Università degli Studi di Padova

Corso di Laurea in Informatica

Base di Dati per un Sistema di gestione di Serie TV in Streaming

Gruppo: Ceron Tommaso, Parolin Dennis

Anno Accademico 2024/2025

Indice

1 Abstract **2**

2 Analisi dei Requisiti **2**

3 Progettazione concettuale **4**

3.1 Lista entità 4

3.2 Lista relazioni 5

3.3 Lista generalizzazioni 6

3.4 Schema E-R 6

4 Progettazione Logica **6**

4.1 Analisi delle ridondanze 6

4.1.1 Ridondanza "NumeroSerieTv" 6

4.1.2 Ridondanza "NumeroUtenti" 8

4.2 Eliminazione delle generalizzazioni 9

4.3 Schema relazionale 10

5 Implementazione in PostgreSQL e Definizione delle Query **11**

5.1 Creazione degli indici 14

1 Abstract

Questo progetto sviluppa una base di dati per gestire delle serie tv, composte da stagioni ed episodi, e dai loro attori e registi. Inoltre, questa base di dati permette di gestire degli utenti e gli episodi che guardano, ai quali possono dare un voto.
L'obbiettivo principale è quello di gestire la struttura delle serie tv, gestire le relazioni di quest' ultime con il cast e il regista che vi partecipa e infine registrare ciò che un utente guarda.
La base di dati distingue le persone in utenti, registi e attori, ognuno con degli attributi e relazioni distinte.
Una serie si compone di almeno una stagione, che a sua volta è composta da almeno un episodio.
Opzionalmente una Serie TV può anche contenere una Opening (ovvero una sigla iniziale).
Infine, il sistema tiene traccia delle piattaforme di streaming che mettono a disposizione una determinata serie tv, e degli utenti che sottoscrivono un abbonamento con uno o più di queste piattaforme.

2 Analisi dei Requisiti

In questa sezione riassumiamo i requisiti che caratterizzano la base di dati.

Serie TV. Una serie TV è identificata da un titolo e un anno e contiene questi attributi:

- Titolo
- Anno di Inizio
- Descrizione
- Genere

Una Serie TV è composta da una o più stagioni, e da zero o una Opening (sigla iniziale)

Stagioni. Una stagione è identificata dal suo numero e dalla Serie Tv a cui appartiene, e contiene questi attributi:

- Numero
- Anno

Una stagione è composta da uno o più episodi

Episodio. Un episodio è identificato dal suo numero e dalla stagione a cui appartiene, e contiene questi attributi:

- Numero
- Durata (in minuti)

- Titolo
- Numero di utenti (che hanno guardato l' episodio)

Opening. Una opening è identificata da un titolo, e contiene questi attributi:

- Titolo
- Compositore
- Durata

Persona. Una persona generica è identificata da un codice fiscale e contiene i seguenti attributi:

- Codice Fiscale
- Nome
- Data di nascita

Una persona si suddivide in **Utente**, **Nome** e **Registi**

Utenti. Un utente, oltre agli attributi di *Persona* contiene:

- Username
- Email

Attore. Un attore, oltre agli attributi di *Persona*, contiene:

- Nazionalita'
- Il numero di film in cui ha partecipato

Regista. Un regista, oltre agli attributi di *Persona*, contiene:

- Nazionalita
- Il genere di film creati

Piattaforma streaming. Una piattaforma streaming e' identificata dal suo nome e contiene i seguenti attributi:

- Nome
- Costo mensile

3 Progettazione concettuale

3.1 Lista entità

Il database si compone delle seguenti entità:

