUES (codice_fiscale, nome,
            cognome, telefono, email);
END;
$BODY$;
```

Op14: aggiunta di un capo

```
CREATE OR REPLACE FUNCTION public.aggiungi_capo(
        codice_fiscale character,
        nome character varying,
        cognome character varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE

BEGIN
        INSERT INTO "Capo" VALUES (codice_fiscale, nome, cognome);
END;
$BODY$;
```

Op15 aggiunta di una copia di un film

```
CREATE OR REPLACE FUNCTION public.aggiungi_copia_film(
        film integer,
        supporto character varying,
        distributore character)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        nr_copia integer;
        nr_copie_poss integer;
BEGIN
        IF(aggiungi_copia_film.distributore IN
           (SELECT "Distribuzione".distributore
                FROM "Distribuzione"
                WHERE aggiungi_copia_film.film="Distribuzione".film)) THEN
        nr_copie_poss := (SELECT COUNT(*) FROM "CopiaFilm"
        WHERE "CopiaFilm".film=aggiungi_copia_film.film);
        IF (nr_copie_poss)=0 THEN
                nr_copia:=1;
                ELSE nr_copia:=(SELECT MAX("CopiaFilm".nr_copia)
                FROM "CopiaFilm"
                WHERE "CopiaFilm".film=aggiungi_copia_film.film)+1;
        END IF;
        INSERT INTO "CopiaFilm" VALUES (nr_copia, film, supporto);
        nr_copie_poss := nr_copie_poss+1;
        UPDATE "Film" SET nr_copie_possedute=nr_copie_poss
        WHERE "Film".id_film=aggiungi_copia_film.film;
        INSERT INTO "Fornitura" VALUES(nr_copia, film,
        (SELECT capo FROM "Negozio"), distributore);
        END IF;
END;
$BODY$;
```

Op16 aggiunta di un film

```
CREATE OR REPLACE FUNCTION public.aggiungi_film(
        titolo character varying,
        trama character varying,
        anno integer,
        paese character varying,
        durata integer,
        attori character varying[][],
        registi character varying[][],
```

```
        produttori character varying[][],
        sottotitoli character varying[],
        doppiaggi character varying[],
        generi character varying[],
        lingua_originale character varying,
        distributore character varying,
        sequel character varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_film integer;
        attore character varying[];
        id_attore integer;
        regista character varying[];
        id_regista integer;
        produttore character varying[];
        id_produttore integer;
        linguaggio character varying;
        id_ling integer;
        id_distr integer;
        id_gen integer;
        genere character varying;
        id_sequel integer;
BEGIN
        IF (SELECT count(*) from "Film")=0 THEN
                id_film:=1;
                ELSE id_film:=(SELECT max("Film".id_film) from "Film")+1;
        END IF;
        INSERT INTO "Film" VALUES (id_film, titolo,
        trama, anno, paese, durata, 0);
        --aggiungi attore mancanti e recitazioni
        FOREACH attore SLICE 1 IN ARRAY attori
    LOOP
           RAISE NOTICE ' nome: % %', attore[1], attore[2];
           id_attore:=(SELECT id_personale FROM "Attore"
           WHERE nome=attore[1] AND cognome=attore[2]);
           IF (id_attore IS NOT NULL) THEN
                           RAISE NOTICE 'Attore già esistente';
                       INSERT INTO "Recitazione" VALUES(id_film, id_attore);
           ELSE
                           PERFORM aggiungi_attore(attore[1], attore[2]);
                       id_attore:=(SELECT id_personale FROM "Attore"
                       WHERE nome=attore[1] AND cognome=attore[2]);
```

