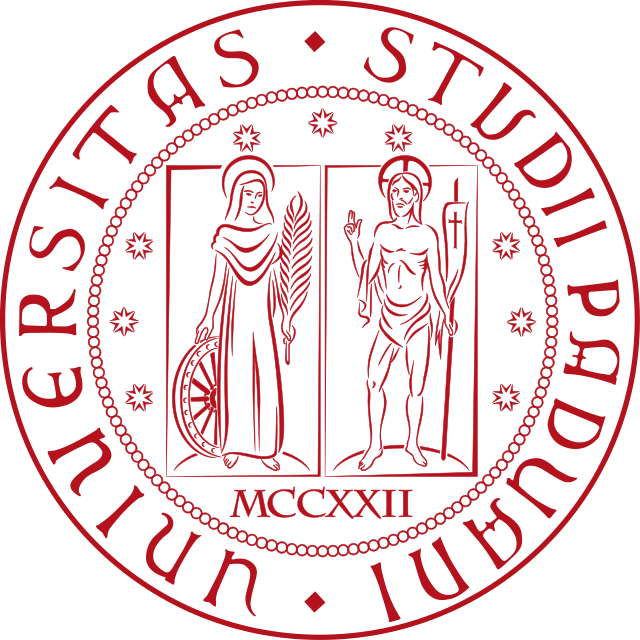
**Università degli studi di Padova**

**Anno Accademico 2023/24**

****

Progetto Basi di Dati

**Videoteche Manager**

Realizzato da:

Alberto Canavese - N. Matricola 2076423

Cheng Feng Liao - N. Matricola INSERIRE NUMERO MATRICOLA

**1 Abstract**

Questo sistema di gestione per le videoteche consente di organizzare e archiviare informazioni importanti. Queste informazioni riguardano i registi dei film, le videoteche stesse, il personale che lavora nelle videoteche, i film su supporto fisico e/o digitale, i clienti, i dettagli dei comuni e altro ancora.

Le tabelle all'interno del database sono interconnesse attraverso relazioni, permettendo di collegare le informazioni in modo significativo: ad esempio, si possono collegare un regista ai film che ha diretto, associare i film alle videoteche in cui sono disponibili, tenere traccia delle iniziative organizzate dalle videoteche.

Questo sistema consente di gestire una varietà di attività all'interno delle videoteche, come il noleggio dei film, l'organizzazione di eventi e la gestione dei dati relativi ai registi e ai clienti, come, per esempio, i dettagli delle carte di credito utilizzate dai clienti.

**2 Analisi dei requisiti**

**Descrizione testuale**

Nel database sono presenti i dati dei **registi** che hanno diretto almeno in film che è presente in una delle videoteche contenuta nella base di dati.

Di ogni regista sono noti:

* Codice fiscale (CF)
* Nome
* Cognome
* Data di nascita
* Nazionalità

Un regista può dirigere film e concederli a una o più videoteche presenti nel database.

Di ogni videoteca si conoscono i seguenti attributi:

* Partita IVA
* Nome
* Orario di apertura
* Orario di chiusura
* Indirizzo

Gli indirizzi di ogni videoteca fanno riferimento a dei **comuni** dei quali si conoscono:

* Nome
* Provincia
* Numero di abitanti

Le varie videoteche sono gestite da **gestori** dei quali si conoscono i seguenti dettagli:

* Codice fiscale (CF)
* Nome
* Cognome
* Data di nascita
* Nazionalità
* Stipendio

Da ogni videoteca un **cliente** può noleggiare sia copie fisiche, sotto forma di **DVD**, oppure, tramite un sito web, è può noleggiare i film fruibili direttamente dal web per un periodo limitato di tempo.

Se un DVD è già in prestito ad un altra persona, non sarà possibile per un cliente noleggiare lo stesso per il medesimo periodo ti tempo.