- **SerieTV:** Rappresenta una Serie Tv univoca, con una o più stagioni.
  - Titolo: string
  - AnnoInizio (ovvero l'anno della prima stagione): int
  - Genere: string
  - Descrizione: string
- **Stagione:** Rappresenta una stagione di una specifica Serie Tv
  - Numero_stagione: int
  - Anno: int
- **Episodio:** Rappresenta un singolo episodio di una specifica stagione
  - Numero_episodio: int
  - Durata (in minuti): int
  - Titolo: string
  - Numero_utenti (che hanno guardato l'episodio): int
- **Opening:** Rappresenta la sigla iniziale di una SerieTv, se presente.
  - Titolo: string
  - Durata: int
  - Compositore: string
- **Persona:** Rappresenta una persona generica.
  - Codice Fiscale: int
  - Nome: string
  - Data_nascita: date
- **Utente:** Rappresenta una specializzazione di Persona, rappresenta un abbonato a una o più piattaforme streaming
  - Username: string
  - Email: string
- **Attore:** Rappresenta una specializzazione di Persona, partecipa a una o più serie TV.
  - Nazionalità: string
  - Numero_serie (in cui ha partecipato): int

- **Regista:** Rappresenta una specializzazione di Persona, ha diretto una o più serie TV.
  - Nazionalita: string
  - Genere_serie: string
- **Piattaforma streaming:** Rappresenta una piattaforma che contiene una o più serie Tv, e a cui gli utenti possono abbonarsi con vari tipi di abbonamento.
  - Nome: string
  - Costo_mensile: float

3.2 Lista relazioni

- Serie TV⇒(1, N)⇒ComposizioneStag⇐(1,1)⇐Stagione
  - Una Serie Tv è composta da una o più stagioni
  - Una stagione compone una sola Serie Tv
- Stagione⇒(1, N)⇒ComposizioneEP⇐(1,1)⇐Episodio
  - Una stagione è composta da uno o più episodi
  - Un episodio compone una sola stagione
- Serie TV⇒(1,1)⇒Contenimento⇐(1, N)⇐Piattaforma-Streaming
  - Una Serie Tv è contenuta (può essere vista) in una sola piattaforma streaming
  - Una piattaforma streaming può contenere una o più serie tv
- Piattaforma-Streaming⇒(1, N)⇒Sottoscrizione⇐(1, N)⇐Utente
  - Ad una piattaforma streaming si sottoscrivono uno o più utenti
  - Un utente può sottoscriversi a una o più piattaforme streaming (minimo una per essere considerato utente)
  - Questa relazione contiene i seguenti attributi:
    - Data_inizio: date
    - Data_fine: date
    - Tpo_abbonamento: string
- Utente⇒(0, N)⇒Visiona⇐(0, N)⇐Episodio
  - Un utente può vedere 0 o più episodi di stagioni differenti
  - Un episodio generico può essere visto da nessuno o da più utenti
  - Questa relazione contiene i seguenti attributi:
    - Data: date
    - Voto: int

    Il voto è opzionale.

- Serie TV⇒(1, N)⇒Perfomance⇐(1, N)⇐Attore
  - In una serie tv recitano uno o più attori
  - Un attore può recitare in una o più serie tv e riceve un compenso per ogni serie.
  - Questa relazione contiene i seguenti attributi:
    * Ruolo: string
    * Compenso: int
- Serie TV⇒(1,1)⇒Direzione⇐(1, N)⇐Regista
  - Una serie tv è diretta da un solo regista
  - Un regista può dirigere una o più serie
- Serie Tv⇒(0,1)⇒ContenimentoOpening⇐(1,1)⇐Opening
  - Una serie tv contiene zero o una opening
  - Una opening appartiene a una sola Serie Tv

3.3 Lista generalizzazioni

Persona è una generalizzazione totale di **UTENTE**, **ATTORE** e **REGISTA**

3.4 Schema E-R

In figura 1 è riportato il diagramma E-R che riassume i requisiti discussi in precedenza.

4 Progettazione Logica

In questa sezione illustriamo la traduzione dello schema concettuale in schema logico, per rappresentare i dati nel modo più efficace possibile. Andiamo quindi ad analizzare le ridondanze, per comprendere se sia preferibile eliminarle o mantenerle. Procediamo poi con l'eliminazione delle generalizzazioni e infine mostriamo lo schema ristrutturato.

