# CinemaDB

## Valerio Pro

**Settembre 2021**

## Indice

# 1 Analisi dei Requisiti e Raccolta delle Specifiche

## 1.1 Introduzione e Requisiti Generali

Si vuole realizzare una base di dati che gestisca i dati degli Studi Cinematografici Statunitensi al fine di avere a disposizione un archivio delle pellicole cinematografiche prodotte dagli studi stessi. Più nello specifico si è interessati non solo a tenere traccia di dati relativi alla produzione cinematografica, bensì anche di aspetti economici e di distribuzione delle pellicole. Non si è interessati a gestire dati di film "indipendenti", ovvero film la cui produzione non è direttamente affidata a studi cinematografici. Sarà importante classificare ciascuna pellicola sulla base di particolari parametri, come ad esempio il genere in cui rientra il film oppure il formato con cui sarà proiettato. Ci saranno dei vincoli che dovranno essere soddisfatti dai dati che descrivono le pellicole in modo da avere una rappresentazione fedele e non ambigua della realtà che verrà rappresentata. Come già anticipato, anche ciò che rappresenta il lato economico è oggetto di interesse, infatti, un altro importante obbiettivo che dovrà essere conseguito sarà quello di determinare con quali "brand" uno studio cinematografico ha stretto accordi per l'inclusione di prodotti a fini commerciali in specifiche pellicole. Al contrario dell'aspetto economico, la distribuzione dei film da parte delle Case di Produzione riguarderà la semplice conoscenza dei cinema e teatri che ricevono pellicole per la proiezione al pubblico. Inoltre si vuole anche tenere traccia degli attori che prendono parte ai film, memorizzando per ciascuno di essi delle informazioni sia di carattere generale che strettamente legate alla loro partecipazione nei film. Similmente si vogliono registrare dati sui registi che si occupano della direzione dei film durante la fase di produzione. Per queste ultime due figure si registreranno anche delle informazioni di "background" affinché ci possa essere una migliore definizione della loro carriera in questo campo. Si lascia aperta la possibilità per un regista di essere anche un attore (o viceversa) e questo è un altro importante aspetto che dovrà essere gestito. Volendo dare una breve anticipazione anche sulle principali operazioni di interesse su questo insieme di dati, si può dire che verrà posta particolare attenzione sull'elaborazione di informazioni riguardanti le spese e i guadagni degli studi cinematografici ma sarà anche fondamentale estrapolare informazioni statistiche (e non) riguardo registi e soprattutto attori. Ovviamente l'organizzazione dei dati dovrà favorire in particolar modo l'elaborazione di informazioni più semplici, quindi di informazioni che non derivano necessariamente da relazioni e legami di diversi oggetti.

## 1.2 Requisiti e Specifiche

Si passa adesso ad una descrizione più dettagliata degli oggetti e dei rispettivi requisiti che devono essere rappresentati e gestiti, partendo dagli oggetti che costituiranno il "nucleo" della base di dati.

I film sono rappresentati da una vasta serie di caratteristiche, sono identificati dal solo titolo della pellicola[1] e bisogna tenere conto della loro durata (in minuti) e della data di uscita. Inoltre per ciascuna pellicola si deve tenere traccia dell'incasso ai *box office* ovvero del totale guadagnato dalla sola vendita dei biglietti, del formato del film, che può essere 2D o 3D e del tipo di film cioè se si tratta di un Lungometraggio oppure di un Cortometraggio. Un cortometraggio è una pellicola la cui durata è al più pari a 30 minuti, mentre un lungometraggio è una pellicola di durata strettamente superiore a 30 minuti. Si vuole evitare che venga violato questo vincolo sulla durata e tipologia di un film. Si vuole inoltre conoscere il regista che si è occupato della regia del film. Infine si vuole classificare un film in base al genere della trama, non si vuole limitare questa caratterizzazione ad un solo genere dato che sarebbe troppo restrittiva come scelta ma allo stesso tempo non si vuole neanche che un film abbia più di 3 generi, altrimenti si avrebbe una caratterizzazione ambigua e non ragionevole.

Gli studi cinematografici sono identificati dal nome e si vuole sapere l'indirizzo (città + indirizzo stradale) dove si trova la sede dello studio e la quota del mercato cinematografico statunitense posseduta, in modo da evidenziare per ogni studio quella che è l'influenza all'interno del mercato. Ogni studio è responsabile della produzione di più film e un film sarà prodotto da uno e un solo studio.

Si vuole conoscere per ogni film prodotto, gli attori che hanno recitato nel film, il compenso che hanno ricevuto dalla casa di produzione per quel ruolo e anche che ruolo hanno ricoperto: protagonista, co-protagonista, personaggio secondario o comparsa. Si può assumere che in ogni film reciti almeno un attore.

Per quanto riguarda gli attori e i registi si vogliono sapere per entrambe le figure alcuni dati anagrafici, quali nome, cognome e data di nascita. Per gli attori è necessario tenere conto di alcune informazioni riguardo la loro carriera, come l'anno di esordio, cioè l'anno in cui hanno ufficialmente debuttato sul grande schermo; inoltre ogni attore deve essere classificato in base al numero di premi e riconoscimenti vinti grazie al loro mestiere, nello specifico ogni attore deve rientrare in una delle seguenti categorie:

- 10 o più riconoscimenti vinti;
- tra 7 e 9 riconoscimenti vinti;
- tra 4 e 6 riconoscimenti vinti;
- tra 1 e 3 riconoscimenti vinti;

[1] Un'assunzione che deriva dal fatto che i diritti delle opere cinematografiche sono protetti e difficilmente si incontrano due produzioni nominate allo stesso modo

- nessun riconoscimento vinto.

Per i registi invece si vuole semplicemente poter sapere il numero totale di film che hanno girato dall'inizio della loro carriera (sia che si tratti di film prodotti da uno studio che di film indipendenti[2]), in questo modo si può intuire il grado di esperienza di un regista. In merito alla questione della regia va sottolineato che un film è girato da un solo regista.

Si deve considerare e gestire anche il caso in cui un attore è anche regista (o viceversa).

Gli attori e i registi, per quanto riguarda proposte o contratti offerti dagli studi cinematografici, non sono mai in diretto contatto con le case di produzione stesse, bensì fanno sempre riferimento ad un' Agenzia di Consulenza, il cui ruolo è proprio quello di intermediario tra i due fronti. Risulta quindi necessario sapere per ogni regista e attore quale è l'agenzia che si occupa dell'aspetto manageriale del loro lavoro, riportandone il nome (identificativo per un' agenzia), l'indirizzo (città + indirizzo stradale) e il numero di telefono. Un regista/attore fa riferimento ad una sola agenzia, ma ad una stessa agenzia possono fare riferimento più registi/attori.

Dal punto di vista della distribuzione dovranno invece essere noti i cinema ai quali gli studi cinematografici distribuiscono le pellicole prodotte. Ogni cinema è identificato dal suo nome e si vuole anche avere l'indirizzo (città + indirizzo stradale), il numero delle sale e il prezzo standard del biglietto.

Gli studi cinematografici sono interessati a stringere accordi con le aziende per l'inserimento di prodotti a fini promozionali all'interno dei film. Non è detto però che ogni singolo film sia oggetto di contratto tra aziende e case di produzione. Si vuole allora conoscere con quali aziende gli studi firmano contratti, il guadagno per gli studi che proviene da tali contratti e si vuole sapere quali film riguardano. Un'azienda è identificata dal suo nome e si vogliono memorizzare solo informazioni che mostrino quanto un'azienda sia rilevante all'interno del mercato, per questo per ogni azienda coinvolta nella sponsorizzazione di prodotti in opere cinematografiche si vuole mantenere l'anno di fondazione e il net-worth.[3]

Si riportano di seguito alcune operazioni di interesse sugli oggetti che devono essere trattati:

- Sapere gli incassi totali dei film prodotti per ogni studio cinematografico;
- Sapere con quali aziende uno specifico studio cinematografico ha stretto accordi;

[2]Nonostante per i registi si voglia sapere un'informazione che racchiude il concetto di "film indipendente", non si devia comunque dal fatto che NON è obbiettivo della base di dati registrare informazioni specifiche (titolo, data di uscita, durata, ecc..) su tale categoria di film che non vengono direttamente prodotti da studi cinematografici

[3]patrimonio netto: valore di tutti i beni e attività finanziarie di un'azienda

- Sapere per ogni studio cinematografico quale è il contratto migliore che hanno stretto con le aziende;
- Sapere tutti gli attori che hanno lavorato almeno una volta insieme ad uno specifico attore;
- Sapere quali sono tutti i registi che sono anche attori;
- Essere in grado di stabilire delle classifiche di film in base agli incassi e al genere in cui rientrano;
- Sapere quali sono i cinema che ricevono pellicole dagli studi cinematografici più influenti.

### 1.3 Dati Relazionati

In questa sezione verranno esplorate le possibili relazioni che sussistono tra gli oggetti presentati precedentemente.

Le relazioni che legano i concetti sono piuttosto chiare, indubbiamente il concetto di film è strettamente legato sia al concetto di attore, dato che si vogliono raccogliere dati significativi sulle recitazioni, che al concetto di regista, poichè dovrà pur essere attribuita a qualcuno la regia di un film. C'è da notare che il concetto di regista più che porsi come una "informazione satellite" (inteso come semplice attributo) al concetto di film, sembra essere un concetto indipendente e di una certa importanza. Risulta banale la relazione tra un film e uno studio cinematografico in merito al processo di produzione stesso.

Altri due legami evidenti sono rispettivamente tra il concetto di studio cinematografico e il concetto di cinema e tra il concetto di agenzia e il concetto di attore/regista. Entrambe le relazioni sono necessarie, infatti, anche qui, i concetti di cinema e di agenzia non possono essere considerati "accessori" di concetti più grandi o importanti. Soprattutto il legame tra cinema e studio cinematografico ci darà informazioni sulla distribuzione dei prodotti.

Una relazione che invece è un po' più sottile è quella tra le aziende e gli studi cinematografici, in particolare la difficoltà potrebbe essere vista nel cercare di collegare uno studio cinematografico ad un'azienda e allo stesso tempo sapere anche per quali film (prodotti da quello stesso studio) l'azienda ha firmato contratti. Questo punto verrà argomentato nel dettaglio nel paragrafo seguente, dopo l'introduzione allo schema logico.

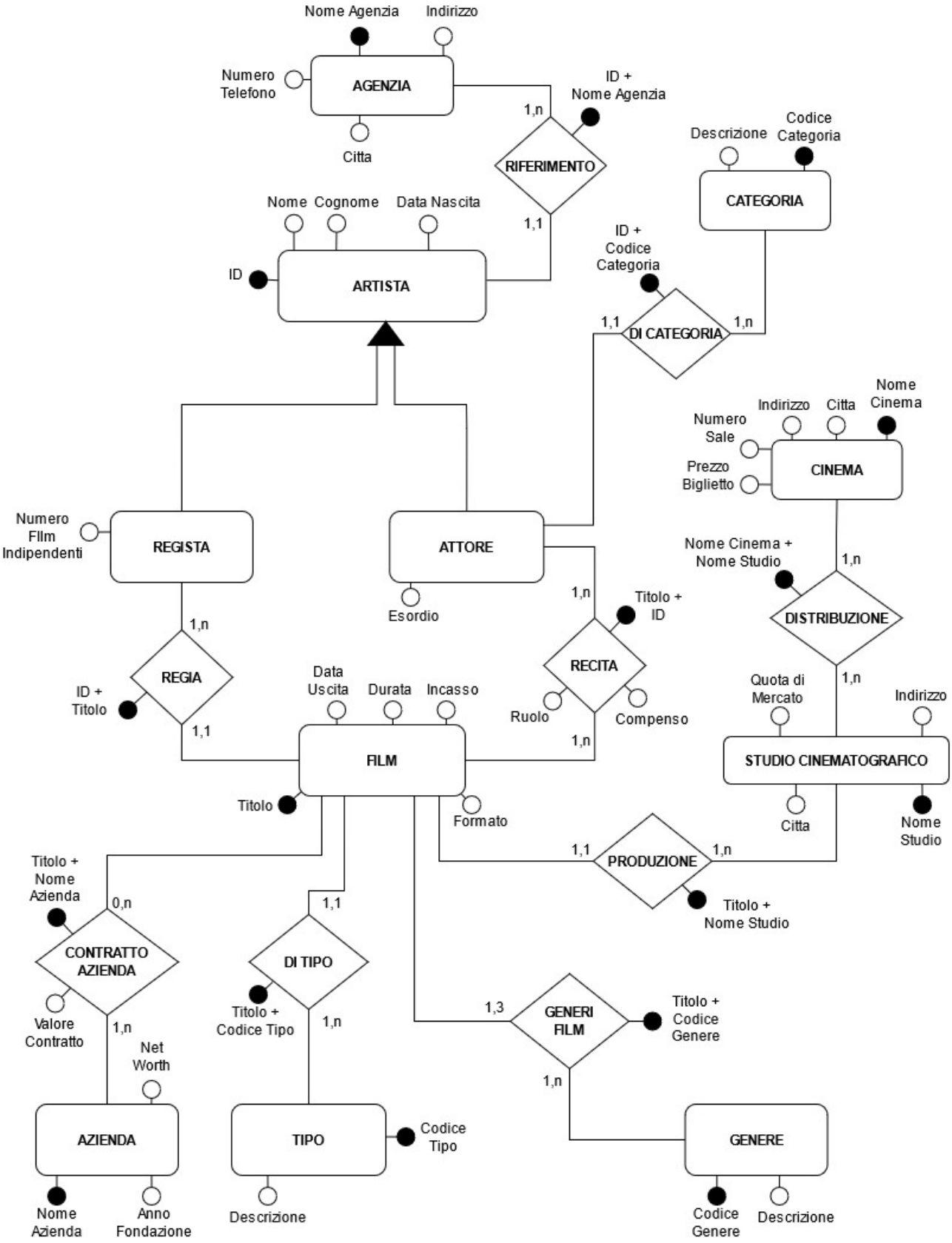
### 1.4 Glossario dei Termini e Regole Aziendali

Di seguito sono riportati il glossario dei termini e le regole aziendali basate sulle specifiche e sui requisiti indicati.

| **Glossario dei Termini** | | | |
|---|---|---|---|
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Studio Cinematografico | Si occupa della produzione e distribuzione di Film | Studio, Casa di Produzione | Film, Azienda, Cinema |
| Film | Opera prodotta da uno Studio Cinematografico, diretta da un regista e nella quale recitano più attori | Pellicola, Pellicola Cinematografica, Opera Cinematografica | Studio Cinematografico, Azienda, Attore, Regista |
| Attore | Persona che interpreta ruoli nei film | - | Film, Agenzia |
| Regista | Responsabile tecnico e artistico di un film | Direttore | Film, Agenzia |
| Cinema | Luogo adibito alla proiezione di pellicole ricevute dagli studi cinematografici | - | Studio Cinematografico |
| Agenzia | Offre consulenza agli artisti e media le relazioni con gli studi cinematografici | Agenzia di Consulenza | Attore, Regista |
| Azienda | Si interessa a firmare contratti con gli Studi Cinematografici per l'inserimento di prodotti nei Film | Brand | Studio Cinematografico, Film |

| **Regole Aziendali** | |
|---|---|
| **Regola** | **Descrizione** |
| (RA1) | Un Film di durata al più pari a 30 minuti deve essere considerato/registrato come *Cortometraggio*. Un Film di durata strettamente superiore a 30 minuti deve essere considerato/registrato come *Lungometraggio* |
| (RA2) | Un Film non può essere caratterizzato da più di 3 generi |

## 2 Schema E-R Logico



### 2.1 Dizionario dei Dati

| Dizionario delle Entità | | | |
|---|---|---|---|
| **Entità** | **Descrizione** | **Attributi** | **Identificatore** |
| Film | Opera Cinematografica in cui recitano attori, prodotta da uno Studio Cinematografico e diretta da un regista | Titolo, Data Uscita, Durata, Incasso, Formato | Titolo |
| Studio Cinematografico | Produce Film, distribuisce pellicole e firma contratti con le aziende | Nome Studio, Città, Indirizzo, Quota di Mercato | Nome Studio |
| Artista | Categoria in cui rientrano Attori e Registi | ID, Nome, Cognome, Data Nascita | ID |
| Attore | Viene pagato per interpretare ruoli nei film | ID, Esordio | ID |
| Regista | Si occupa della regia dei film | ID, Numero Film Indipendenti | ID |
| Agenzia | Si occupa di gestire i rapporti con gli studi cinematografici per gli attori e per i registi | Nome Agenzia, Città, Indirizzo, Numero Telefono | Nome Agenzia |
| Cinema | Riceve le pellicole distribuite dagli studi cinematografici | Nome Cinema, Città, Indirizzo, Numero Sale, Prezzo Biglietto | Nome Cinema |
| Azienda | Stringe accordi con gli studi cinematografici per l'inclusione di prodotti nei film | Nome Azienda, Anno Fondazione, Net Worth | Nome Azienda |
| Genere | Raccoglie i generi che possono avere i film | Codice Genere, Descrizione | Codice Genere |
| Tipo | Raccoglie le possibili tipologie di Film | Codice Tipo, Descrizione | Codice Tipo |
| Categoria | Raccoglie le categorie in cui possono rientrare gli attori | Codice Categoria, Descrizione | Codice Categoria |