Di ogni **DVD** si conosce:

* ID
* Titolo
* Genere
* Lingua
* Anno di pubblicazione
* Durata in minuti

Invece, per ogni **copia digitale**, sono noti:

* ID
* Titolo
* Genere
* Lingua
* Anno di pubblicazione
* Durata in minuti
* Dimensioni in GB

Di ogni **cliente** si conoscono le seguenti informazioni:

* Codice fiscale (CF)
* Nome
* Cognome
* Data di nascita

Ogni cliente deve avere una **carta di credito** come garanzia, di questa vengono salvati:

* Numero
* Data di scadenza

Le copie fisiche dei DVD vengono rese disponibili dai **fornitori,** dei quali si deve conoscere:

* Nome
* Sede

Infine, ogni videoteca può proporre delle iniziative culturali come i **cineforum**, dei quali servono i seguenti dettagli:

* Data
* Nome film

**3 Progettazione concettuale**

**3.1 Lista entità**

se non specificato l’attributo è *non null*

* **Persone**:

CF: varchar(100) primary key

Nome: varchar (100)

Data di nascita: date

Nazionalità varchar (100)

L’entità “persone” si divide in 3 dottogerarchie:

Regista:

Nome comune: varchar(100) foreign key

Clienti:

Nome comune: varchar(100) foreign key

Gestore:

Stipendio: double precision

Nome videoteca: varchar(100) foreign key

Nome comune: varchar(100) foreign key

* **Film:**

ID: varchar(100) primary key

Titolo: varchar(100)

Genere: varchar(100)

Anno: date

Minuti: interger

Lingua: varchar(100)

L’entità “film” si divide in 2 sottogerarchie:

DVD:

CF regista: varchar(100) foreign key

Nome fornitore: varchar (100) foreign key

Nome videoteca: varchar(100) foreign key

File:

GB: integer

CF regista: varchar(100) foreign key

* **Fornitori:**

Nome: varchar(100) primary key

Sede centrale: varchar(100)

* **Videoteca:**

P. IVA: varchar(100) primary key

Nome: varchar(100)

Orario di apertura: time (6) without time zone

Orario di chiusura: time (6) without time zone

Indirizzo: varchar(100)

Nome comune: varchar(100) foreign key

* **Cineforum:**

Data: date primary key

Nome film: varchar(100)

* **Comune:**

Nome: varchar(100) primary key

Provincia: varchar(100)