4.1 Analisi delle ridondanze

4.1.1 Ridondanza "NumeroSerieTv"

Possiamo notare che attributo "Numero Serie Tv" del sottoinsieme "Attore" entità "Persona" può essere calcolato dalla relazione Perfomance, poiché è il conteggio di tutte le Serie Tv per quel determinato attore. Questo attributo viene modificato (almeno una volta) ogni volta che viene inserita una nuova SerieTv (stimiamo circa 5 nuovi inserimenti al giorno) e viene visualizzato ogni volta un utente vuole sapere in quanti film ha recitato un attore (stimiamo 25 al giorno). Il tutto si riassume nelle seguenti operazioni:

- **Operazione 1 (5 al giorno)**: inserimento di una nuova tupla in Perfomance

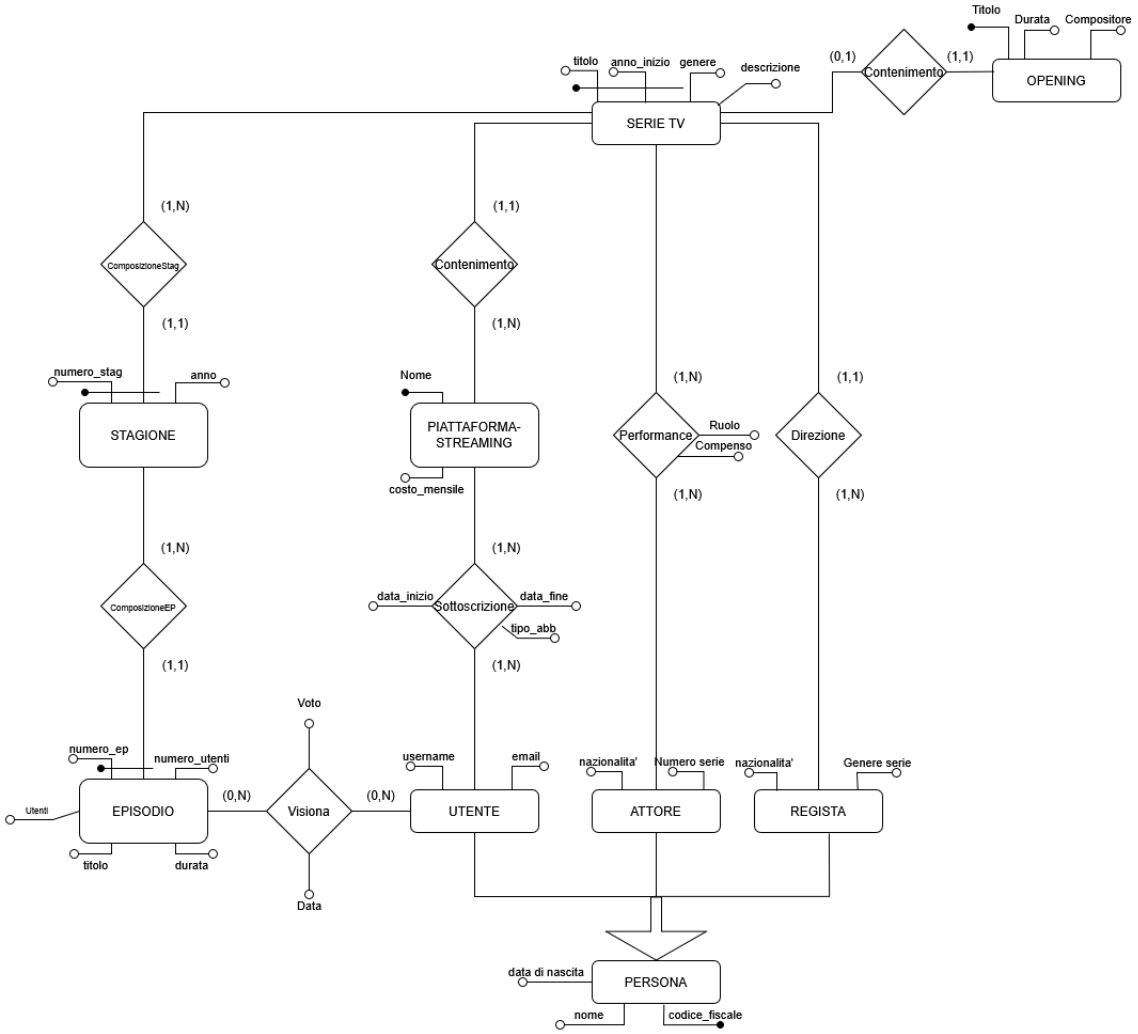

Figura 1: Schema E-R non ristrutturato

- **Operazione 2 (25 al giorno)**: Visualizzare in quanti film ha recitato un attore.