```
                    INSERT INTO "Recitazione" VALUES(id_film, id_attore);
            END IF;
END LOOP;
    --aggiungi registi mancanti e recitazioni
    FOREACH regista SLICE 1 IN ARRAY registi
LOOP
       RAISE NOTICE ' nome: % %', regista[1], regista[2];
       id_regista:=(SELECT id_personale FROM "Regista"
       WHERE nome=regista[1] AND cognome=regista[2]);
       IF (id_regista IS NOT NULL) THEN
                       RAISE NOTICE 'Regista già esistente';
                    INSERT INTO "Direzione" VALUES(id_film, id_regista);
       ELSE
                       PERFORM aggiungi_regista(regista[1], regista[2]);
                    id_regista:=(SELECT id_personale FROM "Regista"
                    WHERE nome=regista[1] AND cognome=regista[2]);
                    INSERT INTO "Direzione"
                    VALUES(id_film, id_regista);
            END IF;
END LOOP;
    --aggiungi produttori mancanti e produzione
FOREACH produttore SLICE 1 IN ARRAY produttori
LOOP
       RAISE NOTICE ' nome: % %', produttore[1], produttore[2];
       id_produttore:=(SELECT id_personale FROM "Produttore"
       WHERE nome=produttore[1] AND cognome=produttore[2]);
       IF (id_produttore IS NOT NULL) THEN
                       RAISE NOTICE 'produttore già esistente';
                    INSERT INTO "Produzione" VALUES(id_film, id_produttore);
       ELSE
                       PERFORM aggiungi_produttore(
                       produttore[1], produttore[2]);
                    id_produttore:=(SELECT id_personale
                    FROM "Produttore" WHERE nome=produttore[1]
                    AND cognome=produttore[2]);
                    INSERT INTO "Produzione"
                    VALUES(id_film, id_produttore);
            END IF;
END LOOP;
    --aggiungi  linguaggio mancanti e sottototili
FOREACH linguaggio IN ARRAY sottotitoli
LOOP
       RAISE NOTICE ' linguaggio: %', linguaggio;
       id_ling:=(SELECT id_linguaggio FROM "Linguaggio"
       WHERE descrizione=linguaggio);
       IF (id_ling IS NOT NULL) THEN
```

```
                        RAISE NOTICE 'linguaggio già esistente';
                    INSERT INTO "Sottotitoli" VALUES(id_film, id_ling);
        ELSE
                        PERFORM aggiungi_linguaggio(linguaggio);
                    id_ling:=(SELECT id_linguaggio
                    FROM "Linguaggio"
                    WHERE descrizione=linguaggio);
                    INSERT INTO "Sottotitoli"
                    VALUES(id_film, id_ling);
             END IF;
 END LOOP;
     --aggiungi linguaggi mancanti e doppiaggi
 FOREACH linguaggio IN ARRAY doppiaggi
 LOOP
        RAISE NOTICE ' linguaggio: %', linguaggio;
        id_ling:=(SELECT id_linguaggio FROM "Linguaggio"
        WHERE descrizione=linguaggio);
        IF (id_ling IS NOT NULL) THEN
                        RAISE NOTICE 'linguaggio già esistente';
                    INSERT INTO "Doppiaggio" VALUES(id_film, id_ling);
        ELSE
                        PERFORM aggiungi_linguaggio(linguaggio);
                    id_ling:=(SELECT id_linguaggio FROM "Linguaggio"
                    WHERE descrizione=linguaggio);
                    INSERT INTO "Doppiaggio"
                    VALUES(id_film, id_linguaggio);
             END IF;
 END LOOP;

     --aggiungi linguaggi mancanti e lingua originale
id_ling:=(SELECT id_linguaggio FROM "Linguaggio"
WHERE descrizione=lingua_originale);
IF (id_ling IS NOT NULL) THEN
             RAISE NOTICE 'linguaggio già esistente';
             INSERT INTO "Originale"
             VALUES(id_film, id_ling);
ELSE
             PERFORM aggiungi_linguaggio(lingua_originale);
             id_ling:=(SELECT id_linguaggio FROM "Linguaggio"
             WHERE descrizione=lingua_originale);
             INSERT INTO "Originale"
             VALUES(id_film, id_ling);
     END IF;
     --aggiungi distributori mancanti e distribuzione
id_distr:=(SELECT p_iva FROM "Distributore"
WHERE nome=distributore);
```

