Laboratorio Interdisciplinare B A.A. 2024/2025

# Book Recommender

# Manuale DATABASE

**Autori:**

Abou Aziz Sara Hesham Abdel Hamid

Hidri Mohamed Taha

Zoghbani Lilia

Ben Mahjoub Ali

Versione 2.0

# Indice


1. Introduzione ..... 4
2. Premessa ..... 4
3. Raccolta ed analisi dei requisiti ..... 4
4. Carico di lavoro previsto ..... 5
5. Schema scheletro ..... 6
6. Schema concettuale ..... 8
6.1. Associazioni Ternarie ..... 9
7. Schema Concettuale Ristrutturato ..... 9
8. Normalizzazione ..... 9
9. Traduzione dello schema ..... 9
9.1. Traduzione delle associazioni ..... 10
10. Schema Logico ..... 11
10.1. Traduzione dei vincoli di integrità ..... 11
11. Indici ..... 13
12. Query SQL ..... 14
12.1 Creazione del databas ..... 14
12.2 Creazione delle tabelle ..... 14
12.2. Operazioni CRUD ..... 16
12.2.1. UtentiRegistrati ..... 16
- **Salva un Nuovo Utente** ..... 16
- **Trova un Utente** ..... 16
- **Aggiorna i Dati Utente** ..... 16
- **Elimina un Utente** ..... 16
- **Seleziona Tutti gli Utenti** ..... 16

12.2.2. Libri ..... 16
12.2.3. Librerie ..... 17
- **Selezione Librerie per ID Utente** ..... 17
- **Selezione Libreria per Utente e Nome** ..... 17
- **Aggiornamento Libreria** ..... 17
- **Cancellazione Libreria** ..... 17
- **Inserimento Libro in Libreria** ..... 17
- **Cancellazione Libro da Libreria** ..... 17
- **Selezione Libri in Libreria** ..... 17
- **Verifica Libro in Libreria** ..... 17
- **Verifica Nome Libreria Esistente** ..... 17

12.2.4. ValutazioniLibri ..... 17
12.2.6. ConsigliLibri..... 19
- Inserimento ..... 19

## 1. Introduzione

“Book Recommender” è un sistema per la valutazione e raccomandazione di libri, che permette agli utenti registrati di inserire recensioni e a tutti gli utenti di consultare le valutazioni e ricevere consigli di lettura. Per ulteriori informazioni sul funzionamento del programma, incluse le istruzioni per l’avvio, consultare il Manuale Utente.

## 2. Premessa

Questo manuale descrive il meccanismo di persistenza utilizzato, documentando le diverse fasi del suo sviluppo. In particolare, il manuale include:

- la raccolta ed analisi dei requisiti per la persistenza dei dati;
- i vari schemi di sviluppo, tra cui:
    - schema scheletro;
    - schema concettuale;
    - schema concettuale ristrutturato;
    - schema logico.
- altre scelte architetturali riguardanti il database e la persistenza dei dati, come gli indici utilizzati;
- tutte le query coinvolte per la creazione del database, delle tabelle, degli indici e per il recupero dei dati

Per ulteriori informazioni circa le scelte architetturali e funzionali del sistema software, consultare il Manuale Tecnico. Si raccomanda di leggere la premessa di tale manuale e successivamente questo documento.

## 3. Raccolta ed analisi dei requisiti

I committenti hanno richiesto lo sviluppo di un’applicazione client-server in Java, adottando come meccanismo di persistenza il database relazionale PostgreSQL.

Il primo passo per poter determinare lo schema di base di dati è l’individuazione delle entità coinvolte nel sistema software. In base alle specifiche fornite, sono state individuate le seguenti entità cardine nel sistema di per la valutazione e raccomandazione di libri “Book Recommender”:

- **Utenti non registrati**: possono consultare liberamente le informazioni sui libri e visualizzare valutazioni e consigli in forma aggregata. Trattandosi di un accesso anonimo, non è necessario memorizzarne i dati.
- **Utenti registrati**: oltre a fruire di tutte le funzionalità degli utenti non registrati, possono creare librerie personali, inserire valutazioni e consigli di lettura. Per la registrazione e l’autenticazione è necessario memorizzare:
    - nome
    - cognome
    - codice fiscale (vincolo di unicità)
    - indirizzo e-mail
    - userid (identificatore visibile e univoco scelto dall’utente)
    - password (conservata in forma sicura, es. hash)

Oltre al userId, viene introdotto anche un campo id interno, generato automaticamente, che funge da chiave primaria tecnica. Questa scelta consente di separare la logica interna di gestione dalla rappresentazione visibile all'utente, migliorando la coerenza progettuale, la flessibilità del sistema e le performance in fase di accesso ai dati.