Anche se nello schema logico le entità *Attore* e *Regista* non presentano una chiave per via della generalizzazione, ho voluto specificare nel dizionario dei dati che come chiave hanno "ID", anticipando quindi che erediteranno la chiave dall'entità genitore.

| Dizionario delle Relazioni | | | |
|---|---|---|---|
| **Relazione** | **Descrizione** | **Entità Coinvolte** | **Attributi** |
| Recita | Associa agli attori i film in cui recitano | Attore (1,n), Film (1,n) | Ruolo, Compenso |
| Contratto Azienda | Associa ai film sponsorizzati le aziende responsabili | Film (0,n), Azienda (1,n) | Valore Contratto |
| Generi Film | Associa ad un film i generi che lo caratterizzano | Film (1,3), Genere(1,n) | - |
| Distribuzione | Associa ai cinema gli studi che distribuiscono pellicole verso quei cinema | Cinema (1,n), Studio Cinematografico (1,n) | - |
| Riferimento | Associa ad un'artista l'agenzia cui fa riferimento | Artista (1,1), Agenzia (1,n) | - |
| Di Categoria | Associa ad un attore la categoria cui appartiene | Attore (1,1), Categoria (1,n) | - |
| Regia | Associa ad un film il regista che lo ha girato | Film (1,1), Regista (1,n) | - |
| Di Tipo | Associa ad un film la sua tipologia | Film (1,1), Tipo (1,n) | - |
| Produzione | Associa ad un film lo studio cinematografico che lo ha prodotto | Film (1,1), Studio Cinematografico (1,n) | - |

| Regole di Vincolo | |
|---|---|
| **Regola** | **Descrizione** |
| (RV1) | Un Film di durata al più pari a 30 minuti deve essere considerato/registrato come *Cortometraggio*. Un Film di durata strettamente superiore a 30 minuti deve essere considerato/registrato come *Lungometraggio* |
| (RV2) | Un Film non può essere caratterizzato da più di 3 generi |

### 2.2 Osservazioni sulle Entità

- **Categoria:** L'entità *Categoria* raccoglie, per l'appunto, le categorie in cui può ricadere un attore in base al numero di premi vinti. La chiave è "Codice Categoria" che sarà semplicemente una lettera (S, A, B, C o D) e abbiamo una descrizione per ciascuna chiave che rispecchia l'elenco riportato nel paragrafo 1 sulle categorie di attori. Si tratta di un'entità che nel tempo potrebbe cambiare poco, il suo scopo è semplice e un altro motivo per la sua presenza riguarda il fatto che è meglio avere 5 righe VARCHAR() in una tabella separata rispetto ad avere migliaia di righe VARCHAR() nell'entità *Attore*; con un grande numero di record memorizzati nella tabella dell'entità *Attore* il risparmio di memoria può diventare consistente.

- **Artista:** La generalizzazione con entità genitore *Artista* ha un motivo ben preciso per essere presente, permette infatti di evitare un minimo di ridondanza e possibili incogruenze dei dati a causa di attori che sono anche registi. Ho dedicato un intero sottoparagrafo alla generalizzazione e più avanti verrà spiegato in dettaglio il suo scopo, le sue caratteristiche e la ristrutturazione adottata. Per adesso si può dire che l'entità *Artista* raccoglie le informazioni più generali ed è in relazione con l'entità *Agenzia*, poiché sia attori che registi fanno riferimento ad un'agenzia.

- **Regista:** Come riportato nelle specifiche richieste, per ogni regista si vuole sapere il numero totale di film girati, sia indipendenti che prodotti da uno studio. Durante la progettazione mi sono però accorto che utilizzare un attributo del tipo "numero_film_totali_girati" nell'entità *Regista* avrebbe potuto portare a delle ambiguità o incongruenze nei dati. Questo perchè, se si utilizza un attributo di questo tipo, avremo: $numero\_film\_totali\_girati = numero\_film\_indipendenti + numero\_film\_girati\_prodotti\_da\_studio$, senza però sapere quale è effettivamente la quantità precisa di *numero_film_indipendenti* e di *numero_film_girati_prodotti_da_studio* e questo porta solamente a dei problemi, ad esempio, cosa succede quando nell'entità *Film* il numero di film girati da uno stesso regista diventa maggiore o uguale del valore dell'attributo $numero\_film\_totali\_girati$ per quello stesso regista? Come dovrebbe essere aggiornato tale attributo se non si conoscono effettivamente le quantità precise di film indipendenti girati?
  A tal proposito ho preferito separare i due concetti e non lasciare che vengano inclusi in uno stesso attributo, nell'entità *Regista* è riportato quindi il solo attributo $numero\_film\_indipendenti$, dato che il numero di film girati e prodotti da uno studio può essere estrapolato dall'entità *Film*. Potendo ricavare quest'informazione da un'interrogazione, ho ritenuto che riportarla anche come attributo sarebbe stato semplicemente ridondante (e ci sarebbe stata anche necessità di tenerlo aggiornato con inserimenti di nuovi film).

- **Attore**: L'entità attore è caratterizzata dall'attributo "esordio" che indica l'anno di esordio di un attore ed è in relazione uno-a-molti con l'entità

categoria, al fine di classificare ogni singolo attore.

- **Genere:** L'entità *Genere* descrive tutti i possibili generi che un film può possedere, è identificata da un codice al quale è abbinata una descrizione ($'H' \rightarrow' Horror'$, $'M' \rightarrow' Musical'$, *ecc...*). Per una migliore normalizzazione ho preferito racchiudere il concetto di genere in un'entità piuttosto che avere un attributo multivalore nell'entità *Film*.

- **Tipo:** L'entità *Tipo* è un'entità semplice, conterrà due tuple (una per "Lungometraggio" e una per "Cortometraggio") e la sua esistenza è dovuta al fatto che ho preferito evitare di avere molte stringhe nella tabella dell'entità film che descrivessero per esteso il tipo di film. Si sarebbe trattato comunque di un "attributo binario" per l'entità film, ma comunque ho preferito racchiudere i due tipi in un'entità per avere a livello fisico un campo "tipo" CHAR(1) nella tabella dell'entità film. Sempre riguardo l'entità *Tipo* avevo pensato di includere in quest'entità anche l'attributo "formato" e avere poi come campo chiave delle combinazioni tra "tipo" e "formato" ($'L2D' \rightarrow\ 'Lungometraggio\ 2D'$, $'C3D' \rightarrow 'Cortometraggio\ 3D'$, *ecc...*). Tuttavia dal punto di vista della normalizzazione non ho pensato fosse la cosa migliore aggregare più attributi (con il passaggio allo schema fisico, nell'entità film avrei avuto questo attributo di informazioni aggregate).

- **Film:** L'entità *Film* è una delle entità più importanti dello schema, racchiude molti attributi che rispecchiano tutti le specifiche richieste. Se nelle specifiche veniva indicato che un film poteva essere associato ad un genere soltanto allora avremmo avuto una relazione con cardinalità 1,1 dal lato dell'entità *Film* verso l'entità *Genere*, questo si sarebbe tradotto poi nello schema fisico in un riferimento diretto all'entità *Genere*, tuttavia viene lasciata più libertà nella caratterizzazione dei film, ad un film può essere associato un massimo di 3 generi.

- **Studio Cinematografico, Cinema, Agenzia, Azienda:** queste quattro entità rispecchiano semplicemente il concetto che si voleva rappresentare descritto nell'analisi dei requisiti.

### 2.3 Generalizzazione

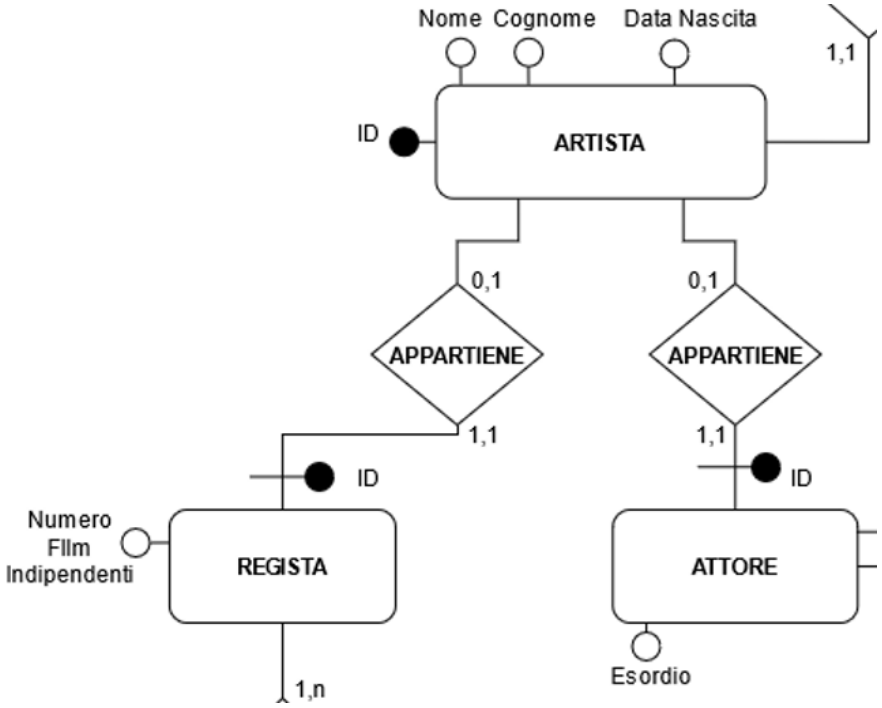
A causa della possibilità di un attore di essere anche regista ho ritenuto che utilizzare una generalizzazione fosse una buona scelta. Se lo schema fosse privo di generalizzazione e quindi se le due entità attore e regista non facessero riferimento all'entità genitore *Artista*, sarebbero sorti principalmente due problemi:

1) Ci sarebbe stata ridondanza nei dati per tutti quegli attori che sono anche registi. Infatti avremmo avuto una serie di stessi dati ripetuti in tabelle di entità diverse, quali nome, cognome, data di nascita (anche il nome dell'agenzia dopo il passaggio al fisico).

2) Ci sarebbe stata possibilità di incorrere in situazioni di incogruenza dei dati, supponendo sempre di avere le due entità *Attore* e *Regista* separate, allora se si vuole aggiungere nella tabella dell'entità *Attore* una riga con chiave identica alla chiave di una riga già presente nell'entità *Regista* (quindi stiamo dicendo che vogliamo memorizzare un attore che è anche regista) allora, in assenza di vincoli concreti, si sarebbe potuto assegnare, ad esempio, una data di nascita diversa. Ancor peggio, se in questa situazione come chiave si usa un intero, sarebbe stato possibile dichiarare il nome o il cognome diversamente da come è riportato nell'altra tabella separata.

La generalizzazione è di tipo *Totale e Sovrapposta*, perchè può esserci una tupla in *Regista* e una tupla in *Attore* che entrambe fanno riferimento alla stessa tupla di *Artista*. Trasformare la generalizzazione da *Sovrapposta* ad *Esclusiva* aggiungendo un'entità figlia *"Regista e Attore"* non è la scelta migliore, nonostante fornirebbe una migliore classificazione dell'oggetto *Artista*, renderebbe lo schema molto più complicato, come fare per esempio se un film fosse diretto da un regista che è anche un attore? La traduzione nello schema logico di questa osservazione ne complicherebbe molto la struttura.
Una possibile soluzione sarebbe quella di introdurre comunque l'entità *"Regista e Attore"*, cambiando però il modo in cui sono relazionate le entità: la relazione *Regia* dovrebbe legare *Film* e *Artista* (con un opportuno cambio di cardinalità in 0,n dal lato di *Artista*) e la relazione *Recita* dovrebbe legare *Film* e *Artista* (con un opportuno cambio di cardinalità in 0,n dal lato di Artista). Tuttavia non ritengo questa soluzione ottimale, perchè nella tabella dell'entità *Film* potrebbe essere fatto un riferimento ad un artista che non è in realtà un regista e mi sembra poco robusta come strada da prendere.

Per quanto riguarda la ristrutturazione della generalizzazione ho scelto la seguente soluzione.




In virtù di quanto è stato già detto mi è sembrata la scelta migliore ristrutturarla in questo modo. Le entità *Regista* e *Attore* erediteranno quindi l'identificativo dell'entità *Artista*.

Uno svantaggio di questa ristrutturazione è che aumenta il numero degli accessi per accedere ad alcune informazioni.

### 2.4 Osservazioni sulle Relazioni

- **Recita:** La relazione *Recita* collega *Film* e *Attore*, si ricorda che siamo sotto l'assunzione che in ogni film recita almeno un attore. Inoltre teniamo traccia solo di attori che effettivamente hanno recitato in almeno uno dei film memorizzati. La relazione oltre ad avere la chiave composta ha due altri attributi: "Ruolo" (che riporta il ruolo del rispettivo attore nel film con titolo in questione) e "Compenso" (che mostra quanto sia stato pagato l'attore per quel ruolo in quello specifico film). Non ho ritenuto necessario promuovere la relazione in un'entità chiamata ad esempio "Performance" o "Recitazione". Grazie alla semplice relazione *Recita* si riescono a mantenere tutte le informazioni di interesse.

- **Generi_Film**: Questa relazione mantiene l'associazione tra un film e i generi che lo contraddistinguono, come evidenziato dalle cardinalità il titolo di un film può occorrere al più 3 volte nelle tuple della relazione. Nel paragrafo 8 si cercherà di rendere quanto più robusta possibile questa condizione.

- **Distribuzione:** Si tratta di una relazione molto semplice che permette di capire a quali cinema vengono distribuite le pellicole da parte degli studi cinematografici.

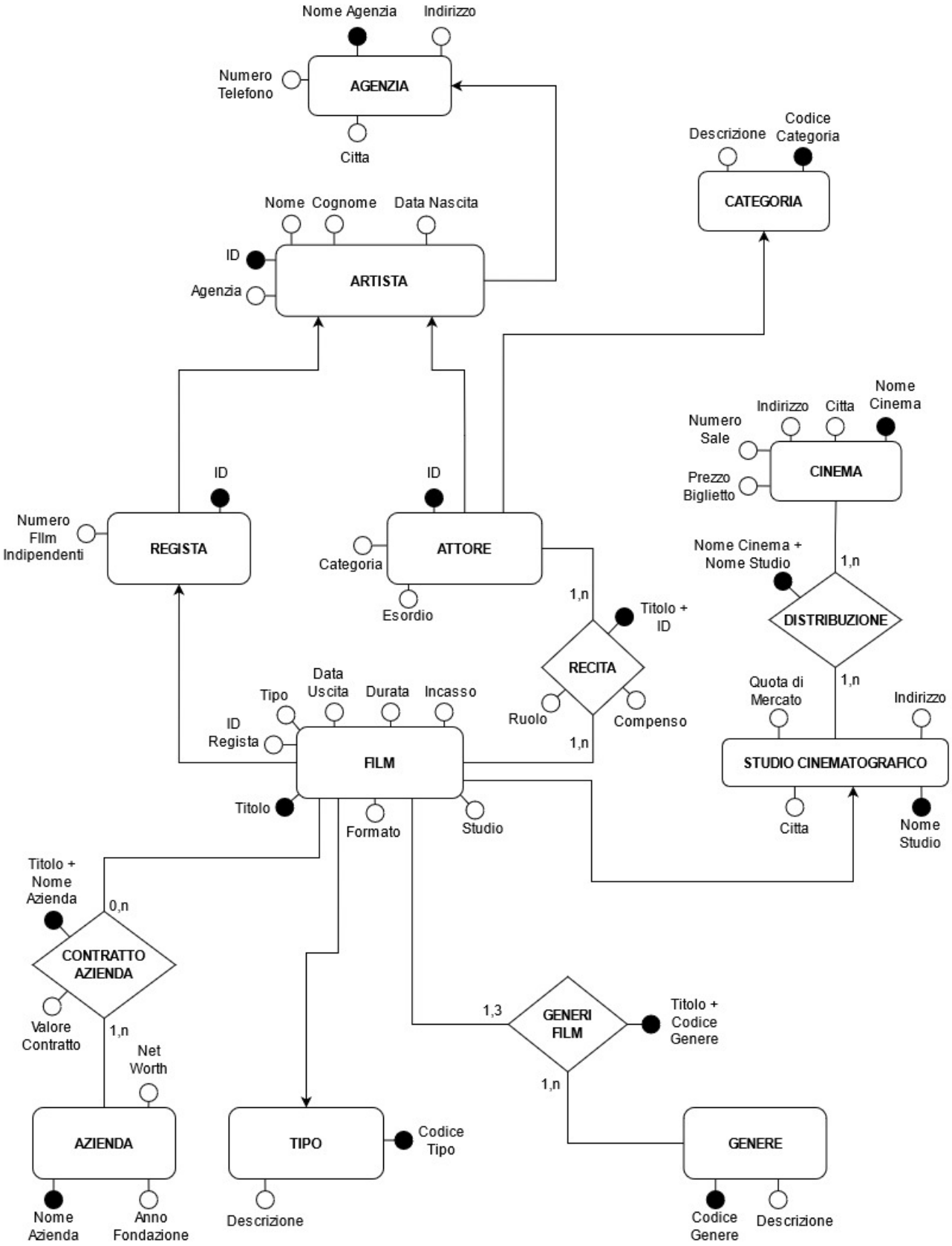
- **Contratto_Azienda:** Attraverso questa relazione si tiene traccia delle sponsorizzazioni all'interno dei film, l'attributo "valore_contratto" mostra quanto un'azienda abbia pagato per l'inclusione di prodotti in uno specifico film. Attraverso la navigazione della base di dati sarà possibile risalire agli studi cinematografici, anche se questi non sono direttamente collegati all'entità *Azienda*. Si ricorda che non è sempre vero che in un film siano inseriti prodotti a fini promozionali, ecco perchè la cardinalità 0,n dal lato dell'entità *Film*.