Numero abitanti: INT

* **Carta di credito:**

Numero: varchar(100) primary key

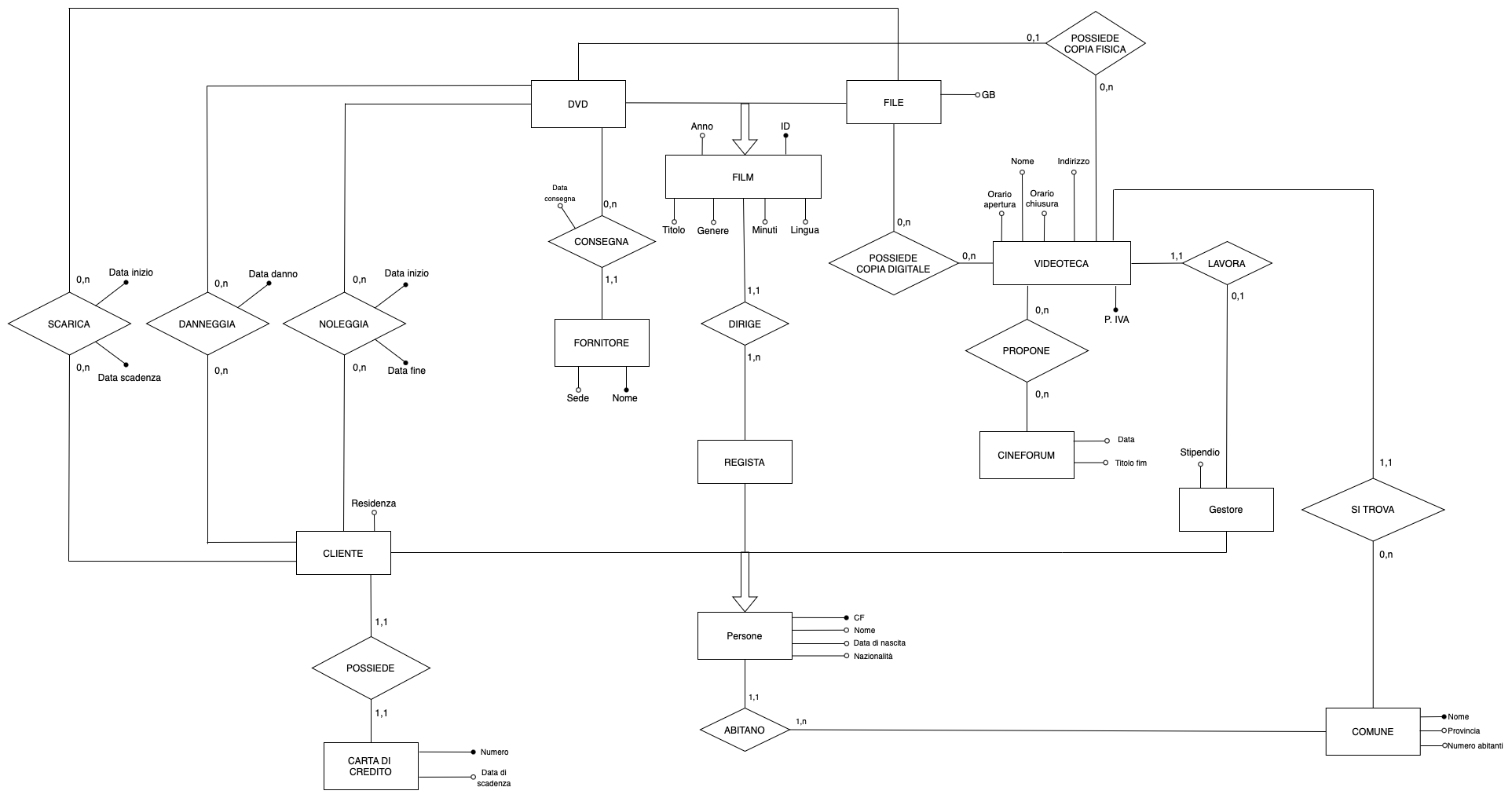
Data di scadenza: date

|  |  |  |  |
| --- | --- | --- | --- |
| **Relazione** | **Entità coinvolte** | **Descrizione** | **Attributi** |
| Scarica | File (0,n)  Cliente (0,n) | Un cliente può scaricare più film, ma anche zero.  Un file può essere scaricato da zero o più clienti. | Data inizio: date  data scadenza: date |
| Danneggia | DVD (0,n)  Cliente (0,n) | Un cliente può danneggiare più DVD, ma anche zero.  Un DVD può essere danneggiato da zero o più clienti. | Nessuno |
| Noleggia | DVD (0,n)  Cliente (0,n) | Un cliente può noleggiare più DVD, ma anche zero.  Un DVD può essere noleggiato da zero o più clienti. | Data inizio: date  Data fine: date |
| Consegna | Fornitore (1,1)  DVD (0,n) | Ogni DVD viene consegnato da un solo fornitore.  Un fornitore può consegnare zero o più DVD | Data consegna: date |
| Possiede | Cliente (1,1)  Carta di credito (1,1) | Ogni cliente è associato a una sola carta di credito.  Ogni carta di credito è associata ad un solo cliente | Nessuno |
| Dirige | Regista (1,n)  Film (1,1) | Un regista può dirigere uno o più film.  Un film può essere diretto da un solo regista. | Nessuno |
| Propone | Videoteca (0,n)  Cineforum (0,n) | Una videoteca può proporre nessun o molti cineforum.  Un cineforum può essere proposto da nessuna o molte videoteche. | Nessuno |
| Possiede copia digitale | Videoteca (0,n)  File (0,n) | Una videoteca può possedere zero o molti file.  Un file può essere posseduto da zero o molte videoteche | Nessuno |
| Possiede copia fisica | Videoteca (0,n)  DVD (0,1) | Un DVD è posseduto da una sola videoteca.  Una videoteca può possedere zero o più DVD. | Nessuno |
| Si trova | Videoteca (1,1)  Comune (0,n) | Una videoteca si trova solo in un comune.  In un comune ci possono essere zero videoteche | Nessuno |
| Lavora | Gestore (0,1)  Videoteca (1,1) | Ogni gestore può gestire una o zero videoteche.  Ogni videoteca può e deve avere solo un gestore. | Nessuno |
| Abitano | Persone (1,1)  Comune (1,n) | Una persona abita in un solo comune.  Ogni comune è abituato da una o più persone | Nessuno |

**3.2 Tabella delle relazioni**

**Vincoli non rappresentabili tramite schema ER:**

* Ogni videoteca è gestita da una sola persona e ogni gestore gestisce solo una videoteca.
* Lo stesso DVD non può essere noleggiato da più persone durante la stessa finestra di tempo.
* Lo stesso film può essere scaricato da più utenti allo stesso tempo.