Assumiamo i seguenti dati:

Concetto	Costrutto	Volume
Attore	Entità	1000
Performance	Relazione	2000

Con Ridondanza

- Operazione 1

Concetto	**Costrutto**	**Accessi**	**Tipo**	
Serie TV	E	1	S	$\times 5$
Perfomance	R	1	S	$\times 5$
Attore	E	1	L	$\times 5$
Attore	E	1	S	$\times 5$

- Operazione 2

Conecetto	Costrutto	Accessi	Tipo	
Attore	E	1	L	$\times 25$

Assumiamo costo doppio per le operazioni in scrittura, il costo totale ottenuto è: $((5 \times 3) \times 2) + 5 + 25 = 60$

Senza Ridondanza

- Operazione 1

Conecetto	Costrutto	Accessi	Tipo	
Serie Tv	E	1	S	$\times 5$
Performance	R	1	S	$\times 5$

- Operazione 2
  Stimiamo circa 2.000 performance attoriali

Conecetto	Costrutto	Accessi	Tipo	
Attore	E	1	L	$\times 25$
Performance	R	2.000	L	$\times 25$

Assumiamo costo doppio per le operazioni in scrittura, il costo totale ottenuto è: $((5 \times 2) \times 2) + 25 + (2000 \times 25) \times 2 = 100.045$
Questo calcolo permette di stabilire che mantenere la ridondanza è preferibile, in modo da ottimizzare gli accessi.

4.1.2 Ridondanza "NumeroUtenti"

L' attributo “Numero utenti” dell'entità Episodio (relativa al numero di utenti che hanno guardato l'episodio) può essere calcolato a partire dalla relazione “Visione”, in quanto è il conteggio di tutti gli utenti che hanno guardato quell'episodio. L'attributo viene modificato ogni volta che un utente guarda un episodio e viene visualizzato 50 volte al giorno per controllare l'andamento della piattaforma. Il tutto si riassume nelle seguenti operazioni:

- **Operazione 1 (10.000 al giorno)**: inserimento di una nuova tupla in Visiona
- **Operazione 2 (50 al giorno)**: Visualizzare il numero di utenti che hanno guardato un episodio.

Assumiamo i seguenti dati:

Concetto	Costrutto	Volume
Episodio	Entità	10.000
Visiona	Relazione	1.000.000

Con ridondanza

- Operazione 1

Concetto	Costrutto	Accessi	Tipo	
Visione	R	1	S	$\times 10.000$
Episodio	E	1	L	$\times 10.000$
Episodio	E	1	S	$\times 10.000$

- Operazione 2

Conecetto	Costrutto	Accessi	Tipo	
Utenti	E	1	L	$\times 25$

Senza ridondanza

- Operazione 1

Conecetto	Costrutto	Accessi	Tipo	
Visione	R	1	S	$\times 10.000$

  Assumiamo costo doppio per le operazioni in scrittura, il costo totale ottenuto è: $(10.000 \times 2) \times 2 + 10.000 + 50 = 50.050$

- Operazione 2

Conecetto	Costrutto	Accessi	Tipo	
Episodio	E	1	L	$\times 200$
Perfomance	R	1.000.000	L	$\times 200$

  Assumiamo costo doppio per le operazioni in scrittura, il costo totale ottenuto è: $(10.000 \times 2) + 200 + (200 \times 2) = 20.600$

In questo caso invece è preferibile eliminare la ridondanza.