Le restanti relazioni sono di tipo *uno-a-molti*, nel passaggio allo schema fisico "scompariranno" e verranno ereditati gli attributi delle entità puntate.

### 2.5 Chiave Semplice contro Chiave Composta

In questa sezione vorrei argomentare la scelta dell'identificatore per l'entità *Artista* e quindi anche per l'entità *Regista* e *Attore*. La scelta riguardava due possibilità principalmente: avere un identificatore composto *nome* + *cognome* oppure avere un identificatore semplice intero in *auto_increment*. Come mostrato nello *Schema Logico* e nel *Dizionario dei Dati* ho adottato un identificatore semplice intero, questo perchè, con la ristrutturazione della generalizzazione, il numero di accessi per accedere a specifiche informazioni è aumentato, se si avesse anche una chiave composta le operazioni di *join* sarebbero ulteriormente appesantite per tutto il ramo superiore dello schema. Ho cercato quindi di bilanciare la situazione con un identificatore intero, tuttavia anche questa scelta presenta i suoi svantaggi, ad esempio, guardando alla relazione *Recita*, se si vogliono sapere delle informazioni come nome e cognome è necessario risalire fino all'entità *Artista*, con una chiave composta invece si sarebbe potuto gestire tutto a livello della sola relazione. La scelta dell'identificatore intero mi è sembrata la migliore tra le due, come se "rafforzasse" l'applicazione della generalizzazione.

# 3 Schema E-R Fisico-Normalizzato



### 3.1 Osservazioni

Nel passaggio allo *Schema Fisico* le relazioni *uno-a-molti* lasciano il posto ai riferimenti diretti tra i campi delle entità. Vorrei fare delle precisazioni sul cambio di nome per alcuni attributi appena acquisiti dalle entità con il passaggio allo schema fisico. Ho effettuato i seguenti cambi per una maggiore leggibilità:

- **Entità Film:** Nome dell'attributo "Codice Tipo" cambiato in "Tipo".
  Nome dell'attributo "ID" cambiato in "ID Regista".
  Nome dell'attributo "Nome Studio" cambiato in "Studio".
- **Entità Artista:** Nome dell'attributo "Nome Agenzia" cambiato in "Agenzia".
- **Entità Attore:** Nome dell'attributo "Codice Categoria" cambiato in "Categoria".

### 3.2 Schemi di Relazione

STUDIO_CINEMATOGRAFICO(NomeStudio, Città, Indirizzo, QuotaDiMercato)

FILM(Titolo, DataUscita, Durata, Incasso, Studio, IdRegista, Tipo, Formato)

ARTISTA(Id, Nome, Cognome, DataNascita, Agenzia)

ATTORE(Id, Esordio, Categoria)

REGISTA(Id, NumeroFilmIndipendenti)

AGENZIA(NomeAgenzia, Città, Indirizzo, NumeroTelefono)

CATEGORIA(CodiceCategoria, Descrizione)

TIPO(CodiceTipo, Descrizione)

GENERE(CodiceGenere, Descrizione)

AZIENDA(NomeAzienda, AnnoFondazione, NetWorth)

CINEMA(NomeCinema, Città, Indirizzo, NumeroSale, PrezzoBiglietto)

RECITA(Titolo, Id, Compenso, Ruolo)

DISTRIBUZIONE(NomeCinema, NomeStudio)

GENERI_FILM(Titolo, CodiceGenere)

CONTRATTO_AZIENDA(Titolo, NomeAzienda, ValoreContratto)

## 4 Creazione Tabelle

Di seguito le istruzioni *Create* per le 15 tabelle necessarie.

```
CREATE TABLE Categoria (codice_categoria CHAR(1) NOT NULL,
                        descrizione VARCHAR(32) NOT NULL,
                        PRIMARY KEY(codice_categoria)) ENGINE=InnoDB;

CREATE TABLE Tipo (codice_tipo CHAR(1) NOT NULL,
                   descrizione VARCHAR(16) NOT NULL,
                   PRIMARY KEY(codice_tipo)) ENGINE=InnoDB;

CREATE TABLE Genere (codice_genere VARCHAR(3) NOT NULL,
                     descrizione VARCHAR(16) NOT NULL,
                     PRIMARY KEY(codice_genere)) ENGINE=InnoDB;

CREATE TABLE Agenzia (nome_agenzia VARCHAR(32) NOT NULL,
                      citta VARCHAR(24),
                      indirizzo VARCHAR(32),
                      numero_telefono CHAR(10) NOT NULL,
                      PRIMARY KEY(nome_agenzia)) ENGINE=InnoDB;


CREATE TABLE Studio_Cinematografico (nome_studio VARCHAR(24) NOT NULL,
                                     citta VARCHAR(24),
                                     indirizzo VARCHAR(32),
                                     quota_di_mercato DECIMAL(3,2),
                                     PRIMARY KEY(nome_studio)) ENGINE=InnoDB;


CREATE TABLE Artista (id INT UNSIGNED NOT NULL AUTO_INCREMENT,
                      nome VARCHAR(16) NOT NULL,
                      cognome VARCHAR(16) NOT NULL,
                      data_nascita DATE,
                      agenzia VARCHAR(32) NOT NULL,
                      FOREIGN KEY(agenzia) REFERENCES Agenzia(nome_agenzia)
                      ON UPDATE CASCADE ON DELETE NO ACTION,
                      PRIMARY KEY(id)) ENGINE = InnoDB;


CREATE TABLE Regista (id INT UNSIGNED NOT NULL,
                      FOREIGN KEY(id) REFERENCES Artista(id)
                      ON UPDATE CASCADE ON DELETE NO ACTION,
                      numero_film_indipendenti TINYINT UNSIGNED NOT NULL,
                      PRIMARY KEY(id)) ENGINE = InnoDB;
```

```
CREATE TABLE Attore (id INT UNSIGNED NOT NULL,
                     FOREIGN KEY(id) REFERENCES Artista(id)
                     ON UPDATE CASCADE ON DELETE NO ACTION,
                     esordio YEAR NOT NULL,
                     categoria CHAR(1) NOT NULL,
                     FOREIGN KEY(categoria) REFERENCES Categoria(codice_categoria)
                     ON UPDATE CASCADE ON DELETE NO ACTION,
                     PRIMARY KEY(id)) ENGINE = InnoDB;

CREATE TABLE Film (titolo VARCHAR(32) NOT NULL,
                   data_uscita DATE NOT NULL,
                   durata TINYINT UNSIGNED NOT NULL,
                   incasso INT UNSIGNED NOT NULL,
                   studio VARCHAR(24) NOT NULL,
                   FOREIGN KEY(studio) REFERENCES
                   Studio_Cinematografico(nome_studio)
                   ON UPDATE CASCADE ON DELETE NO ACTION,
                   id_regista INT UNSIGNED NOT NULL,
                   FOREIGN KEY(id_regista) REFERENCES Regista(id)
                   ON UPDATE CASCADE ON DELETE NO ACTION,
                   tipo CHAR(1) NOT NULL,
                   FOREIGN KEY(tipo) REFERENCES Tipo(codice_tipo)
                   ON UPDATE CASCADE ON DELETE NO ACTION,
                   formato CHAR(2) NOT NULL,
                   PRIMARY KEY(titolo)) ENGINE = InnoDB;


CREATE TABLE Cinema (nome_cinema VARCHAR(24) NOT NULL,
                     citta VARCHAR(24),
                     indirizzo VARCHAR(32),
                     numero_sale TINYINT NOT NULL,
                     prezzo_biglietto DECIMAL(4,2),
                     PRIMARY KEY(nome_cinema)) ENGINE=InnoDB;


CREATE TABLE Azienda (nome_azienda VARCHAR(32) NOT NULL,
                      anno_fondazione YEAR NOT NULL,
                      net_worth BIGINT UNSIGNED NOT NULL,
                      PRIMARY KEY(nome_azienda)) ENGINE=InnoDB;
```

```
CREATE TABLE Generi_Film (titolo VARCHAR(32) NOT NULL,
                          codice_genere VARCHAR(3) NOT NULL,
                          FOREIGN KEY(titolo) REFERENCES Film(titolo)
                          ON UPDATE CASCADE ON DELETE CASCADE,
                          FOREIGN KEY(codice_genere) REFERENCES
                          Genere(codice_genere)
                          ON UPDATE CASCADE ON DELETE CASCADE,
                          PRIMARY KEY(titolo, codice_genere)) ENGINE = InnoDB;

CREATE TABLE Distribuzione (nome_studio VARCHAR(24) NOT NULL,
                            nome_cinema VARCHAR(24) NOT NULL,
                            FOREIGN KEY(nome_studio) REFERENCES
                            Studio_Cinematografico(nome_studio)
                            ON UPDATE CASCADE ON DELETE CASCADE,
                            FOREIGN KEY(nome_cinema) REFERENCES Cinema(nome_cinema)
                            ON UPDATE CASCADE ON DELETE CASCADE,
                            PRIMARY KEY(nome_studio, nome_cinema)) ENGINE=InnoDB;

CREATE TABLE Recita (titolo VARCHAR(32) NOT NULL,
                     id INT UNSIGNED NOT NULL,
                     compenso INT UNSIGNED NOT NULL,
                     ruolo VARCHAR(16) NOT NULL,
                     FOREIGN KEY(titolo) REFERENCES Film(titolo)
                     ON UPDATE CASCADE ON DELETE CASCADE,
                     FOREIGN KEY(id) REFERENCES Attore(id)
                     ON UPDATE CASCADE ON DELETE CASCADE,
                     PRIMARY KEY(titolo, id)) ENGINE = InnoDB;


CREATE TABLE Contratto_Azienda (titolo VARCHAR(32) NOT NULL,
                                nome_azienda VARCHAR(32) NOT NULL,
                                valore_contratto INT UNSIGNED NOT NULL,
                                FOREIGN KEY(titolo) REFERENCES Film(titolo)
                                ON UPDATE CASCADE ON DELETE CASCADE,
                                FOREIGN KEY(nome_azienda) REFERENCES
                                Azienda(nome_azienda)
                                ON UPDATE CASCADE ON DELETE CASCADE,
                                PRIMARY KEY(titolo, nome_azienda)) ENGINE=InnoDB;
```

Per le tabelle delle relazioni ho scelto di inserire i *constraint di foreign key*, ho ritenuto fosse una scelta adeguata per una maggiore robustezza tra i dati inseriti nella base di dati, non sarà possibile inserire una tupla nella tabella di una relazione a meno che i valori di quella tupla non siano già presenti tra le tuple delle tabelle delle entità legate da quella relazione. Sempre per le tabelle delle relazioni ho adottato la politica di reazione *On Delete Cascade*, così, se viene cancellato un record nelle tabelle puntate, si evita di mantenere associazioni ormai prive di significato nella tabella della relazione.

Per molti valori numerici interi ho utilizzato l'opzione *Unsigned*, principalmente perchè non avrebbe avuto senso considerare dei numeri negativi per quei particolari campi, inoltre quest'opzione permette di avere un range più ampio di valori utilizzando sempre lo stesso numero di Bytes.

Dove ho potuto ho utilizzato il tipo di dato $CHAR()$ al posto di $VARCHAR()$, così da evitare spreco di Byte (anche se pochi) di memoria, questo perchè il tipo $VARCHAR()$ utilizza un Byte aggiuntivo per memorizzare la lunghezza della stringa (se la lunghezza massima del campo è 255, se è più di 255 utilizza due Byte).

Le istruzioni riportate sono elencate nello stesso ordine nel file *istruzioni-CREATEcinemaDB.txt*.