```
    IF (id_distr IS NOT NULL) THEN
                RAISE NOTICE 'distributore già esistente';
                INSERT INTO "Distribuzione"
                VALUES(id_film, id_distr);
    ELSE
                PERFORM aggiungi_distributore(distributore);
                id_distr:=(SELECT p_iva FROM "Distributore"
                WHERE nome=distributore);
                INSERT INTO "Distribuzione"
                VALUES(id_film, id_distr);
        END IF;

        --aggiungi generi mancanti e appartenenza
     FOREACH genere IN ARRAY generi
     LOOP
            RAISE NOTICE ' genere: %', genere;
            id_gen:=(SELECT id_genere FROM "Genere"
            WHERE descrizione=genere);
            IF (id_gen IS NOT NULL) THEN
                            RAISE NOTICE 'genere già esistente';
                        INSERT INTO "Appartenenza"
                        VALUES(id_film, id_gen);
            ELSE
                            PERFORM aggiungi_genere(genere);
                        id_gen:=(SELECT id_genere FROM "Genere"
                        WHERE descrizione=genere);
                        INSERT INTO "Appartenenza"
                        VALUES(id_film, id_gen);
                END IF;
     END LOOP;

        --aggiungi generi mancanti e appartenenza
        IF(sequel IS NOT NULL) THEN
                id_sequel:=(SELECT "Film".id_film
                FROM "Film" WHERE "Film".titolo=sequel);
                IF(id_sequel IS NOT NULL) THEN
                        INSERT INTO "Sequel"
                        VALUES(id_sequel, id_film);
                END IF;
        END IF;
END;
$BODY$;
```

Op17 aggiunta di un linguaggio

```
CREATE OR REPLACE FUNCTION public.aggiungi_linguaggio(
        descrizione character varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_linguaggio integer;
BEGIN
        IF (SELECT count(*) from "Linguaggio")=0 THEN
                id_linguaggio:=1;
                ELSE id_linguaggio:=(SELECT max("Linguaggio".id_linguaggio)
                FROM "Linguaggio")+1;
        END IF;
        INSERT INTO "Linguaggio" VALUES (id_linguaggio, descrizione);
END;
$BODY$;
```

Op18 aggiunta di un genere

```
CREATE OR REPLACE FUNCTION public.aggiungi_genere(
        descrizione character varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_genere integer;
BEGIN
        IF (SELECT count(*) from "Genere")=0 THEN
                id_genere:=1;
                ELSE id_genere:=(SELECT max("Genere".id_genere)
                FROM "Genere")+1;
        END IF;
        INSERT INTO "Genere" VALUES (id_genere, descrizione);
END;
$BODY$;
```

Op19 aggiunta di un regista

```
CREATE OR REPLACE FUNCTION public.aggiungi_regista(
        nome character varying,
        cognome character varying
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_personale integer;
BEGIN
        IF (SELECT count(*) from "Regista")=0 THEN
                id_personale:=1;
                ELSE id_personale:=(SELECT max("Regista".id_personale)
                FROM "Regista")+1;
        END IF;
        INSERT INTO "Regista" VALUES (id_personale, nome, cognome);
END;
$BODY$;
```

Op20 aggiungi un attore

```
CREATE OR REPLACE FUNCTION public.aggiungi_attore(
        nome character varying,
        cognome character varying
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_personale integer;
BEGIN
        IF (SELECT count(*) from "Attore")=0 THEN
                id_personale:=1;
                ELSE id_personale:=(SELECT max("Attore".id_personale)
                FROM "Attore")+1;
        END IF;
        INSERT INTO "Attore" VALUES (id_personale, nome, cognome);
END;
$BODY$;
```

Op21 aggiunta di un produttore

```
CREATE OR REPLACE FUNCTION public.aggiungi_produttore(
        nome character varying,
        cognome character varying
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
        id_personale integer;
BEGIN
        IF (SELECT count(*) from "Produttore")=0 THEN
                id_personale:=1;
                ELSE id_personale:=(SELECT max("Produttore".id_personale)
                FROM "Produttore")+1;
        END IF;
        INSERT INTO "Produttore" VALUES (id_personale, nome, cognome);
END;
$BODY$;
```

Op22 aggiunta di un distributore