- **Libro**: rappresenta un'entità del sistema corrispondente a un volume disponibile per la valutazione e la raccomandazione. Ogni libro è identificato da un ID univoco e descritto dai seguenti attributi:
  - titolo
  - autori
  - anno di pubblicazione
  - descrizione
  - categorie
  - editore
  - prezzo
- **Libreria**: collezione personalizzata di libri creata da un utente registrato. Ogni libreria è identificata da un libreria_id tecnico (generato automaticamente) ed è associata a una coppia univoca composta da user_id e nome_libreria, per evitare conflitti tra utenti diversi. La scelta di usare un libraryId interno come chiave primaria risponde all'esigenza di mantenere semplice ed efficiente la gestione del database, evitando chiavi composte e facilitando eventuali modifiche future ai nomi delle librerie senza impatti sulle relazioni.
- **Valutazione Libro**: espresso da un utente registrato per uno specifico libro all'interno della propria libreria. Ciascuna valutazione include:
  - punteggi interi (1–5) per i criteri: stile, contenuti, gradimento, originalità, qualità, valutazione complessiva
  - commenti testuali facoltativi (max 256 caratteri)
  - riferimento al userid dell'utente e all'id del libro
- **Consiglio:** suggerimento di lettura fornito da un utente registrato per un libro presente nella propria libreria. Ogni consiglio comprende:
  - commento facoltativo (max 256 caratteri)
  - lista dei libri consigliati (riferimenti agli id dei libri suggeriti, max 3)

## 4. Carico di lavoro previsto

Il database dell'applicazione **"Book Recommender"** è progettato per gestire un ampio volume di dati relativi a libri, utenti e interazioni associate (valutazioni e raccomandazioni).

Nella versione attuale il sistema supporta oltre 100.000 libri.

Si prevede un numero potenzialmente elevato di **utenti registrati**, ciascuno in grado di interagire attivamente con la piattaforma attraverso operazioni di consiglio e valutazione dei libri.

Ogni utente può esprimere più valutazioni e consigli nel tempo, generando un numero crescente di record nelle tabelle dei consigli e delle valutazioni, che costituiranno due delle componenti pi dinamiche del database.

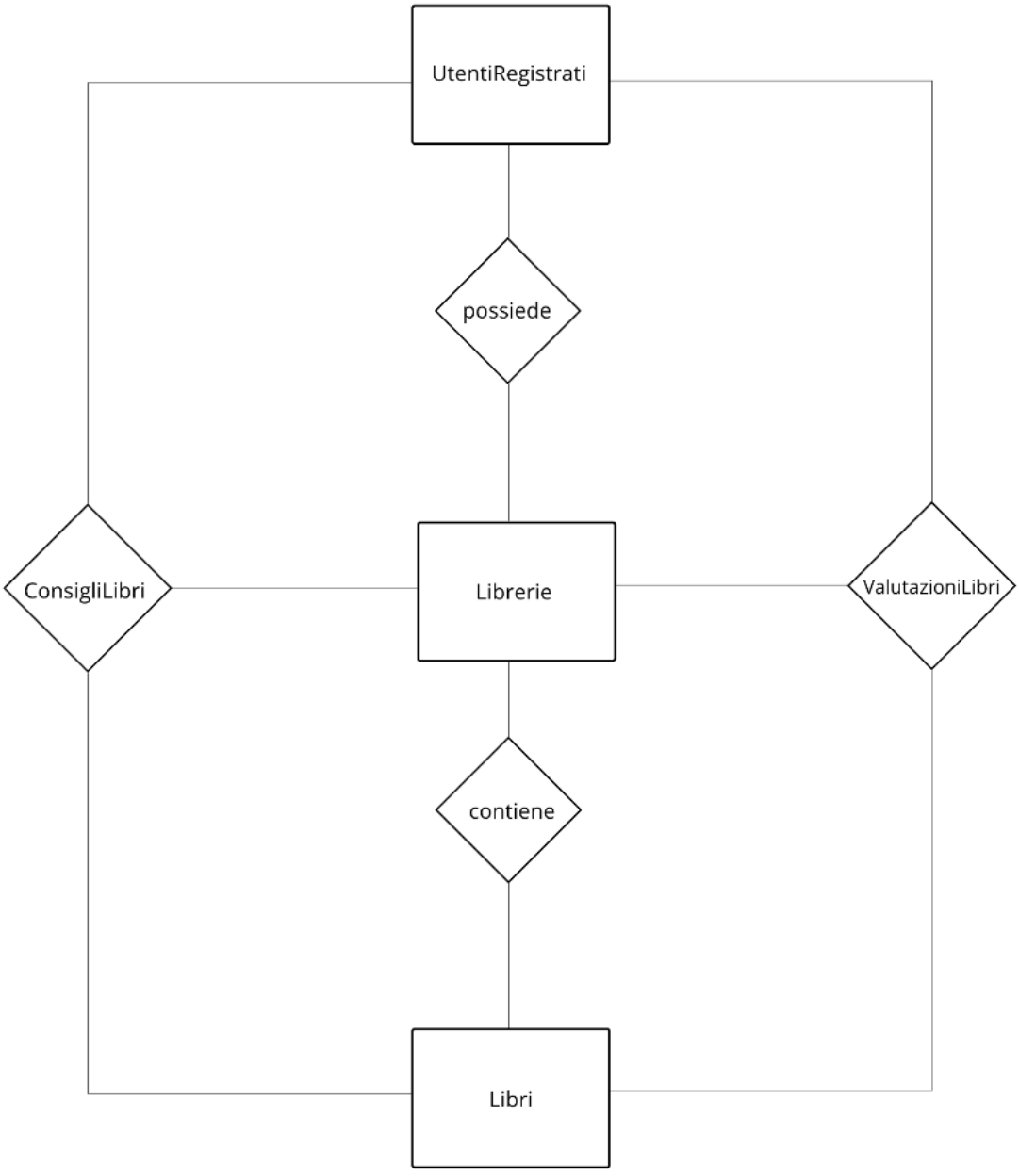
Il carico di lavoro del sistema sarà composto principalmente da:

- **Operazioni di lettura**, relative alla visualizzazione delle schede libro, delle raccomandazioni generate e delle valutazioni aggregate
- **Operazioni di scrittura**, prodotte dalle valutazioni e dai consigli inseriti dagli utenti e dalla registrazione di nuove utenze.