# 5 Inserimenti

Gli inserimenti di prova completi sono riportati nel file *istruzioniINSERTcinemaDB.txt*.

```
INSERT INTO Genere VALUES ('AZ', 'Azione'), ('ANZ', 'Animazione'),
                          ('CMC', 'Comico'), ('DR', 'Dramma'),
                          ('H', 'Horror'), ('M', 'Musical'),
                          ('T', 'Thriller'), ('W', 'Western');

INSERT INTO Tipo VALUES ('C', 'Cortometraggio'),
                        ('L', 'Lungometraggio');

INSERT INTO Categoria VALUES ('S', '10 o piu riconoscimenti vinti'),
                             ('A', 'Tra 7 e 9 riconoscimenti vinti'),
                             ('B', 'Tra 4 e 6 riconoscimenti vinti'),
                             ('C', 'Tra 1 e 3 riconoscimenti vinti'),
                             ('D', 'Nessun riconoscimento vinto');


INSERT INTO Agenzia VALUES
('Management International Inc', 'Miami', '7200 Corporate Center', '6321558789'),
('American National Council', 'Denver', '1223 SW 4TH Street', '1327894371'),
('Neighborhood Housing Services', 'Orlando', '300 NW 12TH Ave', '7548312981'),
('The Housing Network', 'Boston', '1 Washington Mall, 12TH Floor', '8895567432'),
('Incharge Debt Solutions', 'New York City', '5750 Major BLVD', '3346578981'),
('Operation Hope', 'Oakland', '5025 W Colonial Drive', '4432219677');


INSERT INTO Studio_Cinematografico VALUES
('Universal Pictures', 'Los Angeles', '881 Kimberly Spring', 0.16),
('Paramount Pictures', 'Los Angeles', '569 Alicia Pass', 0.10),
('Warner Bros Inc', 'Burbank', '687 Huff Lane', 0.12),
('Walt Disney Pictures', 'Sacramento', '5672 Cobb Lane', 0.18),
('Columbia Pictures', 'Culver City', '611 Buckley Burg Suite 896', 0.11),
('Lucasfilm', 'San Francisco', '7674 Joseph Flat Apt. 352', 0.02),
('Miramax', 'Santa Monica', '27026 Ray Neck Apt. 313', 0.03);
```

```
INSERT INTO Artista(nome, cognome, data_nascita, agenzia) VALUES
('Sergio', 'Leone', '1929-01-03', 'Operation Hope'),
('Quentin', 'Tarantino', '1963-03-27', 'American National Council'),
('John', 'Krasinski', '1979-10-20', 'Operation Hope'),
('Cristopher', 'Nolan', '1970-07-30', 'Trinity Empowerment Consortium'),
('Yorgos', 'Lanthimos', '1973-05-27', 'Incharge Debt Solutions'),
('Edgar', 'Wright', '1974-04-18', 'Community Legal Services'),
('David', 'Lynch', '1946-01-20', 'Management International Inc'),
('Guy', 'Ritchie', '1968-09-10', 'Triangle Counseling Agency'),
('Nicolas', 'Winding Refn', '1970-09-29', 'American National Council'),
('Dario', 'Argento', '1940-09-07', 'Agora Community Services'),
('Zachary', 'Snyder', '1966-03-01', 'Management International Inc'),
('Joel', 'Coen', '1963-08-23', 'Neighborhood Housing Services'),
('Ethan', 'Coen', '1957-09-21', 'Neighborhood Housing Services');


INSERT INTO Regista VALUES (1, 8), (2, 9), (3, 2), (4, 7), (5, 4),
                           (6, 6), (7, 9), (8, 5), (9, 3), (10, 7),
                           (11, 5), (12, 6), (13, 7), (14, 6), (15, 4);


INSERT INTO Attore VALUES (3, 2001, 'C'), (2, 1992, 'S'), (16, 1952, 'S'),
                          (26, 1921, 'C'), (32, 2004, 'S'), (33, 2003, 'A'),
                          (34, 2010, 'B'), (35, 2002, 'C'), (36, 2002, 'B'),
                          (37, 1996, 'S'), (38, 2005, 'A'), (39, 2000, 'B');

INSERT INTO Film VALUES
('Interstellar', '2014-11-06', 169, 2508300, 'Universal Pictures', 4, 'L', '2D'),
('Pulp Fiction', '1994-10-14', 154, 2950000, 'Miramax', 2, 'L', '2D'),
('Drive', '2011-10-05', 100, 1787000, 'RKO Pictures', 9, 'L', '2D'),
('Sunspring', '2017-08-08', 14, 529000, 'Legendary Pictures', 19, 'C', '2D'),
('Jaws', '2017-12-13', 112, 1560700, 'Columbia Pictures', 21, 'L', '3D'),
('Hot Fuzz', '2007-02-14', 121, 1443290, 'Silvercup Studios', 6, 'L', '2D'),
('True Grit', '2010-03-30', 130, 922440, '20th Century Studios', 12, 'L', '2D'),
('True Grit', '2010-03-30', 130, 922440, '20th Century Studios', 12, 'L', '2D'),
('Rango', '2011-03-02', 28, 646000, 'Hanna-Barbera Inc', 5, 'C', '3D'),
('La La Land', '2016-07-14', 132, 1500600, 'Metro-Goldwin-Mayer', 15, 'L', '2D'),
('The Mule', '2019-01-23', 116, 2893000, 'Warner Bros Inc', 16, 'L', '2D');
```

```
INSERT INTO Azienda VALUES ('Coca Cola', 1902, 25200000000),
                           ('Pepsi', 1919, 14890000000),
                           ('Nike', 1937, 23790000000),
                           ('Adidas', 1936, 20985000000),
                           ('Puma', 1948, 10870000000),
                           ('Oreo', 1929, 3000000000),
                           ('Toyota', 1946, 19400000000),
                           ('Ford', 1928, 17800000000),
                           ('Starbucks', 1968, 800000000),
                           ('Timberland', 1907, 1000000000),
                           ('Lacoste', 1952, 13000000000);

INSERT INTO Cinema VALUES
('Dream Cinema', 'Vegaburgh', '334 Stewart Estates', 4, 7.50),
('CineOcean', 'Pittsview', '75056 Bernard Glen', 6, 5.00),
('CineMotion', 'East Paula', '50884 Bird Junctions', 3, 4.50),
('Chicago Film Theater', 'Chicago', '531 Amy Passage', 14, 7.50),
('Boston Film Theater', 'Boston', '287 Glen Fort', 8, 10.00),
('CineWorld', 'Turnerland', '397 Ann Loaf', 6, 6.00),
('Astra Cinema', 'West Davidshire', '2026 Ann Glen', 5, 5.50),
('StarPlex', 'Mariaburgh', '223 Patton Circles', 9, 8.50);


INSERT INTO Distribuzione VALUES ('Lucasfilm', 'Dream Cinema'),
                                 ('20th Century Studios', 'Dream Cinema'),
                                 ('Paramount Pictures', 'Dream Cinema'),
                                 ('Walt Disney Pictures', 'CineOcean'),
                                 ('Hanna-Barbera Inc', 'CineOcean'),
                                 ('Universal Pictures', 'CineMotion'),
                                 ('Miramax', 'CineMotion'),
                                 ('Silvercup Studios', 'CineMotion'),
                                 ('Metro-Goldwin-Mayer', 'CineMotion'),
                                 ('20th Century Studios', 'Chicago Film Theater'),
                                 ('Universal Pictures', 'Chicago Film Theater'),
                                 ('Legendary Pictures', 'Chicago Film Theater');

INSERT INTO Generi_Film VALUES ('Pulp Fiction', 'AZ'),
                               ('Pulp Fiction', 'T'),
                               ('Hot Fuzz', 'P'),
                               ('Hot Fuzz', 'CMC'),
                               ('Drive', 'AZ'),
                               ('Drive', 'CMD'),
                               ('Blade Runner 2049', 'N'),
                               ('Blade Runner 2049', 'AZ'),
                               ('Blade Runner 2049', 'FNZ'),
                               ('La La Land', 'M'),
                               ('La La Land', 'CMD');
```

```
INSERT INTO Recita VALUES ('Pulp Fiction', 53, 68000, 'CoProtagonista'),
                          ('Pulp Fiction', 52, 39000, 'CoProtagonista'),
                          ('Pulp Fiction', 2, 19000, 'Secondario'),
                          ('Pulp Fiction', 55, 59000, 'CoProtagonista'),
                          ('Hot Fuzz', 35, 48000, 'Protagonista'),
                          ('Hot Fuzz', 36, 32000, 'CoProtagonista'),
                          ('Drive', 32, 100000, 'Protagonista'),
                          ('Drive', 37, 34000, 'Secondario'),
                          ('Blade Runner 2049', 32, 250000, 'Protagonista'),
                          ('Blade Runner 2049', 54, 175000, 'Secondario'),
                          ('Blade Runner 2049', 34, 75000, 'Secondario'),
                          ('Blade Runner 2049', 74, 180000, 'Secondario'),
                          ('La La Land', 32, 475000, 'CoProtagonista'),
                          ('La La Land', 33, 475000, 'CoProtagonista');


INSERT INTO Contratto_Azienda VALUES ('GoodFellas', 'Pepsi', 85000),
                                     ('GoodFellas', 'Ford', 67900),
                                     ('Dune', 'Timberland', 110000),
                                     ('Alice in Wonderland', 'Oreo', 24000),
                                     ('One Week', 'Oreo', 3000),
                                     ('The Butterfly Circus', 'Nike', 45000),
                                     ('The Butterfly Circus', 'Oreo', 19000),
                                     ('Beautiful Boy', 'Coca Cola', 268500),
                                     ('Batman vs Superman', 'Adidas', 340000),
                                     ('Blade Runner 2049', 'Starbucks', 52000),
                                     ('Blade Runner 2049', 'Pepsi', 150000),
                                     ('The Departed', 'Nike', 220000),
                                     ('Sound Of Metal', 'Adidas', 130000),
                                     ('Dunkirk', 'Timberland', 99000),
                                     ('Dunkirk', 'Lacoste', 76500);
```

# 6 Interrogazioni

Verrà presentata adesso una sequenza di interrogazioni sulla base di dati. Dove necessario le interrogazioni saranno accompagnate da osservazioni o commenti.

Per quanto riguarda l'output che segue ogni singola interrogazione c'è da fare attenzione al fatto che sono stati utilizzati due set di dati differenti, infatti dove si trova una sottolineatura in corrispondenza di un'interrogazione (ad esempio Interrogazione x) vuol dire che è stato utilizzato un set di dati più ampio, mentre dove non è presente la sottolineatura è stato utilizzato il set di dati di prova (file *istruzioniINSERTcinemaDB.txt*).

Ho generato il set di dati più grande dal sito "filldb.info" che ha permesso di mantenere e rispettare i vincoli di *foreign key* tra le tabelle. Con questo dataset il numero di record varia tra 7000 e 10.000 per ogni tabella[4], non si tratta di un numero di record sensazionale ma permette comunque di vedere dei tempi di risposta leggermente più interessanti.

Dato che le query eseguite sul set di dati generato randomicamente potrebbero ritornare molte righe, ho scelto di riportare nelle immagini solo le ultime righe restituite.

Di seguito qualche query per visualizzare il numero di record in alcune delle tabelle riempite con i dati generati.

```
mysql> select count(*) from recita;
+----------+
| count(*) |
+----------+
|    10000 |
+----------+
1 row in set (0.90 sec)
```

```
mysql> select count(*) from studio_cinematografico;
+----------+
| count(*) |
+----------+
|     6933 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> select count(*) from attore;
+----------+
| count(*) |
+----------+
|    10000 |
+----------+
1 row in set (0.19 sec)
```

[4]Non ho generato un numero di record maggiore perchè il tempo per caricare tutte le insert contenute nel file .sql restituito dal sito sarebbe stato eccessivo

**Interrogazione 1:** Selezionare nome, cognome, categoria di tutti gli attori che hanno recitato come protagonisti in film prodotti da studi cinematografici (selezionarne nome e quota di mercato) che hanno una quota di mercato inferiore a 0.08.

$SELECT\ nome,\ cognome,\ categoria.descrizione,\ studio\_cinematografico.nome\_studio,$
$quota\_di\_mercato$
$FROM\ recita,\ artista,\ attore,\ categoria,\ studio\_cinematografico,\ film$
$WHERE\ categoria.codice\_categoria = attore.categoria\ AND\ artista.id = attore.id$
$AND\ attore.id = recita.id\ AND\ recita.ruolo =' Protagonista'$
$AND\ recita.titolo = film.titolo\ AND\ film.studio = studio\_cinematografico.nome\_studio$
$AND\ quota\_di\_mercato < 0.08;$

| | | | | |
|---|---|---|---|---|
| Mara | Brakus | 10 o piu riconoscimenti vinti | architect holistic conte | 0.06 |
| Brionna | Carroll | 10 o piu riconoscimenti vinti | harness dynamic initiati | 0.04 |
| Aylin | O'Conner | 10 o piu riconoscimenti vinti | iterate frictionless net | 0.01 |
| Rebecca | Smith | 10 o piu riconoscimenti vinti | leverage customized nich | 0.02 |
| Emery | Ullrich | 10 o piu riconoscimenti vinti | enable one-to-one web-re | 0.05 |
| Jeremie | Friesen | 10 o piu riconoscimenti vinti | generate interactive inf | 0.07 |
| Adelbert | Doyle | 10 o piu riconoscimenti vinti | harness user-centric exp | 0.02 |

2031 rows in set (0.97 sec)

**Interrogazione 2:** Selezionare tutti gli attori (nome, cognome, ruolo nel film) che hanno recitato in film (selezionarne titolo e formato) in formato "3D".

$SELECT\ nome,\ cognome,\ ruolo,\ film.titolo,\ formato$
$FROM\ artista,\ attore,\ recita,\ film$
$WHERE\ artista.id = attore.id\ AND\ attore.id = recita.id$
$AND\ recita.titolo = film.titolo\ AND\ formato =' 3D';$

| | | | | |
|---|---|---|---|---|
| Albina | Pacocha | Secondario | Visionary user-facing challenge | 3D |
| Corrine | Kuhn | Secondario | Visionary user-facing challenge | 3D |
| Jesse | Reichert | Secondario | Visionary zeroadministration int | 3D |
| Arlene | Green | Comparsa | Visionary zeroadministration int | 3D |
| Stephan | Stoltenberg | Comparsa | Visionary zerodefect artificiali | 3D |
| Saul | Murray | Secondario | Visionary zerodefect artificiali | 3D |

4949 rows in set (0.12 sec)

**Interrogazione 3:** Selezionare tutti i registi che sono anche attori e riportare per ciascuno il numero totale di film girati nella loro carriera e inoltre anche nome e numero di telefono dell'agenzia cui fanno riferimento.

$SELECT\ CONCAT(nome,\ '\ ',\ cognome)\ AS\ regista\_e\_attore,$
$numero\_film\_indipendenti\ +\ COUNT(film.id\_regista)\ AS\ numero\_totale\_film,$
$nome\_agenzia,\ numero\_telefono$
$FROM\ artista,\ regista,\ attore,\ agenzia,\ film$
$WHERE\ artista.id = regista.id\ AND\ artista.id = attore.id$
$AND\ artista.agenzia = agenzia.nome\_agenzia\ AND\ regista.id = film.id\_regista$
$GROUP\ BY\ id\_regista;$

| Cyrus Kulas | 9 | Kuhn Group | (961)961-8 |
|---|---|---|---|
| Madelynn Weissnat | 4 | Kuhn Inc | 0262902032 |
| Jaycee Johnson | 6 | Kuhn LLC | 0119693905 |
| Maida Botsford | 6 | Kuhn Ltd | 1-318-395- |
| Landen Senger | 6 | Kuhn-Berge | (480)937-1 |
| Anita Goodwin | 13 | Kuhn-Blanda | 910.124.19 |
| Elizabeth Braun | 8 | Kuhn-Kiehn | 0431705682 |
| Vergie Lynch | 4 | Kuhn-McCullough | 243-534-25 |

3800 rows in set (0.26 sec)

**Interrogazione 4:** Selezionare tutti gli attori (nome, cognome e data di nascita) che hanno recitato come Protagonisti o personaggi Secondari in almeno un film d'azione o commedie.

$SELECT\ nome,\ cognome,\ data\_nascita,\ film.titolo$
$FROM\ artista,\ attore,\ recita,\ film,\ generi\_film,\ genere$
$WHERE\ artista.id = attore.id\ AND\ attore.id = recita.id\ AND$
$ruolo\ IN\ ('Protagonista',\ 'Secondario')\ AND\ recita.titolo = film.titolo\ AND$
$film.titolo = generi\_film.titolo\ AND\ generi\_film.codice\_genere = genere.codice\_genere$
$AND\ genere.descrizione\ IN\ ('Azione',\ 'Commedia')$
$GROUP\ BY\ attore.id;$

| nome | cognome | data_nascita | titolo |
|---|---|---|---|
| Gal | Gadot | 1985-04-30 | Batman vs Superman |
| Ryan | Gosling | 1980-11-12 | Blade Runner 2049 |
| Ana | De Armas | 1988-04-30 | Blade Runner 2049 |
| Harrison | Ford | 1942-07-13 | Blade Runner 2049 |
| Jared | Leto | 1971-12-26 | Blade Runner 2049 |
| Jamie | Foxx | 1967-12-13 | Django Unchained |
| Christoph | Waltz | 1956-10-04 | Django Unchained |
| Bryan | Cranston | 1956-03-07 | Drive |
| Joaquin | Phoenix | 1974-10-28 | Joker |
| Quentin | Tarantino | 1963-03-27 | Pulp Fiction |
| Alfredo James | Pacino | 1940-04-25 | Scarface |
| John David | Washington | 1984-07-28 | Tenet |
| Christian | Bale | 1974-01-30 | American Psycho |
| Anne | Hathaway | 1982-11-12 | The Butterfly Circus |
| Bradley | Cooper | 1975-01-05 | War Dogs |

15 rows in set (0.01 sec)

*Oss:* Nell'interrogazione numero 4 ho utilizzato il GROUP BY dato che un film può avere doppio genere Commedia + Azione e siamo principalmente interessati ai dati degli attori. Se avessimo voluto anche vedere tutte le recitazioni degli attori selezionati bastava aggiungere l'attributo "titolo" nel GROUP BY.

**Interrogazione 5:** Selezionare titoli, durata, incassi e tipologia dei film diretti da registi che fanno riferimento ad agenzie nel cui nome contengono la parola "Council".

$SELECT\ film.titolo,\ durata,\ incasso,\ tipo.descrizione,\ tipo.formato,\ agenzia$
$FROM\ film,\ tipo,\ regista,\ artista$
$WHERE\ film.id\_regista = regista.id\ AND\ regista.id = artista.id$
$AND\ artista.agenzia\ LIKE\ '\%Council\%'\ AND\ film.tipo = tipo.codice\_tipo;$

| titolo | durata | incasso | descrizione | formato | agenzia |
|---|---|---|---|---|---|
| Django Unchained | 165 | 3400000 | Lungometraggio | 2D | American National Council |
| Pulp Fiction | 154 | 2950000 | Lungometraggio | 2D | American National Council |
| Drive | 100 | 1787000 | Lungometraggio | 2D | American National Council |
| The Mule | 116 | 2893000 | Lungometraggio | 2D | American National Council |
| La La Land | 132 | 1500600 | Lungometraggio | 2D | Eden Council |
| One Week | 24 | 8000 | Cortometraggio | 2D | Eden Council |

6 rows in set (0.00 sec)

**Interrogazione 6:** Per ogni studio cinematografico selezionare la propria percentuale di mercato e il totale degli incassi dei film da loro prodotti.

$SELECT\ CONCAT('\$',\ SUM(incasso)/1000000,\ '\ milioni')\ AS\ incasso\_totale,$
$studio\_cinematografico.nome\_studio,$
$CONCAT(CAST(quota\_di\_mercato * 100\ AS\ UNSIGNED),\ '',\ '\%')\ AS$
$percentuale\_di\_mercato$
$FROM\ film,\ studio\_cinematografico$
$WHERE\ film.studio = studio\_cinematografico.nome\_studio$
$GROUP\ BY\ studio\_cinematografico.nome\_studio;$

| | | |
|---|---|---|
| $1.9678 milioni | architect strategic infr | 8% |
| $2.2575 milioni | aggregate magnetic funct | 3% |
| $3.6123 milioni | engage cutting-edge mode | 5% |
| $8.9554 milioni | revolutionize vertical e | 6% |
| $9.0738 milioni | redefine B2C technologie | 4% |

4233 rows in set (0.05 sec)

**Interrogazione 7:** Selezionare i titoli dei film di genere "horror" e i rispettivi registi che hanno pero' girato nella loro carriera un numero di film indipendenti che sia strettamente minore di 4.

$SELECT film.titolo,\ genere.descrizione,\ nome,\ cognome,\ numero\_film\_indipendenti$
$FROM\ film,\ genere,\ generi\_film,\ regista,\ artista$
$WHERE\ film.titolo = generi\_film.titolo$
$AND\ genere.codice\_genere = generi\_film.codice\_genere$
$AND\ genere.descrizione =' Horror'\ AND\ film.id\_regista = regista.id$
$AND\ artista.id = regista.id\ AND\ numero\_film\_indipendenti < 4;$

| titolo | descrizione | nome | cognome | numero_film_indipendenti |
|---|---|---|---|---|
| A Quiet Place | Horror | John | Krasinski | 2 |
| Jaws | Horror | Ari | Aster | 3 |
| Midsommar | Horror | Ari | Aster | 3 |

3 rows in set (0.02 sec)

**Interrogazione 8:** Selezionare il nome e l'anno di fondazione delle aziende con cui lo studio cinematografico "Miramax" ha firmato contratti.