```
CREATE OR REPLACE FUNCTION public.aggiungi_distributore(
        p_iva character,
        nome character varying
        )
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE

BEGIN
        INSERT INTO "Distributore" VALUES (p_iva, nome);
END;
$BODY$;
```

Op23 stampa i dati relativi ad un film

```
CREATE OR REPLACE FUNCTION public.stampa_dati_film(
        tit character varying)
    RETURNS TABLE(id_film integer,
    titolo character varying,
    trama character varying,
    anno integer,
    paese character varying,
    durata integer,
    nr_copie_possedute integer)
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
        SELECT * FROM "Film" WHERE titolo=tit;
$BODY$;
```

4 Esecuzione delle operazioni principali

4.1 Aggiunta di un cliente(operazione 3)

```
select aggiungi_cliente('SPSLSN80A01G478G',
                'Alessandra',
                'Esposito',
                'alessandra.esposito@gmail.com',
                1231315633,
                'PG',
                'Perugia',
                'Via Roma',
                50)
```

Data Output

	codice_fiscale [PK] character (16)	nome character varying (30)	cognome character varying (30)	email character varying (30)	telefono integer	provincia character (2)	citta character varying (30)	via character varying (30)	civico integer	saldo numeric (19,2)	id_tessera integer
1	FRRFNC80A01G478W	Francesco	Ferrari	francesco.ferrari@gmail.com	1231315632	PG	Perugia	Via Roma	40	0.00	3
2	RSSLRD80A01G478R	Leonardo	Russo	leonardo.russo@gmail.com	1231315631	PG	Perugia	Via Roma	30	0.00	2
3	SPSLSN80A01G478G	Alessandra	Esposito	alessandra.esposito@gmail.c...	1231315633	PG	Perugia	Via Roma	50	0.00	4
4	VRDGPP80A01G478P	Giuseppe	Verdi	giuseppe.verdi@gmail.com	1231315630	PG	Perugia	Via Roma	20	0.00	1

Figure 12: OP3: modifica tabella cliente

Data Output

	cliente [PK] character (16)	negozio [PK] character (11)	data timestamp without time zone
1	FRRFNC80A01G478W	17293640173	2021-12-17 01:01:56.201418
2	RSSLRD80A01G478R	17293640173	2021-12-17 01:01:48.098688
3	SPSLSN80A01G478G	17293640173	2021-12-17 01:04:56.04667
4	VRDGPP80A01G478P	17293640173	2021-12-17 01:01:35.521515

Figure 13: OP3: modifica tabella iscrizione

4.2 Aggiunta di una ricarica(operazione 5)

```
select aggiungi_ricarica('FRRFNC80A01G478W','RMNGLI80A01G478P', 20)

select aggiungi_ricarica('RSSLRD80A01G478R','RMNGLI80A01G478P', 30)

select aggiungi_ricarica('SPSLSN80A01G478G','RMNGLI80A01G478P', 40)