Il sistema è concepito per supportare **l'accesso concorrente** da parte di più utenti, garantendo la consistenza delle operazioni e prestazioni adeguate anche in condizioni di utilizzo intensivo. Il progetto del database deve pertanto privilegiare **strutture indicizzate**, **integrità referenziale** e **ottimizzazione delle query**, in modo da assicurare tempi di risposta contenuti e affidabilità dell'intero sistema.

## 5. Schema scheletro

Sulla base delle specifiche, è stato elaborato il seguente schema scheletro:



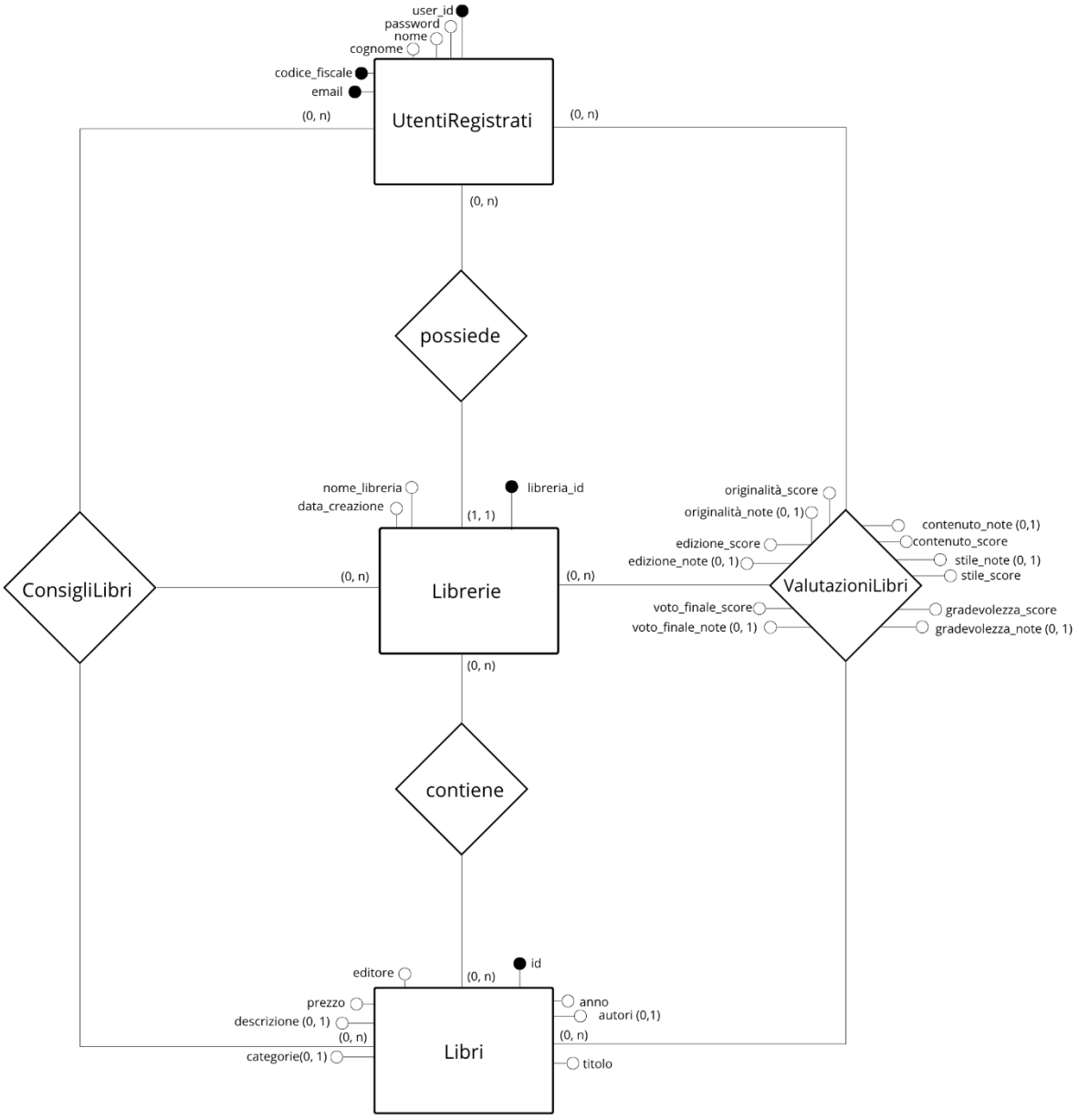
La nomenclatura utilizzata per le entità riflette una richiesta precisa dei committenti.

Le relazioni individuate nello schema ER sono le seguenti:

- *Possiede*: un utente registrato possiede nessuna o più librerie personali; ogni libreria appartiene a un singolo utente registrato.
- *Contiene*: una libreria contiene uno o più libri; un libro può essere contenuto in una o più librerie.
- *ValutazioneLibri* : associazione ternaria tra UtenteRegistrato, Libreria e Libro. Una valutazione è espressa da un singolo utente su un singolo libro nel contesto di una specifica libreria dell'utente; un utente può rilasciare zero o più valutazioni, e un libro può riceverne molte da diversi utenti. Vincolo principale: una tupla ValutazioneLibri(user, libreria, libro, …) è ammessa solo se l'utente possiede quella libreria (Possiede(user,libreria)) e la libreria contiene quel libro (Contiene(libreria,libro)). I punteggi sono interi nell'intervallo [1..5] e i commenti sono ≤ 256 caratteri.
- *ConsigliLibri*: associazione ternaria tra UtenteRegistrato, Libreria e Libro. Un consiglio indica che un utente, nel contesto della propria libreria, suggerisce un determinato libro; uno stesso utente può inserire più consigli (max 3), e uno stesso libro può essere suggerito da più utenti. Vincoli principali: ogni tupla ConsigliLibri(user, libreria, libro, …) è ammessa solo se Possiede(user,libreria) e Contiene(libreria,libro); inoltre un utene non pùò suggerire più di 3 libri per ogni libro presente nelle sue librerie.