**Schema ER:**

**4 Progettazione logica**

**4.1 Ristrutturazione**

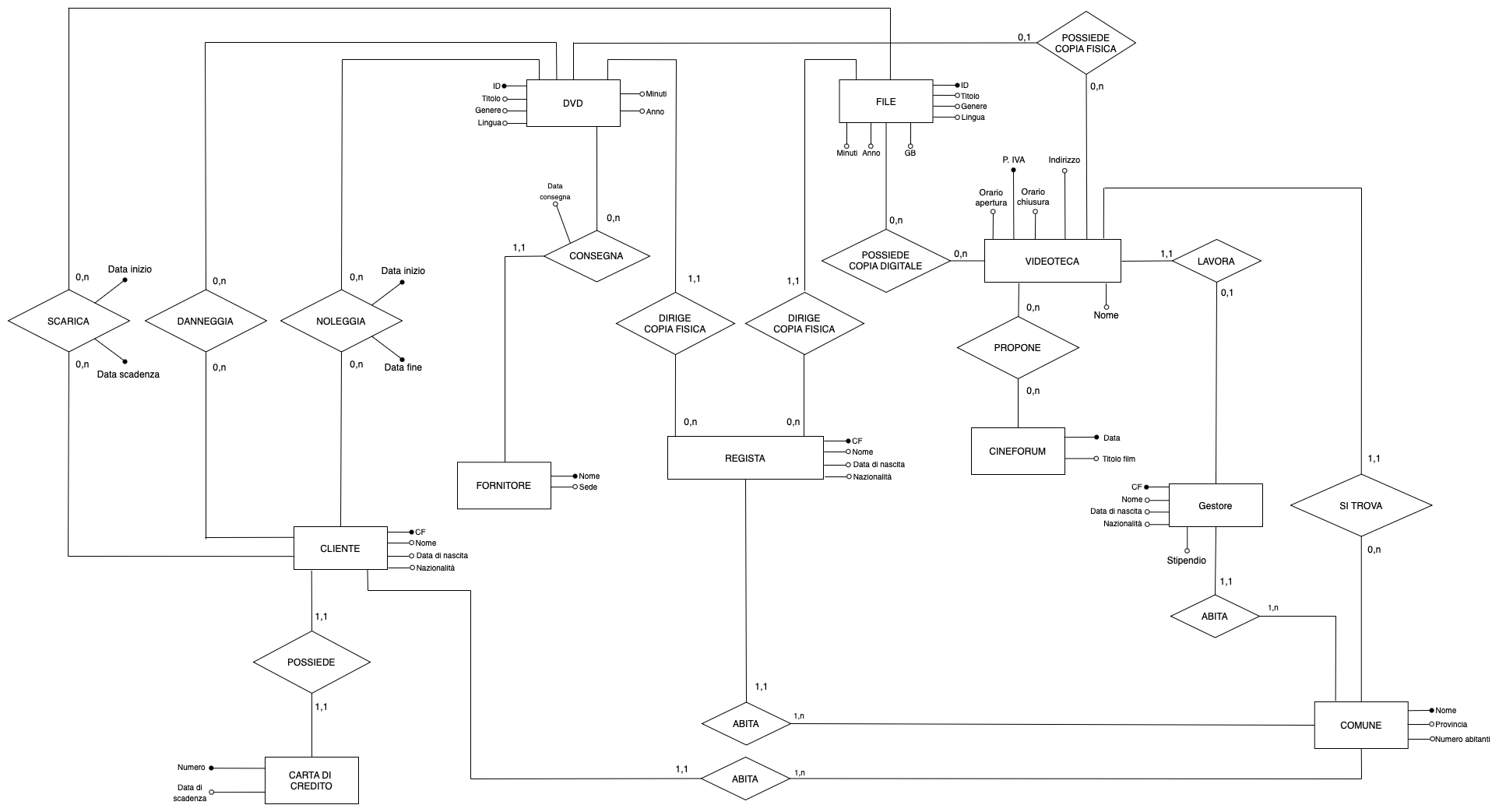
* **Eliminazione delle generalizzazioni**

|  |  |
| --- | --- |
| **Generalizzazione** | **Risoluzione** |
| Regista, cliente, gestore → Persone | L’entità Persone viene rimossa, i sui attributi vengono  assegnati alle entità figlie.  La relazione *“ABITA”* viene assegnata  indipendentemente ad ognuna delle 3 entità figlie. |
| DVD, file → film | L’entità Libri viene rimossa, i sui attributi vengono assegnati  alle entità **DVD** e **file**. La relazione “*DIRIGE*” viene duplicata |