4.2 Eliminazione delle generalizzazioni

Persona è una generalizzazione totale di **UTENTE**, **ATTORE** e **REGISTA**, presenta tre attributi *Nome*, *Data di nascita* e *Identificatore*, ma non è legata a nessuna relazione. Si nota quindi come questa generalizzazione possa essere eliminata. È preferibile infatti accorpare i sue attributi nelle entità figlie. In questo modo si riduce il numero di attributi NULL rispetto all'operazione inversa, ovvero accorpare gli attributi delle entità figlie in una sola entità padre.
Inoltre, e' possibile eliminare l' attributo "codice_fiscale" nell' entità "UTENTE" in quanto possiamo utilizzare l' attributo "username" come chiave primaria. Inoltre conoscere il codice fiscale di un utente puo' essere considerato sostanzialmente insensato e rischioso per la sicurezza.
In figura 2 è riportato il diagramma E-R ristrutturato, in cui sono state effettuate le modifiche discusse precedentemente.

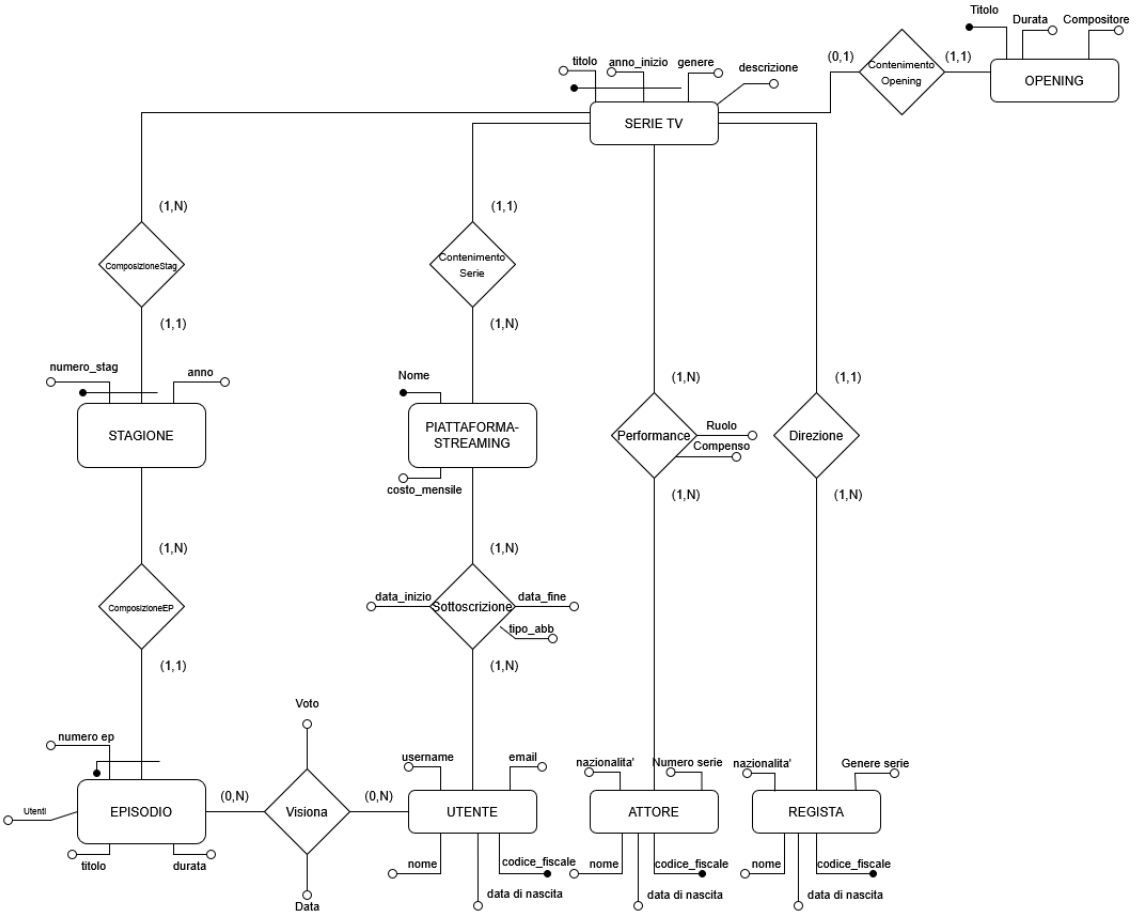

Figura 2: Schema E-R ristrutturato

4.3 Schema relazionale

In seguito è riportato lo schema logico costruito a partire dallo schema E-R ristrutturato. L'asterisco indica gli attrributi che possono essere NULL.