## 6. Schema concettuale

Nel passaggio da schema scheletro a concettuale sono stati aggiunti i vincoli di cardinalità (di cui sopra), gli attributi e i vincoli di identificazione:




**V.I.**

Vincolo di inclusione per Valutazioni:

una tupla ValutazioniLibri(userId, libreriaId, bookId, ...) è ammessa solo se Possiede(userId, libreriaId) e Contiene(libreriaId, bookId)

Vincolo di inclusione per Consigli:

una tupla ConsigliLibri(userId, libreriaId, bookId, ...) è ammessa solo se Possiede(userId, libreriaId) e Contiene(libreriaId, bookId)

Vincolo massimo 3 suggerimenti:

il limite massimo di suggerimenti (≤3) per il raggruppamento richiesto verrà formalizzato a livello di ristrutturazione tramite trigger o gestito a livello applicativo, come specificato nella sezione implementativa.

Vincoli di valore:

i punteggi devono essere interi nell'intervallo [1,5]; i testi di commento hanno lunghezza ≤256.

## 6.1. Associazioni Ternarie

Modellare **ValutazioneLibri** e **ConsigliLibri** come associazioni ternarie (Utente — Libreria — Libro) è una scelta concettuale e pratica che rispecchia fedelmente il dominio funzionale del sistema e offre benefici concreti.

Dal punto di vista semantico, entrambe le informazioni hanno significato solo in un contesto: una valutazione o un consiglio non sono fatti isolati sul libro, ma sono fatti **espressi da un utente in relazione a una specifica libreria**.

Rappresentarle come associazioni rende questa dipendenza esplicita nel modello concettuale, evita ambiguità e facilita la lettura della documentazione.

Dal punto di vista dell'integrità dei dati, la modellazione a associazione si traduce naturalmente, nella ristrutturazione relazionale, in una tabella ValutazioniLibri(user_id, libreria_id, book_id, ...) e in una tabella ConsigliLibri(...) che includono una **foreign key composita** (libreria_id, book_id) verso la tabella ponte Libreria_Libro. Tale vincolo referenziale permette al database di garantire direttamente la regola di business *"si può valutare/suggerire solo un libro che è effettivamente contenuto in una propria libreria"*, senza dover dipendere esclusivamente dalla logica applicativa.

Inoltre, la scelta riduce la ridondanza e semplifica la normalizzazione: la relazione cattura in modo naturale la tripla (utente, libreria, libro) evitando tabelle intermedie con duplicazione di informazioni.
Infine, pur essendo concettualmente associazioni, nella fase di implementazione i record saranno salvati nelle tabelle richieste dai committenti (ValutazioniLibri e ConsigliLibri).

Questa combinazione offre sia chiarezza concettuale sia robustezza operativa: il modello riflette correttamente il dominio e il database diventa l'ultima linea di difesa contro inconsistenze.

# 7. Schema Concettuale Ristrutturato

Nella ristrutturazione dello schema concettuale dato che non sono presenti attributi composti o multi-valore né gerarchie di generalizzazione, non sono stati effettuati cambiamenti.

# 8. Normalizzazione

Non si è ritenuta necessaria una normalizzazione dello schema concettuale ristrutturato.

# 9. Traduzione dello schema

Di seguito viene presentata la traduzione dello schema del database, che descrive le tabelle principali e le loro relazioni:

**UtentiRegistrati**(user_id, password, nome, cognome, codice_fiscale, email)
**Libri**(id, titolo, autori, anno, descrizione, categorie, editore, prezzo)
**Librerie**(libreria_id, user_id $^{\text{UtentiRegistrati}}$, nome_libreria, data_creazione)
**Libreria_Libro** (libreria_id $^{\text{Librerie}}$, book_id)

**ValutazioniLibri**(user_id [UtentiRegistrati], libreria_id [Librerie], libro_id [Libri], stile_score, contenuto_score, gradimento_score, originalita_score, qualita_score, voto_complessivo, stile_note, contenuto_note, gradimento_note, originalita_note, qualita_note, data_valutazione)
**ConsigliLibri**( user_id [UtentiRegistrati], libreria_id [Librerie], libro_letto_id [Libri], libro_consigliato_id [Libri], commento)

## 9.1. Traduzione delle associazioni

Le associazioni molti-a-molti, ovvero quelle relative a ValutazioniLibri, ConsigliLibri e Contiene, sono state tradotte nelle rispettive tabelle di collegamento (ValutazioniLibri, ConsigliLibri e Libreria_Libro), utilizzando come chiavi primarie gli identificatori delle tabelle che collegano. Ciascuna di queste tabelle contiene anche chiavi esterne che fanno riferimento alle tabelle originali.
L'unica associazione uno-a-molti, denominata Possiede, è stata tradotta come chiave esterna, seguendo la prassi consolidata.
Non sono presenti associazioni uno-a-uno.