$SELECT\ azienda.nome\_azienda,\ anno\_fondazione,\ nome\_studio$
$FROM\ azienda,\ film,\ studio\_cinematografico,\ contratto\_azienda$
$WHERE\ studio\_cinematografico.nome\_studio =' Miramax'$
$AND\ studio\_cinematografico.nome\_studio = film.studio$
$AND\ film.titolo = contratto\_azienda.titolo$
$AND\ contratto\_azienda.nome\_azienda = azienda.nome\_azienda$
$GROUP\ BY\ azienda.nome\_azienda;$

*Oss:* il $GROUP\ BY$ è necessario poichè uno studio può aver effettuato accordi con una stessa azienda per film diversi.

| nome_azienda | anno_fondazione | nome_studio |
|---|---|---|
| Pepsi | 1919 | Miramax |
| Starbucks | 1968 | Miramax |
| Timberland | 1907 | Miramax |
| Ford | 1928 | Miramax |
| Coca Cola | 1902 | Miramax |

5 rows in set (0.19 sec)

**Interrogazione 9:** Selezionare la top 3 dei titoli dei film di genere "thriller" con maggiori incassi tra tutti i film che condividono stesso genere, usciti nelle sale prima dell'uscita del film 'Drive'.

$SELECT\ f1.titolo,\ f1.data\_uscita,\ f1.incasso,\ genere.codice\_genere,\ descrizione$
$FROM\ film\ AS\ f1,\ film\ AS\ f2,\ generi\_film,\ genere$
$WHERE\ f2.titolo =' Drive'\ AND\ f1.data\_uscita\ < f2.data\_uscita$
$AND\ f1.titolo = generi\_film.titolo$
$AND\ generi\_film.codice\_genere = genere.codice\_genere$
$AND\ descrizione =' Thriller'$
$ORDER\ BY\ f1.incasso\ DESC\ LIMIT\ 3;$

| titolo | data_uscita | incasso | codice_genere | descrizione |
|---|---|---|---|---|
| Pulp Fiction | 1994-10-14 | 2950000 | T | Thriller |
| Full Metal Jacket | 1978-02-03 | 2210000 | T | Thriller |
| No Country For Old Men | 2007-11-08 | 1805000 | T | Thriller |

3 rows in set (0.06 sec)

**Interrogazione 10:** Per ogni studio cinematografico selezionare il contratto piu' proficuo stabilito con le aziende, visualizzare i risultati in ordine crescente di valore dei contratti.

$SELECT\ studio,\ massimo,\ contratto\_azienda.nome\_azienda$
$FROM\ film,\ contratto\_azienda,\ (SELECT\ studio\ AS\ stud,\ MAX(valore\_contratto)$
$AS\ massimo\ FROM\ contratto\_azienda,\ film\ WHERE$
$film.titolo = contratto\_azienda.titolo\ GROUP\ BY\ stud)\ AS\ tabella$
$WHERE\ contratto\_azienda.valore\_contratto = tabella.massimo$
$AND\ film.titolo = contratto\_azienda.titolo\ AND\ stud = studio$
$ORDER\ BY\ massimo\ ASC;$

| | | |
|---|---|---|
| envisioneer back-end e-c | 85959 | Nikolaus-Stiedemann |
| engage impactful converg | 85965 | Wisoky-Deckow |
| mesh robust content | 85967 | Daugherty, Hauck and Pollich |
| evolve ubiquitous conten | 85974 | Leannon-Tillman |
| morph interactive e-mark | 85978 | Kreiger, Mayer and Monahan |
| brand ubiquitous applica | 85983 | Ledner-Hilll |
| generate end-to-end port | 85984 | Fisher, Hahn and Hamill |

4233 rows in set (0.55 sec)

**Interrogazione 11:** Selezionare gli attori (nome, cognome, categoria) la cui somma dei guadagni sia superiore a $35000 e che rientrano in una delle seguenti categorie: S, A, B.

$SELECT\ nome,\ cognome,\ SUM(compenso)\ AS\ compenso\_totale,$
$categoria.codice\_categoria,\ descrizione$
$FROM\ artista,\ attore,\ recita,\ categoria$
$WHERE\ artista.id = attore.id\ AND\ attore.id = recita.id$
$AND\ attore.categoria = categoria.codice\_categoria$
$AND\ codice\_categoria\ IN\ ('S','A','B')$
$GROUP\ BY\ attore.id\ HAVING\ compenso\_totale > 35000;$

| Jamey | O'Keefe | 182301 | S | 10 o piu riconoscimenti vinti |
|---|---|---|---|---|
| Pearline | Lockman | 737967 | A | Tra 7 e 9 riconoscimenti vinti |
| Kenyon | O'Conner | 319936 | A | Tra 7 e 9 riconoscimenti vinti |
| Oma | Veum | 248336 | B | Tra 4 e 6 riconoscimenti vinti |
| Austen | Reynolds | 459062 | S | 10 o piu riconoscimenti vinti |
| Tillman | Collier | 75000 | B | Tra 4 e 6 riconoscimenti vinti |
| Lamont | Carroll | 72903 | A | Tra 7 e 9 riconoscimenti vinti |
| Reymundo | Hickle | 216492 | A | Tra 7 e 9 riconoscimenti vinti |
| Treva | Goldner | 645148 | S | 10 o piu riconoscimenti vinti |

5998 rows in set (0.45 sec)

**Interrogazione 12:** Selezionare nome, citta' ed indirizzo di tutti i cinema che ricevono pellicole da almeno uno studio cinematografico con quota di mercato superiore o uguale a 0.11.

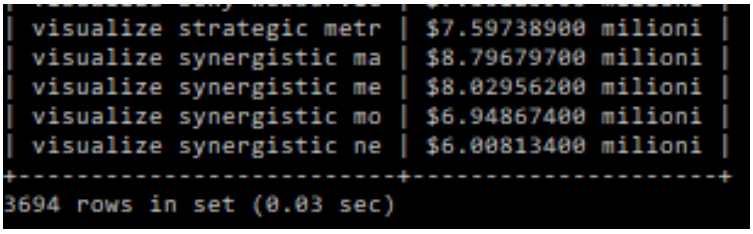
$SELECT\ cinema.nome\_cinema,\ cinema.citta,\ cinema.indirizzo$
$FROM\ cinema,\ distribuzione,\ studio\_cinematografico$
$WHERE\ quota\_di\_mercato >= 0.11$
$AND\ distribuzione.nome\_cinema = cinema.nome\_cinema$
$AND\ studio\_cinematografico.nome\_studio = distribuzione.nome\_studio$
$GROUP\ BY\ cinema.nome\_cinema;$

| nome_cinema | citta | indirizzo |
|---|---|---|
| Boston Film Theater | Boston | 287 Glen Fort |
| Capitol Theater | Armstrongton | 56621 Benjamin Cross |
| Chicago Film Theater | Chicago | 531 Amy Passage |
| Los Angeles Theater | Los Angeles | 372 Hall Skyway |
| Royal Theater | Kelseychester | 8212 Brenda Meadow |
| StarPlex | Mariaburgh | 223 Patton Circles |
| CineMotion | East Paula | 50884 Bird Junctions |
| CineWorld | Turnerland | 397 Ann Loaf |
| Opera Palace | Port Ianhaven | 6775 Douglas Freeway |
| Vista Theater | West Chasetown | 629 Darren Turnpike |
| ArtCraft Theater | Stokesmouth | 350 Scott Fort |
| Astor Theater | Collinschester | 180 Woods Island |
| CineOcean | Pittsview | 75056 Bernard Glen |
| Astra Cinema | West Davidshire | 2026 Ann Glen |

14 rows in set (0.23 sec)

**Interrogazione 13:** Selezionare gli studi cinematografici che hanno una media degli incassi provenienti dai film prodotti superiore a 1.5 milioni di dollari.

$SELECT\ studio,\ CONCAT('\$',\ AVG(incasso)/1000000,\ '\ milioni')$
$AS\ incasso\_medio$
$FROM\ film$
$GROUP\ BY\ studio\ HAVING\ AVG(incasso) > 1500000;$



```
| visualize strategic metr | $7.59738900 milioni |
| visualize synergistic ma | $8.79679700 milioni |
| visualize synergistic me | $8.02956200 milioni |
| visualize synergistic mo | $6.94867400 milioni |
| visualize synergistic ne | $6.00813400 milioni |
+--------------------------+---------------------+
3694 rows in set (0.03 sec)
```

**Interrogazione 14:** Selezionare tutti gli attori che hanno recitato insieme all'attore "Ryan Gosling" e i nomi dei rispettivi film in cui hanno recitato insieme.

$SELECT\ a2.nome,\ a2.cognome,\ r1.titolo$
$FROM\ artista\ AS\ a1,\ artista\ AS\ a2,\ recita\ AS\ r1,\ recita\ AS\ r2$
$WHERE\ a1.nome =' Ryan'\ AND\ a1.cognome =' Gosling'$
$AND\ a2.nome <>' Ryan'\ AND\ a2.cognome <>' Gosling'$
$AND\ r1.id = a1.id\ AND\ a2.id = r2.id\ AND\ r2.titolo = r1.titolo;$

```
+-----------+----------+--------------------+
| nome      | cognome  | titolo             |
+-----------+----------+--------------------+
| Ana       | De Armas | Blade Runner 2049  |
| Harrison  | Ford     | Blade Runner 2049  |
| Jared     | Leto     | Blade Runner 2049  |
| Bryan     | Cranston | Drive              |
| Emma      | Stone    | La La Land         |
| Christian | Bale     | The Big Short      |
| Bradley   | Pitt     | The Big Short      |
| Margot    | Robbie   | The Big Short      |
| Steve     | Carell   | The Big Short      |
+-----------+----------+--------------------+
9 rows in set (0.09 sec)
```

**Interrogazione 15:** Per ogni studio cinematografico selezionare il titolo del film prodotto con durata minore.

$SELECT\ titolo,\ studio,\ durata$
$FROM\ film,\ (SELECT\ studio\ AS\ nome,\ MIN(durata)\ AS\ durata\_minima$
$FROM\ film\ GROUP\ BY\ nome)\ AS\ durate$
$WHERE\ durate.durata\_minima = film.durata\ AND\ durate.nome = film.studio$
$GROUP\ BY\ studio;$

*Oss:* in questa interrogazione ho utilizzato una *derived table* dato che una semplice query come la seguente avrebbe prodotto risultati non corretti:

$SELECT\ titolo,\ studio,\ MIN(durata)\ AS\ durata\_minima$
$FROM\ film\ GROUP\ BY\ studio;$

infatti le colonne *studio*+*durata* avrebbero rappresentato informazioni corrette ma non essendoci particolari condizioni di selezione sul titolo sarebbe stato selezionato un titolo non coerente, questo perche' l'attributo titolo è "fuori" dalle *aggregate functions* $MIN()$ e $GROUP\ BY$.

```
| Reactive systemic workforce      | visualize strategic metr |     80 |
| Self-enabling full-range encodin | visualize synergistic ma |     45 |
| Right-sized assymetric emulation | visualize synergistic me |    100 |
| Multi-tiered exuding pricingstru | visualize synergistic mo |     38 |
| Secured maximized encoding       | visualize synergistic ne |     70 |
+----------------------------------+--------------------------+--------+
4233 rows in set (0.09 sec)
```

**Interrogazione 16:** Selezionare gli incassi prodotti da cortometraggi e contarne anche il numero.

$SELECT\ descrizione,\ COUNT(*)\ AS\ numero,\ SUM(incasso)\ AS\ incasso\_totale$
$FROM\ film,\ tipo$
$WHERE\ film.tipo = tipo.codice\_tipo\ AND\ tipo.descrizione =' Cortometraggio';$

```
+---------------+--------+----------------+
| descrizione   | numero | incasso_totale |
+---------------+--------+----------------+
| Cortometraggio |   2408 |    12045230261 |
+---------------+--------+----------------+
1 row in set (0.01 sec)
```

**Interrogazione 17:** Selezionare tutti i registi (nome, cognome, data di nascita) che fanno riferimento alla stessa agenzia (selezionarne nome e citta) cui fa riferimento "Edgar Wright".

$SELECT\ a1.nome,\ a1.cognome,\ a1.data\_nascita,\ nome\_agenzia,\ citta$
$FROM\ artista\ AS\ a1,\ artista\ AS\ a2,\ regista\ AS\ r1,\ agenzia$
$WHERE\ a2.nome = 'Edgar'\ AND\ a2.cognome = 'Wright'$
$AND\ r1.id = a1.id\ AND\ a1.agenzia = a2.agenzia\ AND$
$a1.agenzia = agenzia.nome\_agenzia;$

| nome | cognome | data_nascita | nome_agenzia | citta |
|---|---|---|---|---|
| Edgar | Wright | 1974-04-18 | Community Legal Services | Seattle |
| George | Lucas | 1944-05-14 | Community Legal Services | Seattle |
| Stanley | Kubrick | 1928-07-26 | Community Legal Services | Seattle |

3 rows in set (0.07 sec)

**Interrogazione 18:** Per ogni ruolo selezionare l'attore (nome e cognome) che ha guadagnato di più per quel ruolo, visualizzare i risultati in ordine crescente di guadagni.

$SELECT\ nome,\ cognome,\ ruolo,\ CONCAT('\$',\ '',\ compenso)\ AS\ compenso$
$FROM\ artista,\ attore,\ recita,\ (SELECT\ ruolo\ AS\ ruolo2,\ MAX(compenso)$
$AS\ massimo\_compenso\ FROM\ recita\ GROUP\ BY\ ruolo)\ AS\ compensi\_massimi$
$WHERE\ recita.id = attore.id\ AND\ attore.id = artista.id$
$AND\ recita.compenso = compensi\_massimi.massimo\_compenso$
$AND\ recita.ruolo = compensi\_massimi.ruolo2$
$GROUP\ BY\ ruolo\ ORDER\ BY\ compenso\ ASC;$

| nome | cognome | ruolo | compenso |
|---|---|---|---|
| Lempi | Wehner | CoProtagonista | $748756 |
| Sanford | Pollich | Secondario | $749618 |
| Sylvia | Jaskolski | Protagonista | $749941 |
| Astrid | Paucek | Comparsa | $749993 |

4 rows in set (0.14 sec)

**Interrogazione 19:** Selezionare per ogni diversa categoria di attori il numero di attori che rientrano in quella categoria in ordine crescente.

$SELECT\ categoria.codice\_categoria,\ descrizione,$
$COUNT(*)\ AS\ numero\_attori\_nella\_categoria$
$FROM\ attore,\ categoria$
$WHERE\ attore.categoria = categoria.codice\_categoria$
$GROUP\ BY\ categoria.codice\_categoria\ ORDER\ BY\ numero\_attori\_nella\_categoria\ ASC;$

| codice_categoria | descrizione | numero_attori_nella_categoria |
|---|---|---|
| D | Nessun riconoscimento vinto | 1977 |
| C | Tra 1 e 3 riconoscimenti vinti | 1980 |
| S | 10 o piu riconoscimenti vinti | 2002 |
| B | Tra 4 e 6 riconoscimenti vinti | 2015 |
| A | Tra 7 e 9 riconoscimenti vinti | 2026 |

5 rows in set (0.02 sec)

**Interrogazione 20:** Selezionare tutti i titoli dei film girati dallo stesso regista (visualizzarne nome e cognome) che ha girato il film "Dunkirk".

$SELECT\ f1.titolo,\ CONCAT(nome,\ '\ ',\ cognome)\ AS\ regista$
$FROM\ film\ AS\ f1,\ film\ AS\ f2,\ artista,\ regista$
$WHERE\ f1.id\_regista = f2.id\_regista\ AND\ f2.titolo =' Dunkirk'$
$AND\ f1.id\_regista = regista.id\ AND\ regista.id = artista.id;$

| titolo | regista |
|---|---|
| Dunkirk | Cristopher Nolan |
| Interstellar | Cristopher Nolan |
| Tenet | Cristopher Nolan |

3 rows in set (0.00 sec)

**Interrogazione 21:** Selezionare per ogni studio cinematografico quale e' il cinema con piu' sale cui distribuiscono pellicole.