* **Eliminazione delle ridondanze**

|  |  |
| --- | --- |
| **Ridondanza** | **Risoluzione** |
| Cliente::Residenza  Il comune di residenza di un cliente è ridondante in quanto esiste la tabella “Comune”.  Si può facilmente risalire alla residenza di un cliente tramite la tabella “Comune”. | Rimozione dell’attributo “Residenza” dalla tabella “Cliente” |
| Danneggia::data danno  La finestra di tempo in cui un DVD può essere stato danneggiato è reperibile grazie alla relazione “Noleggia”.  Quindi l’attributo “Data danno” è ridondante | Rimozione dell’attributo “Data danno” dalla relazione “Danneggia” |

* **Schema ER ristrutturato:**



**4.2 Creazione delle tabelle**

*(A→B indica che B è chiave esterna di A)*

**Comune** (Nome, Provincia, Numero\_ abitanti)

**Regista** (CF, nome, Data\_di\_nascita, Nazionalità, Nome\_comune → comune.Nome)

**Videoteca** (P\_IVA, Nome, Orario\_apertura, Orario\_chiusura, Indirizzo,

Nome\_comune → comune.Nome)

**Gestore** (CF, Nome, data\_di\_nascita, nazionalità, Nome\_comune → Comune.nome,

P\_IVA\_videoteca → Videoteca.P\_IVA)

**DVD** (ID, Titolo, Genere, Lingua, Minuti, Anno, CF\_cliente → Cliente.CF,

CF\_regista → Regista.CF, Nome\_fornitore → Fornitore.nome

Nome\_videoteca → Videoteca.nome)

**File** (ID, Titolo, Genere, Lingua, Minuti, Anno, GB, CF\_cliente → Cliente.CF)

**Cliente** (CF, Nome, Data\_di\_nascita, Nazionalità, Nome\_comune → comune.Nome)

**Danneggia** (ID\_copia → DVD.ID, CF\_cliente → Cliente.CF)

**Carta di credito** (Numero, Data\_di\_scadenza, CF\_cliente → Cliente.CF)

**Fornitore** (Nome, sede\_centrale)

**Cineforum** (Data, titolo film)

**Propone** (P\_IVA\_videoteca → Videoteca.P\_IVA, Data→ cineforum.Data)

**Scarica** (ID\_film → File.ID, CF\_cliente → Cliente.CF, Data\_inizio, Data\_scadenza)

**Noleggia** (ID → DVD\_ID, data\_inizio, data\_fine, CF\_cliente → Cliente.CF)

**5 Query e indici**

**5.1 Queries**

* **Numero totale di noleggi effettuati in ciascuna videoteca:**

SELECT v.Nome AS Videoteca, COUNT(n.ID) AS Totale\_Noleggi

FROM Noleggia n

JOIN DVD d ON n.ID = d.ID

JOIN Videoteca v ON d.Nome\_videoteca = v.Nome

GROUP BY v.Nome

ORDER BY Totale\_Noleggi DESC;



* **Numero totale di DVD danneggiati per ciascun cliente:**

SELECT c.Nome AS Cliente, COUNT(dg.ID\_copia) AS Totale\_Danneggiati

FROM Cliente c

JOIN Danneggia dg ON c.CF = dg.CF\_cliente

GROUP BY c.Nome

ORDER BY Totale\_Danneggiati DESC;



* **Numero di film scaricati per ciascun cliente e l'importo totale di spazio occupato in GB:**

SELECT c.Nome AS Cliente, COUNT(s.ID\_film) AS Totale\_Scaricati, SUM(f.GB) AS Totale\_GB

FROM Cliente c

JOIN Scarica s ON c.CF = s.CF\_cliente

JOIN File f ON s.ID\_film = f.ID

GROUP BY c.Nome

ORDER BY Totale\_Scaricati DESC;



* **Registi che hanno diretto più di un film insieme al numero totale di film diretti**