## 10. Schema Logico

Lo schema logico finale prevede la traduzione dello schema concettuale ristrutturato nelle effettive tabelle della base di dati:


### 10.1. Traduzione dei vincoli di integrità

I vincoli di integrità sono stati definiti nelle tabelle del database per garantire la coerenza dei dati e il rispetto delle relazioni tra le entità. I principali vincoli di integrità implementati sono i seguenti:

- **Chiavi primarie**: Ogni tabella ha una chiave primaria che garantisce l'unicità dei record.
  - UtentiRegistrati: user_id (VARCHAR(50))
  - Libri: id (BIGINT)
  - Librerie: libreria_id (SERIAL, autoincrementale)
  - Libreria_Libro: chiave composta da (libreria_id, libro_id)

    - ValutazioniLibri: chiave composta da (user_id, libreria_id, libro_id)
    - ConsigliLibri: chiave composta da (user_id, libreria_id, libro_letto_id, libro_consigliato_id)
- **Chiavi esterne**: Le relazioni tra le tabelle sono garantite da chiavi esterne.
    - Librerie → UtentiRegistrati (tramite user_id)
    - Libreria_Libro → Librerie (tramite libreria_id)
    - Libreria_Libro → Libri (tramite libro_id)
    - ValutazioniLibri → UtentiRegistrati (tramite user_id)
    - ValutazioniLibri → Libreria_Libro (tramite la coppia libreria_id, libro_id)
    - ConsigliLibri → UtentiRegistrati (tramite user_id)
    - ConsigliLibri → Libreria_Libro (tramite la coppia libreria_id, libro_letto_id)
    - ConsigliLibri → Libri (tramite libro_consigliato_id)
- **Vincoli di integrità referenziale**: Le chiavi esterne sono accompagnate da vincoli che stabiliscono azioni da eseguire in caso di eliminazione dei record correlati.
    - **ON DELETE CASCADE**: Questa azione propaga l'eliminazione. Se un record nella tabella "padre" viene eliminato, vengono eliminati automaticamente anche tutti i record correlati nelle tabelle "figlie". È stato utilizzato per:
        - Eliminare le librerie, le valutazioni e i consigli di un utente quando l'utente viene eliminato (UtentiRegistrati → Librerie, ValutazioniLibri, ConsigliLibri).
        - Eliminare i libri contenuti in una libreria quando la libreria viene eliminata (Librerie → Libreria_Libro).
        - Eliminare le valutazioni e i consigli associati a un libro in una specifica libreria quando il libro viene rimosso da quella libreria (Libreria_Libro → ValutazioniLibri, ConsigliLibri).
        - Eliminare un consiglio se il libro consigliato viene rimosso dal database (Libri → ConsigliLibri).
    - **ON DELETE RESTRICT**: Questa azione impedisce l'eliminazione. Non è possibile eliminare un record dalla tabella "padre" se esistono record correlati nella tabella "figlia". È stato utilizzato per:
        - Impedire l'eliminazione di un libro dalla tabella Libri se quel libro è presente in almeno una libreria utente (Libri → Libreria_Libro). Questo garantisce che non vengano creati riferimenti "orfani" nelle librerie.
        - Sebbene gli utenti finali non possano eliminare libri dal catalogo generale, questo vincolo agisce come una misura di sicurezza per prevenire la rimozione accidentale di un libro a cui gli utenti fanno già riferimento, preservando così l'integrità dei dati delle loro librerie personali.
- **Vincoli di unicità**: Sono stati definiti vincoli di unicità per evitare duplicazioni di dati che non fanno parte della chiave primaria.
    - UtentiRegistrati: I campi codice_fiscale ed email devono essere unici in tutta la tabella.
    - Librerie: La coppia (user_id, nome_libreria) deve essere unica. Questo impedisce a un utente di creare due librerie con lo stesso nome.
- **Vincoli di integrità dei dati**: Sono stati utilizzati vincoli per garantire la validità e la presenza obbligatoria dei dati.

- **NOT NULL**: Assicura che una colonna non possa contenere valori NULL. È stato applicato a tutte le chiavi primarie e a campi fondamentali
  come password, nome, cognome, email in UtentiRegistrati e titolo in Libri.
- **CHECK**: Valida che i valori inseriti in una colonna rispettino una condizione specifica. Nella tabella ValutazioniLibri, è stato usato per garantire che tutti i punteggi (stile_score, contenuto_score, etc.) siano un intero compreso tra 1 e 5.

## 11. Indici

Oltre agli indici generati automaticamente dal DBMS per i vincoli di PRIMARY KEY e UNIQUE (come su user_id, email, codice_fiscale, etc.), sono stati creati indici espliciti su colonne specifiche per ottimizzare le prestazioni delle query più frequenti e critiche per l'applicazione. La scelta di questi indici è stata guidata dalle funzionalità offerte all'utente e dalla necessità di garantire tempi di risposta rapidi.

Gli indici aggiunti sono:

- **Indici sulla tabella Libri:**
  - idx_libri_titolo sulla colonna titolo.

    ```
    CREATE INDEX idx_libri_titolo ON Libri(titolo);
    ```

  - idx_libri_autori sulla colonna autori.