$SELECT\ distribuzione.nome\_studio,\ cinema.nome\_cinema,\ cinema.citta,$
$cinema.indirizzo,\ cinema.numero\_sale$
$FROM\ cinema,\ distribuzione,\ (SELECT\ nome\_studio,\ MAX(numero\_sale)$
$AS\ massimo\_numero\_sale\ FROM\ cinema,\ distribuzione\ WHERE$
$distribuzione.nome\_cinema = cinema.nome\_cinema\ GROUP\ BY\ nome\_studio)$
$AS\ numeri\_sale$
$WHERE\ cinema.nome\_cinema = distribuzione.nome\_cinema$
$AND\ cinema.numero\_sale = massimo\_numero\_sale$
$AND\ numeri\_sale.nome\_studio = distribuzione.nome\_studio;$

| | | | | |
|---|---|---|---|---|
| whiteboard virtual deliv | Reilly Group | Port Cecil | 338 Muriel Fork | 5 |
| whiteboard virtual exper | Reilly Ltd | Kemmerburgh | 835 Medhurst Flats Apt. 722 | 8 |
| whiteboard virtual netwo | Reilly PLC | McKenzieside | 7313 Weissnat Course Suite 722 | 11 |
| whiteboard visionary exp | Reilly-Bruen | Sauertown | 6237 Denesik Flats | 14 |
| whiteboard web-enabled i | Reilly-Graham | North Leon | 854 Spinka Inlet Apt. 763 | 13 |
| whiteboard web-enabled m | Reilly-Haley | Alethaside | 32617 Rubye Turnpike | 9 |
| whiteboard web-enabled t | Reilly-Kub | East Katelynn | 2702 Leatha Knoll | 6 |
| whiteboard wireless acti | Reilly-Ruecker | South Lillian | 69694 Zemlak Parkway Suite 692 | 7 |
| whiteboard wireless netw | Reilly-Williamson | Deckowstad | 83617 Ryan Junction Suite 401 | 5 |
| whiteboard wireless vort | Reilly, Borer and Friese | Evashire | 41095 Ullrich Junction Suite 597 | 8 |
| whiteboard world-class s | Reilly, Cummings and Con | Jonesmouth | 46258 Raynor Way Suite 089 | 4 |

7149 rows in set (0.89 sec)

**Interrogazione 22:** Selezionare la percentuale di film prodotti dallo studio cinematografico "Paramount Pictures".

$SELECT\ CONCAT((COUNT(film.titolo)/totale) * 100,\ '',\ '\%')\ AS\ percentuale,$
$COUNT(film.titolo)\ AS\ numero,\ studio$
$FROM\ film,\ (SELECT\ COUNT(*)\ AS\ totale\ FROM\ film)\ AS\ numero\_totale$
$WHERE\ studio = '\ Paramount\ Pictures';$

| percentuale | numero | studio |
|---|---|---|
| 9.0909% | 4 | Paramount Pictures |

1 row in set (0.01 sec)

**Interrogazione 23:** Selezionare in percentuale per ogni studio cinematografico quanti film hanno prodotto sul totale.

$SELECT\ CONCAT((parziale/totale) * 100,\ '',\ '\%')\ AS\ percentuale,$
$parziale\ AS\ numero,\ studio$
$FROM\ film,\ (SELECT\ studio\ AS\ nome,\ COUNT(*)\ AS\ parziale\ FROM\ film$
$GROUP\ BY\ studio)\ AS\ tmp\_count,\ (SELECT\ COUNT(*)\ AS\ totale\ FROM\ film)$
$AS\ tot\_count$
$WHERE\ studio = nome$
$GROUP\ BY\ studio;$

```
| 0.0208%     |      1 | visualize strategic metr |
| 0.0208%     |      1 | visualize synergistic ma |
| 0.0208%     |      1 | visualize synergistic me |
| 0.0208%     |      1 | visualize synergistic mo |
| 0.0208%     |      1 | visualize synergistic ne |
+-------------+--------+--------------------------+
4233 rows in set (0.06 sec)
```

**Interrogazione 24:** Selezionare il guadagno totale dato dalla produzione del film "Pulp Fiction", tenendo quindi conto degli incassi del film, dei contratti stretti con le aziende e dei soldi necessari a pagare gli attori.

$SELECT\ CONCAT('\$',\ (SUM(valore\_contratto) + incasso - totale\_dovuto)/1000000,$
$'\ milioni')\ AS\ guadagno\_totale, film.titolo$
$FROM\ film,\ contratto\_azienda,\ (SELECT\ SUM(compenso)\ AS\ totale\_dovuto$
$FROM\ recita\ WHERE\ titolo =' Pulp\ Fiction')\ AS\ tot$
$WHERE\ film.titolo =' Pulp\ Fiction'\ AND\ film.titolo = contratto\_azienda.titolo;$

```
+--------------+-----------------+
| titolo       | guadagno_totale |
+--------------+-----------------+
| Pulp Fiction | $3.0054 milioni |
+--------------+-----------------+
1 row in set (0.00 sec)
```

**Interrogazione 25:** Selezionare gli attori (nome, cognome, anno di inizio carriera) e le agenzie (nome agenzia, citta, numero di telefono) cui fanno riferimento ma solo per attori che hanno iniziato la loro carriera nel seguente intervallo chiuso $[esordio\_Bruce\_Willis - 10,\ esordio\_Bruce\_Willis + 10]$.

$SELECT\ a1.nome,\ a1.cognome,\ attore1.esordio,$
$agenzia.nome\_agenzia,\ agenzia.citta,\ agenzia.numero\_telefono$
$FROM\ agenzia,\ artista\ AS\ a1,\ artista\ AS\ a2,\ attore\ AS\ attore1,\ attore\ AS\ attore2$
$WHERE\ a1.id = attore1.id\ AND\ a2.nome =' Bruce'\ AND\ a2.cognome =' Willis'$
$AND\ a2.id = attore2.id\ AND\ attore1.esordio\ BETWEEN\ (attore2.esordio - 10)$
$AND\ (attore2.esordio + 10)\ AND\ a1.agenzia = agenzia.nome\_agenzia;$

| nome | cognome | esordio | nome_agenzia | citta | numero_telefono |
|---|---|---|---|---|---|
| Stefania | Casini | 1969 | Incharge Debt Solutions | New York City | 3346578981 |
| Denzel | Washington | 1980 | Agora Community Services | Chicago | 7989946473 |
| Bruce | Willis | 1973 | Smart Money Housing | Boston | 2431127278 |
| Harrison | Ford | 1982 | Community Legal Services | Seattle | 2445456672 |
| Robert | De Niro | 1971 | Agora Community Services | Chicago | 7989946473 |
| Joseph | Pesci | 1972 | Trinity Empowerment Consortium | San Francisco | 2234531198 |
| Morgan | Freeman | 1978 | Incharge Debt Solutions | New York City | 3346578981 |
| Jeffrey | Bridges | 1982 | Management International Inc | Miami | 6321558789 |

8 rows in set (0.04 sec)

**Interrogazione 26:** Selezionare tutti i titoli, date di uscita, durate e incassi dei film di genere "biografico" che hanno incassato piu' del film "Full Metal Jacket", visualizzare i risultati in ordine decrescente di incasso.

$SELECT\ f1.titolo,\ descrizione,\ f1.data\_uscita,\ f1.incasso,$
$CONCAT(f1.durata,\ '',\ '\ minuti')\ AS\ durata$
$FROM\ film\ AS\ f1,\ film\ AS\ f2,\ generi\_film,\ genere$
$WHERE\ generi\_film.codice\_genere = genere.codice\_genere$
$AND\ descrizione =' Biografico'\ AND\ f1.titolo = generi\_film.titolo$
$AND\ f2.titolo =' Full\ Metal\ Jacket'\ AND\ f1.incasso > f2.incasso$
$ORDER\ BY\ f1.incasso\ DESC;$

| titolo | descrizione | data_uscita | incasso | durata |
|---|---|---|---|---|
| Scarface | Biografico | 1983-12-15 | 2900000 | 170 minuti |
| The Mule | Biografico | 2019-01-23 | 2893000 | 116 minuti |
| GoodFellas | Biografico | 1990-12-25 | 2800000 | 140 minuti |
| Joker | Biografico | 2019-08-31 | 2400000 | 122 minuti |

4 rows in set (0.00 sec)

**Interrogazione 27:** Selezionare le agenzie (tutti gli attributi) cui fanno riferimento solo ed esclusivamente artisti che sono registi e cui non fa riferimento alcun attore.

$SELECT\ nome\_agenzia,\ citta,\ indirizzo,\ numero\_telefono$
$FROM\ regista,\ artista,\ agenzia$
$WHERE\ artista.agenzia = agenzia.nome\_agenzia\ AND\ regista.id = artista.id$
$AND\ artista.agenzia\ NOT\ IN\ (SELECT\ agenzia\ FROM\ artista,\ attore$
$WHERE\ artista.id = attore.id\ GROUP\ BY\ agenzia)$
$GROUP\ BY\ agenzia;$

*Oss:* questa soluzione con la *subquery* nella clausola *where* è l'unica soluzione che sono riuscito a trovare all'interrogazione. Ho provato diversi approcci con derived tables, alias e tentato anche alcuni "trucchetti" con il *left join* (ansi style) per realizzare l'operatore insiemistico di differenza ma purtroppo non sono riuscito a finalizzare nessuno di questi approcci (tutti sicuramente più eleganti della *subquery* nella clausola *where*).

```
+---------------------------+---------+--------------------+------------------+
| nome_agenzia              | citta   | indirizzo          | numero_telefono  |
+---------------------------+---------+--------------------+------------------+
| Triangle Counseling Agency | Burbank | 2815 Wendy Summit | 7789546632       |
+---------------------------+---------+--------------------+------------------+
1 row in set (0.05 sec)
```

**Interrogazione 28:** Selezionare l'azienda che ha net worth piu' grande e il film (titolo, data di uscita, incasso, studio di produzione) sponsorizzato che ha incassato di meno rispetto tutti gli altri film sponsorizzati dalla stessa azienda.

$SELECT\ azienda.nome\_azienda,\ CONCAT('\$',\ CAST(azienda.net\_worth/1000000000$
$AS\ FLOAT(2)),\ '\ miliardi')\ AS\ net\_worth,\ film.titolo,\ data\_uscita,\ incasso,\ studio$
$FROM\ (SELECT\ MAX(net\_worth)\ AS\ net\_massimo\ FROM\ azienda)\ AS$
$azienda\_massimo,\ (SELECT\ nome\_azienda\ AS\ nome,\ MIN(incasso)\ AS$
$incassi\_minimi\ FROM\ film,\ contratto\_azienda\ WHERE$
$film.titolo = contratto\_azienda.titolo\ GROUP\ BY\ nome\_azienda)\ AS$
$contratti\_minimi,\ azienda,\ film$
$WHERE\ azienda.net\_worth = azienda\_massimo.net\_massimo$
$AND\ azienda.nome\_azienda = contratti\_minimi.nome\ AND\ incasso = incassi\_minimi;$

```
+---------------+----------------------+--------------------------------+------------+---------+-------------------------+
| nome_azienda  | net_worth            | titolo                         | data_uscita | incasso | studio                  |
+---------------+----------------------+--------------------------------+------------+---------+-------------------------+
| Kulas and Sons | $31.995578144 miliardi | Advanced zerotolerance software | 1990-04-21  | 7824461 | incentivize dot-com mind |
+---------------+----------------------+--------------------------------+------------+---------+-------------------------+
1 row in set (0.42 sec)
```

# 7 Interrogazioni in Algebra Relazionale

Di seguito la traduzione di due interrogazioni in *Algebra relazionale*, rispettivamente l'interrogazione numero 2 e l'interrogazione numero 20.[5]

**Interrogazione numero 2:** Selezionare tutti gli attori (nome, cognome, ruolo nel film) che hanno recitato in film (selezionarne titolo e formato) in formato "3D".

$$\pi_{nome,cognome,ruolo,titolo,formato}(ARTISTA \bowtie_{id=idAttore} (((\rho_{idAttore \leftarrow id}(ATTORE)) \bowtie_{idAttore=idRecita} (\rho_{idRecita,titoloRecita \leftarrow id,titolo}(RECITA))) \bowtie_{titoloRecita=titolo} (\sigma_{formato='3D'}(FILM))))$$

**Interrogazione numero 20:** Selezionare tutti i titoli dei film girati dallo stesso regista (visualizzarne nome e cognome) che ha girato il film "Dunkirk".

$$\pi_{titolo1,nome,cognome}(((\sigma_{titolo2='Dunkirk'}(\rho_{idRegFilm2,titolo2 \leftarrow idRegista,titolo}(FILM))) \bowtie_{idRegFilm2=idRegFilm1} (\rho_{idRegFilm1,titolo1 \leftarrow idRegista,titolo}(FILM))) \bowtie_{idRegFilm1=idRegista} (\rho_{idRegista \leftarrow id}(REGISTA)) \bowtie_{idRegista=id} ARTISTA)$$

[5]Le ridenominazioni sono state effettuate per ragioni di spazio solo su attributi che dovevano essere selezionati dalla *Proiezione* (e non su tutti gli attributi della relazione) oppure semplicemente per avere attributi di nome diverso nell'operazione di *join* sulle relazioni

# 8 Triggers

Ho realizzato i seguenti *Triggers* per mantenere e rispettare alcune regole e relazioni che sussistono tra gli oggetti del database.

## 8.1 Trigger: Controllo di Tipo

L'obbiettivo è quello di realizzare un trigger che aiuti a rispettare la regola aziendale presentata nel paragrafo 1: "Un film è considerato *cortometraggio* se la sua durata è pari o inferiore a 30 minuti, altrimenti è considerato *lungometraggio*".

Il Trigger seguente esegue dei controlli, prima dell'inserimento di un record nella tabella film, al fine di verificare se la combinazione $durata + tipo$ è ammissibile, se non è rispettata la regola aziendale viene modificato il campo tipo in accordo alla durata del film.

```
DELIMITER //

CREATE TRIGGER checkMovieType BEFORE INSERT ON film
FOR EACH ROW

BEGIN

        IF NEW.tipo = 'L' AND NEW.durata <= 30 THEN
                SET NEW.tipo = 'C';
        END IF;

        IF NEW.tipo = 'C' AND new.durata > 30 THEN
                SET NEW.tipo = 'L';
        END IF;

END; //

DELIMITER ;
```

### 8.2 Trigger: Controllo del Numero di Generi

Una relazione che può essere importante mantenere è quella tra un film e i generi che lo caratterizzano. Come specificato nel paragrafo 1 e come mostrato nello schema logico, un film può essere caratterizzato al più da 3 generi diversi.

Quello che si vuole quindi fare è impedire che il titolo di un film occorra più di 3 volte nella tabella *generi_film*, purtroppo MySQL non mette a disposizione strumenti o comandi specifici per impedire l'inserimento di un record al verificarsi di una certa condizione.

Per annullare l'inserimento del record ho sfruttato il segnale *sqlstate 45000*, se ad un film sono già associati esattamente 3 generi, allora viene invocato il segnale e viene mostrato il messaggio.

```
DELIMITER //

CREATE TRIGGER checkNumeroGeneri BEFORE INSERT ON generi_film
FOR EACH ROW

BEGIN
        DECLARE currentNumber TINYINT;

        SELECT COUNT(*) AS numero INTO currentNumber
        FROM generi_film
        WHERE generi_film.titolo = NEW.titolo;

        IF currentNumber = 3 THEN
                SIGNAL SQLSTATE '45000'
                SET message_text = 'Inserimento non consentito,
                                    numero massimo di generi raggiunto';

        END IF;

END; //

DELIMITER ;
```

# 9 Cursori

Utilizzando il linguaggio *Python* e alcune librerie ho sviluppato un paio di semplici programmi che utilizzano cursori.

## 9.1 Interfaccia con Cursore

Ho realizzato una interfaccia a riga di comando sul database che utilizza un cursore per l'esecuzione di query (*DDL*, *DML* e *SQL*).