select aggiungi_ricarica('VRDGPP80A01G478P','RMNGLI80A01G478P', 50)
```

Data Output

	codice_fiscale [PK] character (16)	nome character varying (30)	cognome character varying (30)	email character varying (30)	telefono integer	provincia character (2)	citta character varying (30)	via character varying (30)	civico integer	saldo numeric (19,2)	id_tessera integer
1	FRRFNC80A01G478W	Francesco	Ferrari	francesco.ferrari@gmail.com	1231315632	PG	Perugia	Via Roma	40	20.00	3
2	RSSLRD80A01G478R	Leonardo	Russo	leonardo.russo@gmail.com	1231315631	PG	Perugia	Via Roma	30	30.00	2
3	SPSLSN80A01G478G	Alessandra	Esposito	alessandra.esposito@gmail.com	1231315633	PG	Perugia	Via Roma	50	40.00	4
4	VRDGPP80A01G478P	Giuseppe	Verdi	giuseppe.verdi@gmail.com	1231315630	PG	Perugia	Via Roma	20	50.00	1

Figure 14: OP5: modifica tabella cliente

Data Output

	id_ricevuta [PK] integer	importo numeric (19,2)
1	1	20.00
2	2	30.00
3	3	40.00
4	4	50.00

Figure 15: OP5: modifica tabella ricevuta fiscale

Data Output

	ricevuta_fiscale [PK] integer	cliente character (16)	cassiere character (16)
1	1	FRRFNC80A01G478W	RMNGLI80A01G478P
2	2	RSSLRD80A01G478R	RMNGLI80A01G478P
3	3	SPSLSN80A01G478G	RMNGLI80A01G478P
4	4	VRDGPP80A01G478P	RMNGLI80A01G478P

Figure 16: OP5: modifica tabella ricarica

4.3 Aggiunta di un film(operazione 16)

```
select aggiungi_film('Quarto potere',
        'Nel castello di Candalù, in Florida, il vecchio magnate dell''editoria
        Charles Foster Kane è in punto di morte...',
        1941,
        'USA',
        119,
        '{{"Orson","Welles"},{"Joseph","Cotten"},{"Everett","Sloane"}}',
    '{{"Orson","Welles"}}',
        '{{"Orson","Welles"}}',
        '{"Italiano","Inglese","Francese"}',
        '{"Italiano","Francese"}',
        '{"Drammatico"}',
        'Inglese',
        'Mercury Theatre',
        NULL
        )
```

Data Output

	id_film [PK] integer	titolo character varying (30)	trama character varying (300)	anno integer	paese character varying (30)	durata integer	nr_copie_possedute integer
1	1	Il buono,il brutto,il cattivo	1862. Negli Stati Uniti infuria la guerra di sece...	1966	Italia	175	0
2	2	2001: Odissea nello spazio	Il film si compone di quattro sezioni distinte e...	1968	USA	141	0
3	3	Quarto potere	Nel castello di Candalù, in Florida, il vecchio ...	1941	USA	119	0

Figure 17: OP16: modifica tabella film

Data Output

	film [PK] integer	genere [PK] integer
1	1	1
2	2	2
3	3	3

Data Output

	id_genere [PK] integer	descrizione character varying (30)
1	1	Western
2	2	Fantascienza
3	3	Drammatico

Figure 18: OP16: modifica tabelle genere e appartenenza

Data Output

	film [PK] integer	attore [PK] integer
1	1	1
2	1	2
3	1	3
4	2	4
5	2	5
6	2	6
7	3	7
8	3	8
9	3	9

Data Output

	id_personale [PK] integer	nome character varying (30)	cognome character varying (30)
1	1	Clint	Eastwood
2	2	Eli	Wallach
3	3	Lee	Van Cleef
4	4	Keir	Dullea
5	5	Gary	Lockwood
6	6	William	Sylvester
7	7	Orson	Welles
8	8	Joseph	Cotten
9	9	Everett	Sloane

Figure 19: OP16: modifica tabelle attore e recitazione

Data Output

	film [PK] integer	produttore [PK] integer
1	1	1
2	2	2
3	3	3

Data Output

	id_personale [PK] integer	nome character varying (30)	cognome character varying (30)
1	1	Alberto	Grimaldi
2	2	Stanley	Kubrick
3	3	Orson	Welles

Figure 20: OP16: modifica tabelle produttore e produzione

Data Output

	film [PK] integer	regista [PK] integer
1	1	1
2	2	2
3	3	3

Data Output

	id_personale [PK] integer	nome character varying (30)	cognome character varying (30)
1	1	Sergio	Leone
2	2	Stanley	Kubrick
3	3	Orson	Welles

Figure 21: OP16: modifica tabelle regista e direzione

Data Output

	id_linguaggio [PK] integer	descrizione character varying (30)
1	1	Italiano
2	2	Inglese
3	3	Francese

Data Output

	film [PK] integer	linguaggio [PK] integer
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	2	3
7	3	1
8	3	2
9	3	3

Data Output

	film [PK] integer	linguaggio integer
1	1	2
2	2	2
3	3	2

Data Output

	film [PK] integer	linguaggio [PK] integer
1	1	1
2	1	3
3	2	1
4	2	3
5	3	1
6	3	3

Figure 22: OP16: modifica tabelle linguaggio, sottotitoli, doppiaggio e originale

Data Output

	film [PK] integer	distributore [PK] character (11)
1	1	1
2	2	5
3	3	3

Data Output

	p_iva [PK] character (11)	nome character varying (30)
1	1	PEA
2	3	Mercury Theatre
3	5	CIC

Figure 23: OP16: modifica tabelle distributore e distribuzione

4.4 Aggiunta di una copia film(operazione 15)