- **Serie**(Titolo, AnnoInizio, Genere, Descrizione, Regista, PiattaformaStreaming, Opening)
  - Serie.Regista=Regista.CodiceFiscale
  - Serie.PiattaformaStreaming=PiattaformaStreaming.Nome
- **Stagione**(TitoloSerie, AnnoSerie, NumeroStagione, Anno)
  - Stagione.TitoloSerie=Serie.Titolo
  - Stagione.AnnoSerie=Serie.AnnoInizio
- **Episodio**(TitoloSerie, AnnoSerie, NumeroStagione, NumeroEpisodio, Durata, Titolo)
  - Episodio.TitoloSerie=Stagione.TitoloSerie
  - Episodio.AnnoSerie=Stagione.AnnoSerie
  - Episodio.NumeroStagione=Stagione.NumeroStagione
- **Opening**(TitoloSerie, AnnoSerie, Titolo, Compositore, Durata)

- **Visualizzazione**(Username, TitoloSerie, AnnoSerie, NumeroStagione, NumeroEpisodio, Data, Voto*)
  - Visualizzazione.Username=Utente.Username
  - Visualizzazione.TitoloSerie=Episodio.TitoloSerie
  - Visualizzazione.AnnoSerie=Episodio.AnnoSerie
  - Visualizzazione.NumeroStagione=Episodio.NumeroStagione
  - Visualizzazione.NumeroEpisodio=Episodio.Numero
- **Utente**(Username, Nome, DataNascita, Email)
- **Attore**(CodiceFiscale, Nome, DataNascita, Nazionalita, NumeroSerie)
- **Regista**(CodiceFiscale, Nome, DataNascita, Nazionalita, GenereSerie)
- **PiattaformaStreaming**(Nome, CostoMensile)
- **Sottoscrizione**(Username, Piattaforma, DataInizio, DataFine*, TipoAbbonamento)
  - Sottoscrizione.Username=Utente.Username
  - Sottoscrizione.Piattaforma=PiattaformaStreaming.Nome
- **Performance**(IDAttore, TitoloSerie, AnnoSerie, Ruolo, Compenso)
  - Performance.Attore=Attore.ID
  - Perfomance.TitoloSerie=Serie.Titolo
  - Performance.AnnoSerie=Serie.AnnoInizio

5 Implementazione in PostgreSQL e Definizione delle Query

Definizione delle Query

Di seguito vengono presentate e descritte le query con i relativi output generati e viene motivato l'utilizzo dell'indice proposto.

Query 1

Obiettivo: Trovare tutti i titoli delle serie TV disponibili su una piattaforma specifica (ad esempio *Netflix*), la sua opening (se presente) e il numero di stagioni di ognuna.

```
SELECT stv.titolo, COUNT(*) as numero_stagioni, op.titolo
FROM serie_tv stv
JOIN stagione st
ON st.titolo_serie=stv.titolo AND st.anno_serie=stv.anno_inizio
JOIN opening op
```

```
ON op. titolo_serie=stv.titolo AND op.anno_serie=stv.anno_inizio
WHERE piattaforma_streaming= 'Netflix'
GROUP BY stv.titolo, op.titolo
```