SELECT r.Nome AS Regista, COUNT(d.ID) AS Totale\_Film

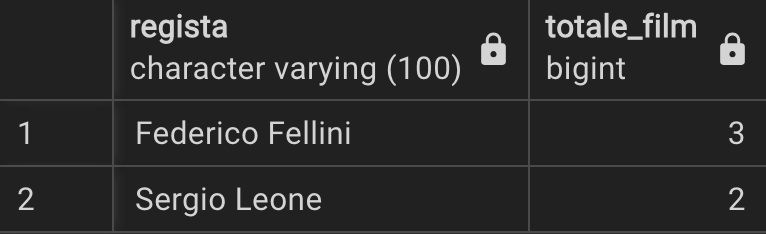
FROM Regista r

JOIN DVD d ON r.CF = d.CF\_regista

GROUP BY r.Nome

HAVING COUNT(d.ID) > 1

ORDER BY Totale\_Film DESC;



* **Numero medio di noleggi per cliente per ciascuna videoteca:**

SELECT v.Nome AS Videoteca, AVG(n.Noleggi\_Per\_Cliente) AS Media\_Noleggi\_Per\_Cliente

FROM (

SELECT c.CF, d.Nome\_videoteca, COUNT(n.ID) AS Noleggi\_Per\_Cliente

FROM Cliente c

JOIN Noleggia n ON c.CF = n.CF\_cliente

JOIN DVD d ON n.ID = d.ID

GROUP BY c.CF, d.Nome\_videoteca

) AS n

JOIN Videoteca v ON n.Nome\_videoteca = v.Nome

GROUP BY v.Nome

ORDER BY Media\_Noleggi\_Per\_Cliente DESC;



**5.2 Indici**

Per la definizione di un indice prendiamo in esame la query 5: “*Numero medio di noleggi per cliente per ciascuna videoteca*”.

**Motivazione per la scelta dell’indice**:

Questa query aggrega i dati dalla tabella **Noleggia** e **DVD** per contare il numero totale di noleggi per ciascuna videoteca. L'indicizzazione delle colonne utilizzate nelle condizioni di join e di aggregazione può migliorare significativamente le prestazioni della query, riducendo i tempi di scansione delle tabelle.

**Creazione dell'indice sulla tabella Noleggia per ottimizzare le ricerche di noleggi per DVD**

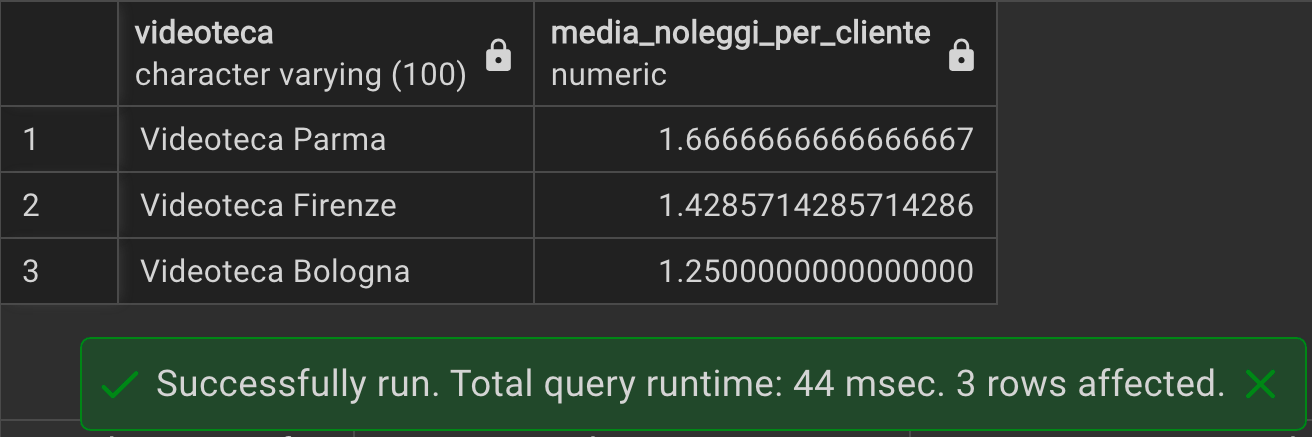
CREATE INDEX idx\_noleggia\_id ON Noleggia (ID);