    ```
    CREATE INDEX idx_libri_autori ON Libri(autori);
    ```

  - Questi due indici sono fondamentali per accelerare le operazioni di ricerca testuale effettuate dall'utente nell'interfaccia "Cerca Libri". Senza di essi, ogni ricerca comporterebbe una scansione completa (*Full Table Scan*) della tabella Libri, un'operazione estremamente lenta su un catalogo di grandi dimensioni. Grazie a questi indici, le ricerche per titolo e autore diventano quasi istantanee.
- **Indice sulla tabella ValutazioniLibri:**
  - idx_valutazioni_libro_id sulla colonna libro_id.

    ```
    CREATE INDEX idx_valutazioni_libro_id ON ValutazioniLibri(libro_id);
    ```

  - Questo indice è cruciale per recuperare rapidamente tutte le valutazioni associate a un singolo libro. Viene utilizzato per calcolare la media dei voti e per visualizzare l'elenco delle recensioni degli altri utenti, operazioni comuni nella schermata di dettaglio di un libro.
- **Indici sulla tabella ConsigliLibri:**
  - idx_consigli_libro_letto_id sulla colonna libro_letto_id.

    ```
    CREATE INDEX idx_consigli_libro_letto_id ON ConsigliLibri(libro_letto_id);
    ```

  - idx_consigli_libro_consigliato_id sulla colonna libro_consigliato_id.

    ```
    CREATE INDEX idx_consigli_libro_consigliato_id ON
    ConsigliLibri(libro_consigliato_id);
    ```

- Questi indici sono critici per le prestazioni delle query che alimentano il sistema di raccomandazione.
    - L'indice idx_consigli_libro_letto_id ottimizza la ricerca dei libri consigliati a partire da un libro letto specifico
    - L'indice idx_consigli_libro_consigliato_id accelera le query di aggregazione. Permette di calcolare in modo efficiente la popolarità di un libro come raccomandazione (ad esempio, contando quante volte è stato suggerito) o di effettuare analisi sulle origini dei consigli.

## 12. Query SQL

Di seguito verranno riportati i comandi SQL utilizzati.

### 12.1 Creazione del databas

```
CREATE DATABASE "dbBR";
```

### 12.2 Creazione delle tabelle

```
CREATE TABLE UtentiRegistrati (
    user_id       SERIAL PRIMARY KEY,
    password      TEXT NOT NULL,
    nome          VARCHAR(256) NOT NULL,
    cognome       VARCHAR(256)NOT NULL,
    codice_fiscale CHAR(16) UNIQUE,
    email         VARCHAR(256) UNIQUE NOT NULL,
    );
CREATE TABLE Libri (
    id            BIGINT PRIMARY KEY,
    titolo        VARCHAR(256)NOT NULL,
    autori        VARCHAR(256),
    anno          SMALLINT,
    descrizione   VARCHAR(256),
    categorie     VARCHAR(256),
    editore       VARCHAR(256),
    prezzo        NUMERIC(10,2) CHECK (prezzo >= 0)
    );


CREATE TABLE Librerie (
    libreria_id   SERIAL PRIMARY KEY,
    user_id       INT NOT NULL REFERENCES UtentiRegistrati(user_id) ON DELETE CASCADE,
    nome_libreria VARCHAR(256), NOT NULL,
    data_creazione TIMESTAMP DEFAULT now(),
```

```
    UNIQUE (user_id, nome_libreria)
  );

 CREATE TABLE Libreria_Libro (
    libreria_id   INT NOT NULL,
    book_id       BIGINT NOT NULL,
    data_inserimento TIMESTAMP DEFAULT now(),
    PRIMARY KEY (libreria_id, libro_id),
    FOREIGN KEY (libreria_id) REFERENCES Librerie(libreria_id) ON DELETE CASCADE,
    FOREIGN KEY (libro_id)    REFERENCES Libri(id)   ON DELETE RESTRICT
  );

CREATE TABLE ValutazioniLibri (
    user_id         VARCHAR(50) NOT NULL,
    libreria_id     INT NOT NULL,
    libro_id        BIGINT NOT NULL,
    stile_score     SMALLINT NOT NULL CHECK (stile_score BETWEEN 1 AND 5),
    contenuto_score  SMALLINT NOT NULL CHECK (contenuto_score BETWEEN 1 AND 5),
    gradimento_score SMALLINT NOT NULL CHECK (gradimento_score BETWEEN 1 AND 5),
   originalita_score SMALLINT NOT NULL CHECK (originalita_score BETWEEN 1 AND 5),
   qualita_score    SMALLINT NOT NULL CHECK (qualita_score BETWEEN 1 AND 5),
   voto_complessivo SMALLINT NOT NULL CHECK (voto_complessivo BETWEEN 1 AND 5),
   stile_note       VARCHAR(256),
   contenuto_note   VARCHAR((256),
   gradimento_note  VARCHAR(256),
   originalita_note VARCHAR(256),
   qualita_note     VARCHAR(256),
   commento_finale  VARCHAR(256),
   data_valutazione TIMESTAMP DEFAULT now(),
  PRIMARY KEY (user_id, libreria_id, libro_id),
  FOREIGN KEY (user_id) REFERENCES UtentiRegistrati(user_id) ON DELETE CASCADE
  FOREIGN KEY (libreria_id, libro_id) REFERENCES Libreria_Libro(libreria_id, libro_id) ON DELETE
CASCADE
 );

CREATE TABLE ConsigliLibri (
    user_id           INT NOT NULL,
    libreria_id       INT NOT NULL,
    libro_letto_id    BIGINT NOT NULL,
    libro_consigliato_id BIGINT NOT NULL,
    commento          VARCHAR(256),
```

```
 data_consiglio      TIMESTAMP DEFAULT now(),
 PRIMARY KEY (user_id, libreria_id, libro_letto_id, libro_consigliato_id),
  FOREIGN KEY (user_id) REFERENCES UtentiRegistrati(user_id) ON DELETE CASCADE,
 FOREIGN KEY (libreria_id, libro_letto_id) REFERENCES Libreria_Libro(libreria_id, libro_id),
 FOREIGN KEY ( libro_consigliato_id) REFERENCES Libri(id)
);
```

## 12.2. Operazioni CRUD

Di seguito le query CRUD utilizzate, raggruppate per relazione. Sono fornite operazioni non espressamente richieste, ma relative a funzionalità che ci si aspetta siano richieste in futuro.