Listing 1: Query 1

Query 2

Obiettivo: Calcolare la media dei voti per ogni episodio di una serie specifica, visualizzando solo gli episodi con voto maggiore di 7,5.

```
SELECT E.TITOLO_episodio, AVG(V.Voto) as MediaVoto
FROM Episodio E
JOIN VISUALIZZAZIONE V ON
        E.Titolo_Serie = V.Titolo_Serie AND
        E.Anno_Serie = V.Anno_Serie AND
        E.Numero_Stagione = V.Numero_Stagione AND
        E.Numero_Episodio = V.Numero_Episodio
WHERE E.Titolo_Serie = 'The Bear'
GROUP BY E.Titolo_episodio
HAVING AVG(V.Voto)>7.5
```

Listing 2: Query 2

Query 3

Obiettivo: Visualizzare le visualizzazioni dell'ultimo mese e il numero di utenti iscritti a ogni piattaforma.

```
SELECT
    ps.nome AS piattaforma,
        vstats.totale_visualizzazioni AS totale_visualizzazioni,
        sstats.utenti_iscritti AS utenti_iscritti
FROM
    PIATTAFORMA_STREAMING ps

LEFT JOIN (
    SELECT
        st.piattaforma_streaming AS piattaforma,
        COUNT(*) AS totale_visualizzazioni
    FROM
        VISUALIZZAZIONE v
    JOIN
```

```
        SERIE_TV st ON v.titolo_serie = st.titolo AND v.anno_serie = st.
            anno_inizio
    WHERE
        v.data BETWEEN DATE '2025-04-20' AND DATE '2025-05-20'
    GROUP BY
        st.piattaforma_streaming
) AS vstats ON ps.nome = vstats.piattaforma

LEFT JOIN (
    SELECT
        nome_piattaforma AS piattaforma,
        COUNT(DISTINCT username) AS utenti_iscritti
    FROM
        SOTTOSCRIZIONE
    GROUP BY
        nome_piattaforma
) AS sstats ON ps.nome = sstats.piattaforma

ORDER BY
    totale_visualizzazioni DESC;
```

Listing 3: Query 3

Query 4

Obiettivo: Trovare i 10 attori con il compenso più alto, e la relativa serie TV.

```
SELECT DISTINCT a.nome, p.compenso, stv.titolo
FROM attore AS a
JOIN performance AS p
ON p.id_attore=a.cod_fiscale
JOIN serie_tv as stv
ON stv.titolo=p.titolo_serie and stv.anno_inizio=p.anno_serie
ORDER BY p.compenso DESC
LIMIT 10
```

Listing 4: Query 4

Query 5

Obiettivo: Trovare le prime 3 serie TV (per numero totale di visualizzazioni) dirette da registi italiani e disponibili su piattaforme con costo mensile minore a 13 euro.

```
SELECT stv.titolo, COUNT(*) AS Visualizzazioni, r.nome, stv.
    piattaforma_streaming as PiattaformaStreaming
FROM serie_tv AS stv
JOIN Visualizzazione AS v
ON stv.titolo=v.titolo_serie and stv.anno_inizio=v.anno_serie
JOIN regista AS r
ON r.cod_fiscale=stv.regista
JOIN piattaforma_streaming AS p
ON p.nome=stv.piattaforma_streaming
WHERE r.nazionalita='Italy' and p.costo_mensile<13
GROUP BY stv.titolo, r.nome, stv.piattaforma_streaming
ORDER BY Visualizzazioni DESC
LIMIT 3
```

Listing 5: Query 5

5.1 Creazione degli indici

Vogliamo ottimizzare la query 3, che calcola il numero di visualizzazioni per ogni piattaforma streaming nell'ultimo mese e il numero di utenti iscritti a ogni piattaforma.
In questo caso abbiamo deciso di creare un indice sulla tabella **Visualizzazione** per velocizzare le operazioni di join e di filtro. Le visualizzazioni infatti sono l'elemento maggiormente presente nel nostro database (oltre 10.000 tuple), e verranno spesso filtrate per data, quindi è utile avere un indice di tipo B+ Tree che ci permette di accedere più rapidamente alle tuple che soddisfano il filtro temporale. L'indice creato è il seguente:

```
CREATE INDEX idx_visualizzazione_data ON VISUALIZZAZIONE(data);
```

Listing 6: Creazione indice