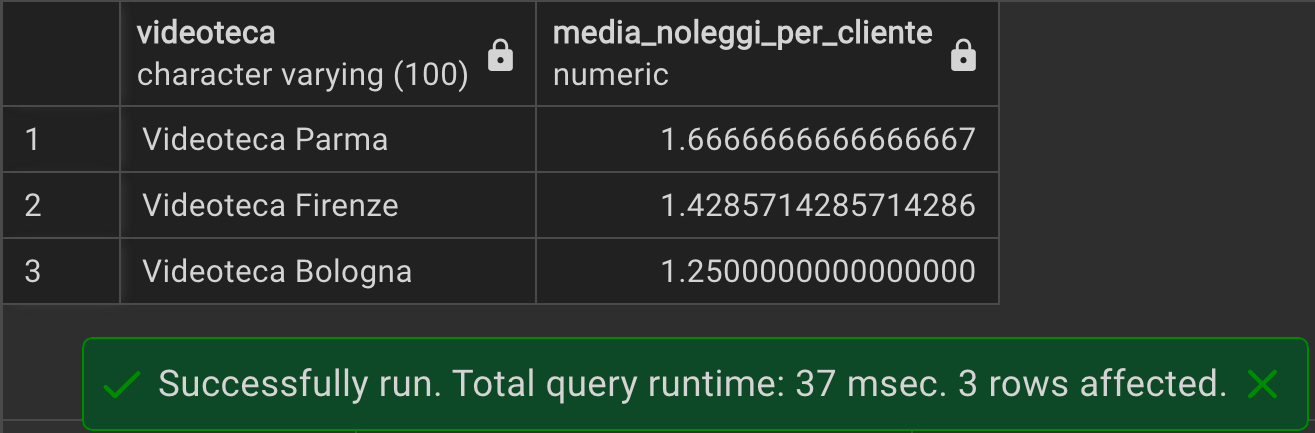
**Creazione dell'indice sulla tabella DVD per ottimizzare le ricerche di noleggi per videoteca**

CREATE INDEX idx\_dvd\_id\_nome\_videoteca ON DVD (ID, Nome\_videoteca);

Le queries originali e gli indici definiti contribuiranno a migliorare le prestazioni complessive delle operazioni di lettura del database, specialmente per queries che richiedono l'aggregazione e il join su tabelle grandi.

Includendo questi indici, si garantisce che le operazioni di ricerca e aggregazione siano eseguite in modo più efficiente, riducendo i tempi di risposta per le queries più complesse.

*Tempo impiegato per eseguire la query 5 senza indici:*

*Tempo impiegato per eseguire la query 5 con gli indici appena descritti:*

Come si può notare il tempo d’esecuzione della query è inferiore quando vengono utilizzati degli indici.

**6 Codice C**

Il codice C per l’esecuzione delle query consiste in un unico file .c che va compilato attraverso il comando:

**INSERIRE COMANDO COMPILAZIONE**

Prima di poter compilare il codice è necessario inserire i file libpq.dll, libpq.lib, libpq-fe.h,

pg\_config\_ext.h, postgres\_ext.h nella cartella designata e modificare il comando precedente

cambiando <percorso\_cartella> con la cartella dove sono stati inseriti i file

Sullo schermo apparirà il seguente Menù con la lista delle query, identificate da 1 a 6, mentre lo 0 terminerà.

Alcune query richiedono l’inserimento di dati aggiuntivi da parte dell’utente, in particolare:

* La query numero 1 restituisce tutti i gestori che lavorano in videoteche situate in comuni con più di un certo numero di abitanti. L’utente deve inserire il numero di abitanti.
* La query numero 2 restituisce tutti i registi che hanno diretto DVD con più di un certo numero di minuti e i loro rispettivi fornitori. All’utente è richiesto di inserire il minutaggio.
* La questi numero 6 restituisce i cineforum organizzati in videoteche con orari di apertura precedenti ad un certo orario. L’utente deve inserire un orario di apertura

Prima della compilazione è necessario sostituire nelle variabili globali **PG\_HOST**, **PG\_USER**, **PG\_DB**, **PG\_PASS**, **PG\_PORT** rispettivamente l’indirizzo dell’host, il nome utente di PostgreSQL, il nome del database, la password e la porta utilizzata.

Il programma utilizza la funzione PQconnectdb per connettersi al database, la funzione PQstatus per verificare lo stato della connessione, la funzione PQfinish per terminare la connessione.

**AMBIENTE DI SVILUPPO**

|  |  |
| --- | --- |
| **Alberto Canavese** | **Cheng Feng Liao** |
| SO: MacOS 14.2 |  |
| Editor C: Visual Studio Code |  |
| Compilatore: g++ 14.0.3 |  |
| Editor SQL: PostgreSQL 16.3 |  |