La libreria di Python che ho utilizzato è "mysql.connector" (installazione da linea di comando: "pip3 install mysql-connector-python").

```
1   import mysql.connector as ms
```

Il primo frammento di codice vero e proprio riguarda la connessione all'istanza locale. Attraverso la funzione "connect(...)" vengono specificati in input i parametri per ottenere la connessione (di default sono impostati ai valori a destra dell'uguale per ogni parametro), dopodichè viene richiesto di inserire la password relativa all'utente e all'hostname di input. Con il metodo "ms.connect(...)" si ottiene effettivamente l' *oggetto connessione* (questo se i parametri indicati sono tutti corretti, altrimenti verranno sollevate delle eccezioni) e con tale *oggetto connessione* si può finalmente richiedere il *cursore* sul database. Con il metodo "execute(...)" del cursore ho richiesto l'esecuzione dell'istruzione "Use <nome_database>", l'ho fatto più per sicurezza che per necessità[6].

```
21  def connect(hostName = 'localhost', userName = 'root', databaseName = 'cinemadb'):
22      passwd = input('Inserisci la password per utente ' + userName + ': ')
23      connessione = ms.connect(host = hostName, user = userName,
24                               database = databaseName, password = passwd)
25      cursore = connessione.cursor()
26
27      cursore.execute('USE ' + databaseName)
28
29      eseguiQuery(cursore)
30
31
32  if __name__ == '__main__':
33      connect()
34
```

[6]Il cursore dovrebbe operare già in automatico sul database specificato nei parametri precedenti

Come ultima istruzione della funzione "connect" viene chiamata la funzione "eseguiQuery" il cui codice è riportato di seguito.

```
3   def eseguiQuery(cursore):
4
5       query = ""
6
7       while True:
8
9           query = input('Inserisci la query che vorresti eseguire: ')
10
11          if query in ("exit", "Exit", "logout", "Logout", "quit", "Quit", "q", "Q"):
12              print('Bye!')
13              break
14
15          cursore.execute(query)
16          result = cursore.fetchall()
17
18          for row in result:
19              print(row, '\n')
```

All'interno del ciclo while viene chiesto all'utente di inserire una query da eseguire sul database, dopo un rapido controllo per verificare se la query è un comando di uscita dalla sessione, viene eseguita la query e con il metodo "fetchall()" si recuperano i risultati, i quali vengono inseriti nella lista chiamata "result". Infine con un ciclo for si scorre la lista dei risultati e si mostrano a schermo le tuple ritornate.

```
Inserisci la query che vorresti eseguire: SELECT titolo, durata FROM Film where durata > 115
('A Quiet Place', 123)

('Alice in Wonderland', 118)

('American Gangster', 157)

('Batman vs Superman', 151)

('Beautiful Boy', 129)

('Blade Runner 2049', 146)
```

## 9.2 Inserimenti con Cursore

Il secondo programma che ho realizzato per l'utilizzo dei cursori riguarda gli inserimenti di record di prova (*dummy data*) all'interno della base di dati, nello specifico il codice che verrà mostrato in questa sezione si occuperà di inserire dei record all'interno della tabella *Azienda*.

```python
import mysql.connector as ms
import numpy as np
from faker import Faker


def generaNomeAzienda():
    faker = Faker()
    nome = faker.company()
    while True:
        if len(nome) <= 32:
            break
        nome = faker.company()
    return nome

def generaIntero(low = 1, high = 15):
    return np.random.randint(low, high)

```

Vengono utilizzate tre librerie in totale: "mysql.connector" per l'ottenimento del cursore, "numpy" per generare randomicamente dei valori numerici e "faker" per generare randomicamente informazioni che riempiranno i campi della tabella. Le prime due funzioni sono utilizzate rispettivamente per: ottenere il nome di un'azienda (la cui lunghezza sia $\leq 32$, dato che il campo "nome_azienda" è dichiarato come VARCHAR(32)) e ottenere un intero che si trova all'interno di un certo range specificato in input.

La seguente funzione "connect()" è la stessa mostrata precedentemente e serve sempre per l'ottenimento del cursore.

```python
def connect(hostName = 'localhost', userName = 'root', databaseName = 'cinemadb'):
    passwd = input('Inserisci la password per utente ' + userName + ': ')
    connessione = ms.connect(host = hostName, user = userName,
                             database = databaseName, password = passwd)
    cursore = connessione.cursor()

    cursore.execute('USE ' + databaseName)

    addData(cursore, connessione)


if __name__ == '__main__':
    connect()

```

Nella funzione che segue avvengono effettivamente gli inserimenti nella tabella *Azienda*. In input, insieme al cursore e alla connessione, è passato anche il numero di record che si vogliono inserire (di default 50). La prima cosa che si fa è selezionare dalla tabella *Azienda* tutti i valori di chiave che sono già in utilizzo con la funzione "fetchCompanyData()", tali dati recuperati vengono poi passati alla funzione "formatData()" che ha il semplice compito di formattare meglio i valori di chiave all'interno di una lista (*inUseKeyData* è una copia della lista ritornata da "formatData()"). All'interno del ciclo for (una iterazione per ogni inserimento) si generano i valori, si controlla che il nome dell'azienda generato non sia già presente tra i valori di chiave usati e si inseriscono in una tupla. Viene eseguita la query dichiarata fuori dal ciclo for con i $\%s$ sostituti dai valori della tupla appena costruita. Al termine di tutti gli inserimenti viene effettuato il *commit* per finalizzare le operazioni.

```
43  def addData(cursore, connessione, numberOfRecords = 50):
44
45      companyData = fetchCompanyData(cursore, connessione)
46
47      formattedCompanyData = formatData(companyData)
48      inUseKeyData = formattedCompanyData[:]
49
50      query = "INSERT INTO azienda VALUES (%s, %s, %s)"
51
52      for i in range(numberOfRecords):
53          nome = generaNomeAzienda()
54          if nome not in inUseKeyData:
55              inUseKeyData.append(nome)
56          else:
57              continue
58          year = generaIntero(1902, 2005)
59          net_worth = generaIntero(320000, 2147483646)
60          data = (nome, year, net_worth)
61
62          cursore.execute(query, data)
63
64      connessione.commit()
```

In generale questo script è utile solo quando i record da inserire sono relativamente pochi, il codice Python, di per sé, è abbastanza lento ad essere eseguito rispetto altri linguaggi di programmazione, inserire un numero consistente di record potrebbe richiedere molto tempo.

Riporto anche il codice delle due funzioni "fetchCompanyData()" e "formatData()".

```
6   def fetchCompanyData(cursore, connessione):
7   
8       companyData = []
9   
10      query = "SELECT nome_azienda from azienda"
11  
12      cursore.execute(query)
13      result = cursore.fetchall()
14  
15      return result
16  
17  
18  def formatData(Data = []):
19  
20      formattedData = []
21  
22      for i in Data:
23          formattedData.append(i[0])
24  
25      return formattedData
```

Questi i risultati dopo l'inserimento di 20 record usando lo *script Python*.

```
mysql> select * from azienda;
+----------------------------------+-----------------+------------+
| nome_azienda                     | anno_fondazione | net_worth  |
+----------------------------------+-----------------+------------+
| Bailey Group                     |            1954 | 1079852800 |
| Bauer, Jones and Hood            |            1938 | 1295894831 |
| Burns, Hunter and Dennis         |            1935 | 1117655007 |
| Cruz, Monroe and Duncan          |            1997 | 1565367905 |
| Davies-Moran                     |            1978 | 1273705963 |
| Davis Group                      |            1953 |  592001098 |
| Fisher, Lopez and Harris         |            1921 |  948427130 |
| Gonzalez, Williams and Armstrong |            1919 |  410535261 |
| House Ltd                        |            1916 | 1390019544 |
| Jenkins, Johnson and Tran        |            1952 | 1342855716 |
| Jones, Parks and Palmer          |            1915 | 1156353810 |
| Miller, Smith and Bond           |            1961 | 1603636950 |
| Morgan, Gibson and Calderon      |            1972 | 1419182095 |
| Nelson LLC                       |            1980 |  697167722 |
| Nelson Ltd                       |            1925 |  743259805 |
| Robinson-Wiley                   |            1933 |  970402367 |
| Smith-Smith                      |            1961 |  206635506 |
| Thompson, Elliott and Perez      |            1951 |  143417029 |
| Vaughn-Soto                      |            1916 | 1639029737 |
| Wade PLC                         |            1998 |  182509512 |
+----------------------------------+-----------------+------------+
20 rows in set (0.00 sec)
```

# 10 Viste

In questa sezione riporto la creazione di qualche vista che può essere utile per avere subito a disposizione informazioni interessanti.

La prima vista é chiamata "totaleProduzione" e raccoglie, per ogni studio cinematografico, il totale degli incassi provenienti dai film da questi prodotti.

```
CREATE VIEW totaleProduzione(studio, totale_incassi)
        AS SELECT studio, SUM(incasso)
           FROM film
           GROUP BY studio;
```

```
mysql> create view totaleProduzione(studio, incassi_totali) as select studio, sum(incasso) from film group by studio;
Query OK, 0 rows affected (0.23 sec)

mysql> select * from totaleProduzione;
+----------------------+----------------+
| studio               | incassi_totali |
+----------------------+----------------+
| 20th Century Studios |        9622440 |
| Columbia Pictures    |       11515700 |
| Hanna-Barbera Inc    |         646000 |
| Legendary Pictures   |        5806000 |
| Lucasfilm            |        2925000 |
| Metro-Goldwin-Mayer  |        1520600 |
| Miramax              |       16029000 |
| Paramount Pictures   |        7404600 |
| RKO Pictures         |        7391000 |
| Silvercup Studios    |        6967910 |
| Universal Pictures   |        4054370 |
| Walt Disney Pictures |        6330000 |
| Warner Bros Inc      |        5301000 |
+----------------------+----------------+
13 rows in set (0.10 sec)
```

Questa vista risolve, ad esempio, quasi completamente l'interrogazione numero 6 riportata nel paragrafo 6.

La seconda e ultima vista[7], chiamata "partecipazioniAttori", mostra invece per ogni attore il numero di partecipazioni totali nei film.

```
CREATE VIEW partecipazioniAttori(nome, cognome, numeroPartecipazioni)
        AS SELECT nome, cognome, COUNT(*)
           FROM artista, attore, recita
           WHERE artista.id = attore.id
           AND attore.id = recita.id
           GROUP BY attore.id;
```

Ho pensato che queste viste potessero essere interessanti dato che, nel campo del cinema e della produzione cinematografica, in molti sono interessati ai numeri e alle statistiche.

[7]Non riporto una schermata di output per questa vista poiché raccoglie 56 righe e servirebbe una pagina intera solo per l'output

# 11 MongoDB

Il ramo del database che ho scelto per l'implementazione in MongoDB è quello che coinvolge i seguenti oggetti: *Azienda*, *Contratto Azienda*, *Film*, *Recita* e *Attore*. Parliamo quindi del ramo che parte dall'entità *Azienda* e risale fino all' entità *Attore*. Ho racchiuso la totalità delle informazioni all'interno di 3 *collections* che prendono il nome delle rispettive entità coinvolte.

## 11.1 Creazione delle Collections e Struttura dei Documents

Per generare tutti i *documents* delle collections *Azienda*, *Film* e *Attore*, ho utilizzato un software chiamato "mgoDataGen" che è liberamente disponibile su *GitHub* (indirizzo: https://github.com/feliixx/mgodatagen). Questo software permette di creare velocemente molti documents, specificandone semplicemente la struttura. Come esempio riporto la struttura dei documents della collection *Film* che "mgoDataGen" ha utilizzato per generare randomicamente i documents.

```
{
        "database": "cinemaDB",
        "collection": "Film",
        "count": 500000,
        "content": {

                "titolo": { "type": "ref", "id": 1, "refContent": {
                "type": "string",
                "unique": true,
                "minLength": 8,
                "maxLength": 24
                }},
                "dataUscita": {"type": "date", "startDate": "1980-01-01T00:00:00+00:00",
                                "endDate": "2020-01-01T22:00:00+00:00"},
                "durata": {"type": "int", "minInt": 25, "maxInt": 145},
                "incasso": {"type": "int", "minInt": 38000, "maxInt": 1450000},
                "_idRegista": {"type": "int", "minInt": 0, "maxInt": 20000},
                "studio": {"type": "faker", "method": "Company"},
                "tipo": {"type": "fromArray", "in": ["L", "C"], "randomOrder": true},
                "formato": {"type": "fromArray", "in": ["2D", "3D"],
                            "randomOrder": true},
                "contratti": {"type": "array", "minLength": 0, "maxLength": 4,
                             "arrayContent":
                              {"type": "object",
                              "objectContent": {
                                        "nomeAzienda": {"type": "ref", "id": 2,
                                        "refContent": {"type": "string", "id": 1,
                                        "minLength": 12, "maxLength":32}},
                                        "valoreContratto": {"type": "int",
                                        "minInt": 23000, "maxInt": 234000}
                                         }
                             }
                            }
                }
},
```

Questi i risultati dopo l'esecuzione di "mgoDataGen" con input la struttura delle collections.

```
C:\Users\ACER\Desktop>mgodatagen -f config.json
connecting to mongodb://127.0.0.1:27017
MongoDB server version 4.4.4

collection Attore: done            [========================================================================] 100%
collection Film: done              [========================================================================] 100%
collection Azienda: done           [========================================================================] 100%

+------------+--------+-----------------+---------------+
| COLLECTION | COUNT  | AVG OBJECT SIZE |    INDEXES    |
+------------+--------+-----------------+---------------+
| Attore     | 500000 |             377 | _id_  4644 kB |
| Film       | 500000 |             319 | _id_  4712 kB |
| Azienda    | 500000 |             100 | _id_  4100 kB |
+------------+--------+-----------------+---------------+

run finished in 24.77s
```

Prima di commentare nello specifico i campi dei documents della collection *Film* vorrei soffermarmi su un paio di parametri che "mgoDataGen" ha permesso di gestire. Prima di tutto il parametro "count" al quale viene associato il numero di documents da generare, in questo caso ho scelto 500.000 documents per ogni collection (1.500.000 documents in totale); inoltre "mgoDataGen" ha permesso di gestire i riferimenti tra oggetti di collezioni diverse, consentendo quindi di effettuare ricerche "incrociate" tra collections. Ad esempio nei documents della collection *Film*, all'interno dell'array chiamato "contratti", il campo "nomeAzienda" andrà effettivamente a coincidere con il campo di uno specifico document nella collection *Azienda*.

```
{
        "_id" : ObjectId("61320524864ee5deb8ef7b35"),
        "contratti" : [
                {
                        "nomeAzienda" : "S61W0aaaaa7z13qaaaaauChmZ",
                        "valoreContratto" : 111485
                },
                {
                        "nomeAzienda" : "gRez0caaaaBLzzybaaaaH67B",
                        "valoreContratto" : 34404
                },
                {
                        "nomeAzienda" : "EUl2-baaaa7kmmMbaaaaopCBfdaaaaO",
                        "valoreContratto" : 193809
                }
        ],
        "studio" : "Maponics",
        "incasso" : 273761,
        "formato" : "3D",
        "tipo" : "C",
        "_idRegista" : 2927,
        "durata" : 120,
        "dataUscita" : ISODate("1984-04-15T17:45:40Z"),
        "titolo" : "Ta0D4baa"
}
```

L'immagine qui sopra riporta la struttura adottata per i documents nella collection *Film*. Il campo "_id" è automaticamente assegnato da MongoDB mentre tutti i restanti campi sono gli stessi che troviamo come attributi nell'entità

*Film* dello schema relazionale, eccezion fatta per il campo "contratti" che conferisce ai documents una struttura denormalizzata. Lo scopo dell'array "contratti" è quello di tenere traccia, per ogni film, dei contratti stretti con le aziende. I valori associati ai campi "nomeAzienda" che occorrono nel vettore fanno "riferimento" ai rispettivi valori nella collection *Azienda*.

Tornando brevemente alla generazione randomica dei documents della collezione *Film*, i campi "incasso", "durata", "_idRegista"[8] e "valoreContratto" (che occorre nell'array di oggetti) sono dichiarati come tipi interi; i campi "studio", "formato", "tipo" e "titolo" sono invece delle stringhe (il campo "titolo" è inoltre dichiarato come *unique*); il campo "dataUscita" è invece un oggetto *date*.

Anche i documents della collection *Attore* hanno una struttura denormalizzata, questo grazie alla presenza dell'array "partecipazioni" che consente di tenere traccia, per ogni attore, dei ruoli recitati nei film.


```
{
        "_id" : ObjectId("6132051c864ee5deb8e7da15"),
        "id" : 18,
        "partecipazioni" : [
                {
                        "titolo" : "_Ne_JcaaaaAJ0CTc",
                        "compenso" : 857566,
                        "ruolo" : "Comparsa"
                },
                {
                        "titolo" : "Y0CNxdaaa",
                        "compenso" : 77175,
                        "ruolo" : "Comparsa"
                },
                {
                        "titolo" : "85IP-baaa",
                        "compenso" : 121156,
                        "ruolo" : "Protagonista"
                },
                {
                        "titolo" : "kQ7eMdaaaaBbh93aaaaazt",
                        "compenso" : 377192,
                        "ruolo" : "Comparsa"
                }
        ],
        "esordio" : 1977,
        "categoria" : "D",
        "cognome" : "Tremblay",
        "nome" : "Odessa"
}
```


*Osservazione*: Dato che l'implementazione in MongoDB non tiene conto dei registi e quindi non esiste il concetto di "artista", ogni attore presenta anche "nome" e "cognome" diversamente da come accade nello schema relazionale.

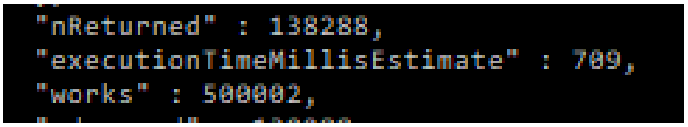
[8] Anche se nell'implementazione su MongoDB non si tiene traccia dei Registi, ho comunque voluto riportare tale campo nei documents della collection *Film*

### 11.2 Operazioni di Ricerca e Inserimenti di Prova

Viene presentata adesso una sequenza di interrogazioni sulle collections. Se necessario verranno fatti confronti con le rispettive query in SQL e riportati anche i tempi di risposta che sono stati stimati grazie al metodo explain("executionStats"). In caso di find semplice il metodo explain deve essere appeso al termine della query:
db.<collection>.find(<query>).explain("executionStats")
Altrimenti, in caso di aggregate, lo stesso metodo deve essere appeso dopo il nome della collection:
db.<collection>.explain("executionStats").aggregate(<query_stages>)

**Interrogazione 1:** Selezionare tutti gli attori che rientrano in categoria "S" o "A" e che hanno fatto il loro esordio dopo l'anno 1978.