### 12.2.1. UtentiRegistrati

- **Salva un Nuovo Utente**

```
INSERT INTO UtentiRegistrati (user_id, password, nome, cognome, codice_fiscale, email)
VALUES (?, ?, ?, ?, ?, ?);
```

- **Trova un Utente**

```
SELECT user_id, password, nome, cognome, codice_fiscale, email FROM UtentiRegistrati WHERE
user_id = ? OR email = ? OR codice_fiscale = ?
```

- **Aggiorna i Dati Utente**

```
UPDATE UtentiRegistrati SET password = ?, nome = ?, cognome = ?, codice_fiscale = ?, email = ?
WHERE user_id = ?;
```

- **Elimina un Utente**

```
DELETE FROM UtentiRegistrati WHERE user_id = ?;
```

- **Seleziona Tutti gli Utenti**

```
SELECT user_id, password, nome, cognome, codice_fiscale, email FROM UtentiRegistrati;
```

### 12.2.2. Libri

- **Creazione del libro**

```
INSERT INTO Libri (titolo, autori, anno, descrizione, categorie, editore, prezzo) VALUES (?, ?, ?, ?, ?,
?, ?) RETURNING *;
```

- **Cerca libro per titolo,autori,autori e anno ,id**

```
SELECT * FROM Libri WHERE id = ?";

SELECT * FROM Libri WHERE LOWER(titolo) LIKE LOWER(?) ORDER BY titolo;

SELECT * FROM Libri WHERE LOWER(autori) LIKE LOWER(?) ORDER BY titolo;
```

SELECT * FROM Libri WHERE LOWER(autori) LIKE LOWER(?) AND anno = ? ORDER BY titolo;

SELECT * FROM Libri WHERE id = ?;

### 12.2.3. Librerie

- **Inserimento Libreria**

INSERT INTO Librerie (user_id, nome_libreria, data_creazione) VALUES (?, ?, ?) RETURNING libreria_id";

- **Selezione Libreria per ID**

SELECT * FROM Librerie WHERE libreria_id = ?;

- **Selezione Librerie per ID Utente**

SELECT * FROM Librerie WHERE user_id = ? ORDER BY nome_libreria;

- **Selezione Libreria per Utente e Nome**

SELECT * FROM Librerie WHERE user_id = ? AND nome_libreria = ?;

- **Aggiornamento Libreria**

UPDATE Librerie SET nome_libreria = ? WHERE libreria_id = ?;

- **Cancellazione Libreria**

DELETE FROM Librerie WHERE libreria_id = ?";

- **Inserimento Libro in Libreria**

INSERT INTO Libreria_Libro (libreria_id, libro_id, data_inserimento) VALUES (?, ?, ?);

- **Cancellazione Libro da Libreria**

DELETE FROM Libreria_Libro WHERE libreria_id = ? AND libro_id = ?;

- **Selezione Libri in Libreria**

SELECT libro_id FROM Libreria_Libro WHERE libreria_id = ? ORDER BY data_inserimento;

- **Verifica Libro in Libreria**

SELECT COUNT(*) FROM Libreria_Libro WHERE libreria_id = ? AND libro_id = ?;

- **Verifica Nome Libreria Esistente**

SELECT COUNT(*) FROM Librerie WHERE user_id = ? AND nome_libreria = ?";

### 12.2.4. ValutazioniLibri

- **Verifica Esistenza Valutazione Libro Utente**

SELECT 1 FROM ValutazioniLibri WHERE user_id = ? AND libro_id = ? LIMIT 1;

- **Inserisci Nuova Valutazione Libro Utente**

INSERT INTO ValutazioniLibri (user_id, libreria_id, libro_id, stile_score, contenuto_score, gradimento_score, originalita_score, qualita_score, voto_complessivo, stile_note, contenuto_note, gradimento_note, originalita_note, qualita_note, commento_finale) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);

- **Dettagli Valutazione per libro**

SELECT user_id, libro_id, stile_score, stile_note, contenuto_score, contenuto_note, gradimento_score, gradimento_note, originalita_score, originalita_note, qualita_score, qualita_note, voto_complessivo, commento_finale FROM ValutazioniLibri WHERE libro_id = ?;

- **Media Voto Complessivo Libro**

SELECT AVG(voto_complessivo) AS media FROM ValutazioniLibri WHERE libro_id = ?;

- **Numero di Valutazioni Libro**

SELECT COUNT(*) AS n FROM ValutazioniLibri WHERE libro_id = ?;

- **Media Voto Stile**

SELECT AVG(stile_score) AS media FROM ValutazioniLibri WHERE libro_id = ?;

- **Media Voto Contenuto**

SELECT AVG(contenuto_score) AS media FROM ValutazioniLibri WHERE libro_id = ?;