```
select aggiungi_copia_film(3, 'DVD', 'Mercury Theatre')
```

Data Output

	nr_copia [PK] integer	film [PK] integer	supporto character varying (30)
1	1	1	DVD
2	1	2	DVD
3	1	3	DVD
4	2	1	DVD
5	2	2	DVD
6	2	3	DVD
7	3	1	DVD
8	3	2	DVD
9	3	3	DVD
10	4	1	BlueRay
11	4	2	DVD
12	5	1	BlueRay
13	5	2	DVD
14	6	2	BlueRay

Figure 24: OP15: modifica tabella copia film

Data Output

	id_film [PK] integer	titolo character varying (30)	trama character varying (300)	anno integer	paese character varying (30)	durata integer	nr_copie_possedute integer
1	1	Il buono,il brutto,il cattivo	1862. Negli Stati Uniti infur...	1966	Italia	175	5
2	2	2001: Odissea nello spazio	Il film si compone di quattr...	1968	USA	141	6
3	3	Quarto potere	Nel castello di Candalù, in ...	1941	USA	119	3

Figure 25: OP15: modifica tabella film

Data Output

	copia_film [PK] integer	film [PK] integer	capo character (16)	distributore character (11)
1	1	1	RSSMRA80A01G478P	1
2	1	2	RSSMRA80A01G478P	5
3	1	3	RSSMRA80A01G478P	3
4	2	1	RSSMRA80A01G478P	1
5	2	2	RSSMRA80A01G478P	5
6	2	3	RSSMRA80A01G478P	3
7	3	1	RSSMRA80A01G478P	1
8	3	2	RSSMRA80A01G478P	5
9	3	3	RSSMRA80A01G478P	3
10	4	1	RSSMRA80A01G478P	1
11	4	2	RSSMRA80A01G478P	5
12	5	1	RSSMRA80A01G478P	1
13	5	2	RSSMRA80A01G478P	5
14	6	2	RSSMRA80A01G478P	5

Figure 26: OP15: modifica tabella fornitura

Data Output

	nr_copia [PK] integer	film [PK] integer	negozio character (11)
1	1	1	17293640173
2	1	2	17293640173
3	1	3	17293640173
4	2	1	17293640173
5	2	2	17293640173
6	2	3	17293640173
7	3	1	17293640173
8	3	2	17293640173
9	3	3	17293640173
10	4	1	17293640173
11	4	2	17293640173
12	5	1	17293640173
13	5	2	17293640173
14	6	2	17293640173

Figure 27: OP15: modifica tabella possesso2

4.5 Avvio di un noleggio(operazione 6)