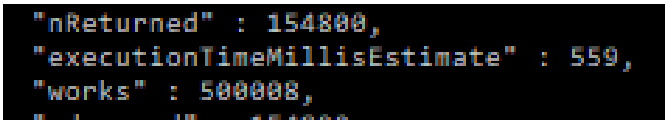
```
db.Attore.find({"categoria": {$in: ["S", "A"]}, "esordio": {$gt: 1978}},
               {nome:1, cognome:1, categoria:1, esordio:1})
```




Quando si esegue il metodo explain("executionStats") su una query, il parametro "nReturned" indica il numero di documents che verranno ritornati, il parametro "works" (se presente) indica il numero di documents che verranno processati e il parametro "executionTimeMillisEstimate" indica la stima (in millisecondi) del tempo di esecuzione, in questo caso è di 0.709 secondi.

**Interrogazione 2:** Selezionare tutti i film di tipo lungometraggio e con durata inferiore a 100 minuti.

```
db.Film.find({"tipo":{$eq: "C"}, "durata": {$lt: 100}},
             {titolo:1, dataUscita:1, tipo:1, durata:1})
```



**Interrogazione 3:** Seleziona nome e cognome degli attori che hanno recitato almeno una volta come protagonisti.

```
db.Attore.aggregate([{"$unwind":"$partecipazioni"},
                     {"$match":{"partecipazioni.ruolo": "Protagonista"}},
                     {"$group": {_id: "$id", nome: {"$max": "$nome"},
                                             cognome: {"$max": "$cognome"}}}])
```

```
> db.Attore.aggregate([{"$unwind":"$partecipazioni"},
... {"$match":{"partecipazioni.ruolo": "Protagonista"}},
... {"$group": {_id: "$id", nome: {"$max": "$nome"}, cognome: {"$max": "$cognome"}}}])
{ "_id" : 494828, "nome" : "Tia", "cognome" : "Upton" }
{ "_id" : 34101, "nome" : "Piper", "cognome" : "Herzog" }
{ "_id" : 139048, "nome" : "Vicky", "cognome" : "Hilpert" }
{ "_id" : 8504, "nome" : "Jevon", "cognome" : "Jacobs" }
{ "_id" : 331261, "nome" : "Alberta", "cognome" : "Romaguera" }
{ "_id" : 454348, "nome" : "Marley", "cognome" : "Kozey" }
{ "_id" : 426840, "nome" : "Fern", "cognome" : "Hodkiewicz" }
{ "_id" : 33732, "nome" : "Kennedi", "cognome" : "Kuhn" }
{ "_id" : 364014, "nome" : "Olin", "cognome" : "Labadie" }
{ "_id" : 458706, "nome" : "Neil", "cognome" : "Wolff" }
{ "_id" : 473957, "nome" : "Donna", "cognome" : "Kessler" }
```

```
"nReturned" : NumberLong(294923),
"executionTimeMillisEstimate" : NumberLong(7995)
```

Dato che lo stage $project non sembra funzionare dopo che è stata eseguita una $group, ho usato l'operatore $max sui campi nome e cognome per poterli visualizzare nell'output. Ho trovato questa strana alternativa a $project online e ho poi anche verificato che restituisse dati corretti, ho eseguito svariate find sugli "id" di alcuni documents restituiti ed effettivamente le informazioni combaciano.

Come si vede dalle statistiche, le interrogazioni che usano aggregate richiedono molto più tempo rispetto le semplici find, per questa query è stata fatta una stima di quasi 8 secondi.

**Interrogazione 4:** Seleziona per ogni ruolo quale è il compenso più alto.

```
db.Attore.aggregate([{"$unwind": "$partecipazioni"},
                     {"$group": {_id: "$partecipazioni.ruolo",
                                 compenso: {"$max": "$partecipazioni.compenso"}}}])
```

```
> db.Attore.aggregate([{"$unwind": "$partecipazioni"},
... {"$group": {_id: "$partecipazioni.ruolo", compenso: {"$max": "$partecipazioni.compenso"}}}])
{ "_id" : "Secondario", "compenso" : 979997 }
{ "_id" : "Comparsa", "compenso" : 980000 }
{ "_id" : "Protagonista", "compenso" : 979999 }
{ "_id" : "CoProtagonista", "compenso" : 979997 }
```

```
"nReturned" : NumberLong(4),
"executionTimeMillisEstimate" : NumberLong(6620)
```

Questa interrogazione è molto simile all'interrogazione numero 18 in MySQL, come si vede dalle statistiche l'aggregate richiede 6.62 secondi e opera su un totale di 500.000 documents, mentre l'interrogazione 18 fatta in SQL richiede 0.14 secondi e opera su un totale di 10.000 record. MongoDB ha gestito 50 volte gli oggetti che ha gestito MySQL e ha richiesto 47 volte il tempo che ha impiegato MySQL.

**Interrogazione 5:** Selezionare gli attori (nome, cognome, categoria) la cui somma dei guadagni sia superiore a $950000 e che rientrano in una delle seguenti categorie: S, A, B.

```
db.Attore.aggregate([{"$match": {categoria: {$in: ["S", "A", "B"]}}},
                     {"$unwind": "$partecipazioni"},
                     {"$group": {_id: "$id",
                     sommaGuadagni: {"$sum": "$partecipazioni.compenso"}}},
                     {"$match": {sommaGuadagni: {$gte: 950000}}}])
```

```
> db.Attore.aggregate([{"$match": {categoria: {$in: ["S", "A", "B"]}}},
... {"$unwind": "$partecipazioni"},
... {"$group": {_id: "$id", sommaGuadagni: {"$sum": "$partecipazioni.compenso"}}},
... {"$match": {sommaGuadagni: {$gte: 950000}}}])
{ "_id" : 439793, "sommaGuadagni" : 2906957 }
{ "_id" : 401881, "sommaGuadagni" : 3143557 }
{ "_id" : 34164, "sommaGuadagni" : 1657435 }
{ "_id" : 317828, "sommaGuadagni" : 2152929 }
{ "_id" : 147206, "sommaGuadagni" : 2357179 }
{ "_id" : 14290, "sommaGuadagni" : 1475088 }
{ "_id" : 218993, "sommaGuadagni" : 1902212 }
{ "_id" : 126632, "sommaGuadagni" : 1659724 }
```

```
"nReturned" : NumberLong(222242),
"executionTimeMillisEstimate" : NumberLong(6069)
```

Questa interrogazione coincide con l'interrogazione SQL numero 11. MongoDB ha impiegato 6 secondi su 500.000 documents, MySQL ha impiegato 0.45

secondi su 10.000 record. MongoDB ha gestito 50 volte gli oggetti gestiti da MySQL e questa volta ha richiesto solo 13 volte il tempo impiegato da MySQL.

**Interrogazione 6:** Selezionare per ogni diversa categoria di attori il numero di attori che rientrano in quella categoria in ordine crescente di numero di attori.

```
db.Attore.aggregate([{"$group": {_id: "$categoria", total: {"$sum":1}}},
                     {"$sort": {total: 1}}])
```

```
> db.Attore.aggregate([{"$group": {_id: "$categoria", total: {"$sum":1}}},
... {"$sort": {total: 1}}])
{ "_id" : "S", "total" : 99684 }
{ "_id" : "B", "total" : 99896 }
{ "_id" : "C", "total" : 99955 }
{ "_id" : "A", "total" : 100123 }
{ "_id" : "D", "total" : 100342 }
>
```

```
"nReturned" : NumberLong(5),
"executionTimeMillisEstimate" : NumberLong(1963)
```

Questa interrogazione coincide con l'interrogazione SQL numero 19. MongoDB ha stimato 1.963 secondi mentre MySQL ha impiegato 0.02 secondi. Il rapporto tra gli oggetti considerati è lo stesso, MongoDB ha stimato 98 volte il tempo che ha impiegato MySQL. Su questa interrogazione sembra che MySQL abbia avuto la meglio.

**Interrogazione 7:** Selezionare i 5 film che hanno incassato di più e che sono in formato "2D".

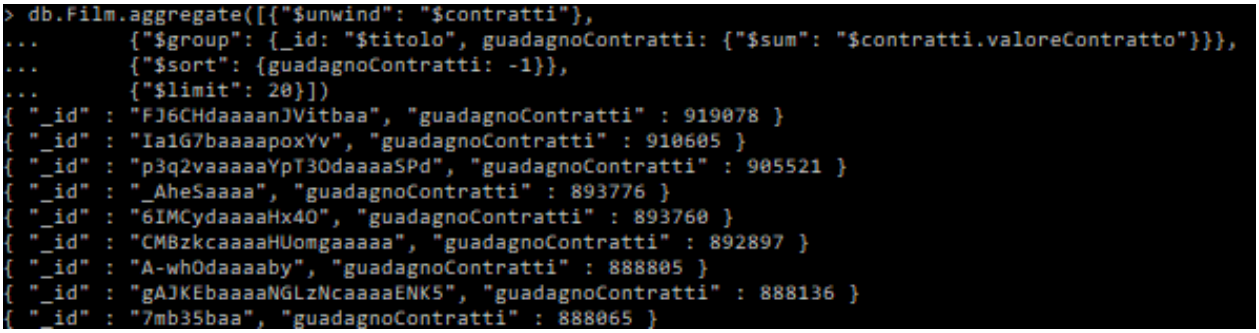
```
db.Film.find({"formato":{$eq: "2D"}},
             {titolo: 1, incasso: 1, formato: 1}).sort({incasso: -1}).limit(5)
```

```
> db.Film.find({"formato":{$eq: "2D"}}, {titolo: 1, incasso: 1, formato: 1}).sort({incasso: -1}).limit(5)
{ "_id" : ObjectId("61320526864ee5deb8f18974"), "incasso" : 1449999, "formato" : "2D", "titolo" : "tk6Fcbaa" }
{ "_id" : ObjectId("61320529864ee5deb8f404c1"), "incasso" : 1449995, "formato" : "2D", "titolo" : "b6lYsaaaa" }
{ "_id" : ObjectId("6132052a864ee5deb8f4f4a9"), "incasso" : 1449991, "formato" : "2D", "titolo" : "GpSbRdaaaa" }
{ "_id" : ObjectId("6132052a864ee5deb8f4f386"), "incasso" : 1449987, "formato" : "2D", "titolo" : "NLVdraaaaa_q9ela" }
{ "_id" : ObjectId("61320526864ee5deb8f0bd27"), "incasso" : 1449985, "formato" : "2D", "titolo" : "VAjh0daaaaGpdpJd" }
>
```

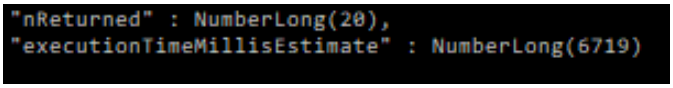
```
"nReturned" : 249858,
"executionTimeMillisEstimate" : 373,
"works" : 500008,
```

**Interrogazione 8:** Selezionare per ogni film la somma dei guadagni dai contratti fatti con le aziende in ordine decrescente di guadagno, limitando i documents ritornati a 20.

```
db.Film.aggregate([{"$unwind": "$contratti"},
                   {"$group": {_id: "$titolo",
                   guadagnoContratti: {"$sum": "$contratti.valoreContratto"}}},
                   {"$sort": {guadagnoContratti: -1}},
                   {"$limit": 20}])
```



```
> db.Film.aggregate([{"$unwind": "$contratti"},
...       {"$group": {_id: "$titolo", guadagnoContratti: {"$sum": "$contratti.valoreContratto"}}},
...       {"$sort": {guadagnoContratti: -1}},
...       {"$limit": 20}])
{ "_id" : "FJ6CHdaaaanJVitbaa", "guadagnoContratti" : 919078 }
{ "_id" : "Ia1G7baaaapoxYv", "guadagnoContratti" : 910605 }
{ "_id" : "p3q2vaaaaaYpT3OdaaaaSPd", "guadagnoContratti" : 905521 }
{ "_id" : "_AheSaaaa", "guadagnoContratti" : 893776 }
{ "_id" : "6IMCydaaaaHx4O", "guadagnoContratti" : 893760 }
{ "_id" : "CMBzkcaaaaHUomgaaaaa", "guadagnoContratti" : 892897 }
{ "_id" : "A-whOdaaaaby", "guadagnoContratti" : 888805 }
{ "_id" : "gAJKEbaaaaNGLzNcaaaaENK5", "guadagnoContratti" : 888136 }
{ "_id" : "7mb35baa", "guadagnoContratti" : 888065 }
```



```
"nReturned" : NumberLong(20),
"executionTimeMillisEstimate" : NumberLong(6719)
```

**Interrogazione 9:** Selezionare per ogni azienda che ha sponsorizzato almeno un film, il suo net worth.

```
db.Film.aggregate([{"$unwind": "$contratti"},
                   {"$lookup": {from: "Azienda",
                   localField: "contratti.nomeAzienda",
                   foreignField: "nomeAzienda", as: "AziendaC"}},
                   {"$unwind": "$AziendaC"},
                   {"$group": {_id: "$AziendaC.nomeAzienda"}},
                   {"$project": {"AziendaC.nomeAzienda": 1,
                                 "AziendaC.netWorth": 1}}])
```

In questa interrogazione ho utilizzato lo stage di $lookup nella *pipeline* dell'aggregate. Questo stage permette di fare ricerche incrociate tra le collections, tuttavia il grande difetto che presenta è che richiede operazioni molto onerose e rallenta incredibilmente l'interrogazione. Il tempo di risposta per quest'interrogazione va dai 2 minuti ai 3 minuti, indubbiamente si tratta di una query molto lenta, il fatto che sta praticamente operando su 1.000.000 di documents in totale (collection *Azienda* + collection *Film*) ha a che fare con gli altissimi tempi di risposta. Rimane comunque interessante vedere MongoDB sottoposto a questo carico a causa dello stage $lookup.

**Interrogazione 10:** Selezionare tutti i film che sono stati sponsorizzati con almeno un contratto di valore superiore a 150000.

```
db.Film.aggregate([{"$unwind": "$contratti"},
                {"$match": {"contratti.valoreContratto": {"$gte": 150000}}},
                {"$group": {_id: "$titolo",
                valoreContratto: {"$max": "$contratti.valoreContratto"}}}])
```

Questa prima soluzione con l'aggregate non è effettivamente la più efficiente per l'interrogazione. Infatti sarebbe meglio sfruttare l'organizzazione ad array dei contratti per utilizzare una find che è sicuramente più veloce dell'aggregate precedente. Nello specifico ho utilizzato la clausola $where e dichiarato una funzione che itera sull'array e filtra i documenti in base a ciò che si sta cercando.

```
db.Film.find({"$where": function(){
                            for(i = 0; i < this.contratti.length; i++){
                                    if(this.contratti[i].valoreContratto >= 150000){
                                            return true;
                                    }
                                    else{
                                            return false;
                                    }
                            }
                        }}).pretty()
```

**Inserimento di Prova:** Di seguito l'inserimento di 3 documenti di prova nella collection *Film*, utilizzando il metodo "runCommand()" e indicando in input l'array che contiene i 3 documents da inserire.

```
> db.runCommand({
... insert: "Film",
... documents: [
...     {titolo: "provaTitolo", dataUscita: "1982-01-01T00:00:00+00:00", durata: 110,
...     incasso: 560000, _idRegista: 2, studio: "LucasFilms", tipo: "L", formato: "2D",
...     contratti: [{nomeAzienda: "provaAzienda", valoreContratto: 89000}]},
...
...     {titolo: "provaTitolo2", dataUscita: "1983-01-01T00:00:00+00:00", durata: 88,
...     incasso: 780000, _idRegista: 3, studio: "provaStudio2", tipo: "C", formato: "2D",
...     contratti: [{nomeAzienda: "provaAzienda2", valoreContratto: 99000},
...     {nomeAzienda: "provaAzienda3", valoreContratto: 119000}]},
...
...         {titolo: "provaTitolo3", dataUscita: "1984-01-01T00:00:00+00:00", durata: 168,
...     incasso: 1400000, _idRegista: 4, studio: "provaStudio3", tipo: "L", formato: "3D",
...     contratti: [{nomeAzienda: "provaAzienda4", valoreContratto: 29000},
...     {nomeAzienda: "provaAzienda5", valoreContratto: 239000}]}
...     ],
...
...     ordered: false})
{ "n" : 3, "ok" : 1 }
>
```

```
> db.Film.find({titolo: /prova/i}, {titolo:1})
{ "_id" : ObjectId("61339360bb68dc7e984c7320"), "titolo" : "provaTitolo" }
{ "_id" : ObjectId("61339360bb68dc7e984c7321"), "titolo" : "provaTitolo2" }
{ "_id" : ObjectId("61339360bb68dc7e984c7322"), "titolo" : "provaTitolo3" }
```