- **Media Voto Gradimento**

SELECT AVG(gradimento_score) AS media FROM ValutazioniLibri WHERE libro_id = ?;

- **Media Voto Originalità**

SELECT AVG(originalita_score) AS media FROM ValutazioniLibri WHERE libro_id = ?;

- **Media Voto Qualità**

SELECT AVG(qualita_score) AS media FROM ValutazioniLibri WHERE libro_id = ?;

- **Trova Libreria Utente**

SELECT libreria_id FROM Librerie WHERE user_id = ? AND nome_libreria = ?;

- **Crea Nuova Libreria**

INSERT INTO Librerie (user_id, nome_libreria) VALUES (?, ?) RETURNING libreria_id;

- **Valutazioni Complete Utente**

```
SELECT v.user_id, v.libro_id, l.titolo, l.autori, lib.nome_libreria,
       v.stile_score, v.stile_note, v.contenuto_score, v.contenuto_note,
       v.gradimento_score, v.gradimento_note, v.originalita_score, v.originalita_note,
       v.qualita_score, v.qualita_note, v.voto_complessivo, v.commento_finale
    FROM ValutazioniLibri v
    JOIN Libri l ON v.libro_id = l.id
    JOIN Librerie lib ON v.libreria_id = lib.libreria_id
    WHERE v.user_id = ?
    ORDER BY l.titolo;
```

- **Aggiorna Valutazione Libro Utente**

```
UPDATE ValutazioniLibri SET
       stile_score = ?, stile_note = ?,
       contenuto_score = ?, contenuto_note = ?,
       gradimento_score = ?, gradimento_note = ?,
       originalita_score = ?, originalita_note = ?,
       qualita_score = ?, qualita_note = ?,
       voto_complessivo = ?, commento_finale = ?
    WHERE user_id = ? AND libro_id = ?;
```

- **Elimina Valutazione Libro Utente**

```
DELETE FROM ValutazioniLibri WHERE user_id = ? AND libro_id = ?;
```

### 12.2.6. ConsigliLibri

- **Inserimento**

```
INSERT INTO ConsigliLibri (user_id, libreria_id, libro_letto_id, libro_consigliato_id, commento)
VALUES (?, ?, ?, ?, ?)";

SELECT COUNT(*) FROM ConsigliLibri WHERE user_id = ? AND libro_letto_id = ?";
```

- **Trovare i libri consigliati**

```
 SELECT l.id, l.titolo, l.autori, l.anno, l.descrizione, l.categorie, l.editore, l.prezzo
 FROM ConsigliLibri cl
JOIN Libri l ON cl.libro_consigliato_id = l.id
```

WHERE cl.libreria_id = ? AND cl.libro_letto_id = ?;

- **Trovare i libri consigliati con il conteggio delle raccomandazioni, per una specifica libreria**.

  SELECT l.id, l.titolo, l.autori, l.anno, l.descrizione, l.categorie, l.editore, l.prezzo,
  COUNT(cl.libro_consigliato_id) as conteggio

  FROM ConsigliLibri cl

  JOIN Libri l ON cl.libro_consigliato_id = l.id

  WHERE cl.libreria_id = ? AND cl.libro_letto_id = ?

  GROUP BY l.id, l.titolo, l.autori, l.anno, l.descrizione, l.categorie, l.editore, l.prezzo

  ORDER BY conteggio DESC;

- **Trovare i libri consigliati con il conteggio delle raccomandazioni, aggregando da tutte le librerie.**

SELECT l.id, l.titolo, l.autori, l.anno, l.descrizione, l.categorie, l.editore, l.prezzo,
COUNT(cl.libro_consigliato_id) as conteggio

FROM ConsigliLibri cl

JOIN Libri l ON cl.libro_consigliato_id = l.id

WHERE cl.libro_letto_id = ?

GROUP BY l.id, l.titolo, l.autori, l.anno, l.descrizione, l.categorie, l.editore, l.prezzo

ORDER BY conteggio DESC;

- **Trovare tutti i consigli (in forma base) dati da un utente.**

SELECT user_id, libreria_id, libro_letto_id, libro_consigliato_id, commento, data_consiglio

FROM ConsigliLibri

WHERE user_id = ?

ORDER BY data_consiglio DESC;

- **Aggiornare il commento di un consiglio esistente**.

UPDATE ConsigliLibri

SET commento = ?

WHERE user_id = ? AND libreria_id = ? AND libro_letto_id = ? AND libro_consigliato_id = ?;

- **Eliminare un Consiglio Specifico.**

```
DELETE FROM ConsigliLibri
WHERE user_id = ? AND libreria_id = ? AND libro_letto_id = ? AND libro_consigliato_id = ?;
```

- **Trovare tutti i consigli di un utente, arricchiti con dettagli**.

```
SELECT cl.user_id, cl.libreria_id, l.nome_libreria as nome_libreria,
 cl.libro_letto_id, ll.titolo as titolo_letto, ll.autori as autore_letto,
 cl.libro_consigliato_id, lc.titolo as titolo_consigliato, lc.autori as autore_consigliato,
 cl.commento, cl.data_consiglio
FROM ConsigliLibri cl
JOIN Librerie l ON cl.libreria_id = l.libreria_id
JOIN Libri ll ON cl.libro_letto_id = ll.id
JOIN Libri lc ON cl.libro_consigliato_id = lc.id
WHERE cl.user_id = ?
ORDER BY l.nome_libreria, ll.titolo, lc.titolo;
```