```
select aggiungi_noleggio('VRDGPP80A01G478P',1,'DVD')
```

Data Output

	nr_copia [PK] integer	film [PK] integer	cliente character (16)	data timestamp with time zone
1	1	1	FRRFNC80A01G478W	2021-12-17 01:49:31.803034+01
2	1	2	FRRFNC80A01G478W	2021-12-17 01:49:35.945244+01
3	1	3	RSSLRD80A01G478R	2021-12-17 01:49:49.172744+01
4	2	1	RSSLRD80A01G478R	2021-12-17 01:49:52.912194+01
5	2	2	VRDGPP80A01G478P	2021-12-17 01:50:03.897064+01
6	3	1	VRDGPP80A01G478P	2021-12-17 01:50:06.514446+01

Figure 28: OP6: modifica tabella noleggio corrente

Data Output

	codice_fiscale [PK] character (16)	nome character varying (30)	cognome character varying (30)	email character varying (30)	telefono integer	provincia character (2)	citta character varying (30)	via character varying (30)	civico integer	saldo numeric (19,2)	id_tessera integer
1	FRRFNC80A01G478W	Francesco	Ferrari	francesco.ferrari@gmail.com	1231315632	PG	Perugia	Via Roma	40	18.00	3
2	RSSLRD80A01G478R	Leonardo	Russo	leonardo.russo@gmail.com	1231315631	PG	Perugia	Via Roma	30	28.00	2
3	SPSLSN80A01G478G	Alessandra	Esposito	alessandra.esposito@gmail.com	1231315633	PG	Perugia	Via Roma	50	40.00	4
4	VRDGPP80A01G478P	Giuseppe	Verdi	giuseppe.verdi@gmail.com	1231315630	PG	Perugia	Via Roma	20	48.00	1

Figure 29: OP6: modifica tabella cliente

4.6 Chiusura di un noleggio(operazione7)

```
select chiudi_noleggio('FRRFNC80A01G478W',1,1)
```

Data Output

	cliente [PK] character varying (16)	data_inizio [PK] timestamp with time zone	data_fine timestamp with time zone	nr_copia integer	film integer
1	FRRFNC80A01G478W	2021-12-17 01:49:31.803034+01	2021-12-17 01:54:31.930957...	1	1

Figure 30: OP7: modifica tabella noleggio passato

Data Output

	nr_copia [PK] integer	film [PK] integer	cliente character (16)	data timestamp with time zone
1	1	2	FRRFNC80A01G478W	2021-12-17 01:49:35.945244+01
2	1	3	RSSLRD80A01G478R	2021-12-17 01:49:49.172744+01
3	2	1	RSSLRD80A01G478R	2021-12-17 01:49:52.912194+01
4	2	2	VRDGPP80A01G478P	2021-12-17 01:50:03.897064+01
5	3	1	VRDGPP80A01G478P	2021-12-17 01:50:06.514446+01

Figure 31: OP7: modifica tabella noleggio corrente

Data Output

	codice_fiscale [PK] character (16)	nome character varying (30)	cognome character varying (30)	email character varying (30)	telefono integer	provincia character (2)	citta character varying (30)	via character varying (30)	civico integer	saldo numeric (19,2)	id_tessera integer
1	FRRFNC80A01G478W	Francesco	Ferrari	francesco.ferrari@gmail.com	1231315632	PG	Perugia	Via Roma	40	18.00	3
2	RSSLRD80A01G478R	Leonardo	Russo	leonardo.russo@gmail.com	1231315631	PG	Perugia	Via Roma	30	28.00	2
3	SPSLSN80A01G478G	Alessandra	Esposito	alessandra.esposito@gmail.com	1231315633	PG	Perugia	Via Roma	50	40.00	4
4	VRDGPP80A01G478P	Giuseppe	Verdi	giuseppe.verdi@gmail.com	1231315630	PG	Perugia	Via Roma	20	48.00	1

Figure 32: OP7: modifica tabella noleggio cliente

4.7 Stampa la lista dei film disponibili per il noleggio relativi ad un attore(operazione 10)

```
select stampa_film_disponibili_con_attore('{"Clint","Eastwood"}')
```

Data Output

	stampa_film_disponibili_con_attore record
1	("Il buono,il brutto,il cattivo",1)

Figure 33: OP10: stampa della lista dei film disponibili per il noleggio in cui ha recitato Clint Eastwood

4.8 Stampa i dati relativi ad un film(operazione 23)

```
select stampa_dati_film('Il buono,il brutto,il cattivo')
```

Figure 34: OP23: stampa dei dati relativi al film Il buono,il brutto,il cattivo