Book Recommender

Manuale Database

Alessandro Grassi, Luca Giorgio Rotter, Aleksandar Kastratovic, Davide Bilora

Indice

Introduzione

“Book Recommender” è un software per la valutazione e raccomandazione di libri, in grado di permettere agli utenti registrati di inserire recensioni e a tutti gli utenti di consultare le valutazioni e ricevere consigli di lettura

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,
- 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

Raccolta e 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 libreria "Book Recommender":

- **Libri**: sono già presenti nel database, e sono circa 100.000. Vengono usati dagli utenti per essere principalmente consultati da qualsiasi tipo di utente, inseriti nelle librerie e valutati da parte degli utenti registrati. I libri già presenti nel database non possono essere modificati. Gli attributi di questi sono:
    - Titolo,
    - Autore,
    - Anno di pubblicazione
    - editore,
    - Genere

  I libri possono essere valutati riguardo la loro: edizione, stile, originalità, contenuto e gradevolezza. Per ognuna di queste valutazione si può inserire una valutazione(da 1 a 5) e un commento. I libri a cui si inserisce un consiglio devono essere necessariamente essere appartenenti ad una propria libreria. Si possono consigliare fino ad un massimo di 3 libri per ciascun libro.

- **Utenti**: gli utenti sono coloro che possono accedere al software per consultare i libri e se si registrano possono anche creare delle proprie librerie e inserire dei libri in esse. Inoltre gli utenti registrati possono inserire valutazione ai libri e dei libri consigliati riguardanti un libro specifico. Gli attributi indicati per gli utenti registrati sono:
    - Nome
    - Cognome
    - email
    - Codice Fiscale
    - password

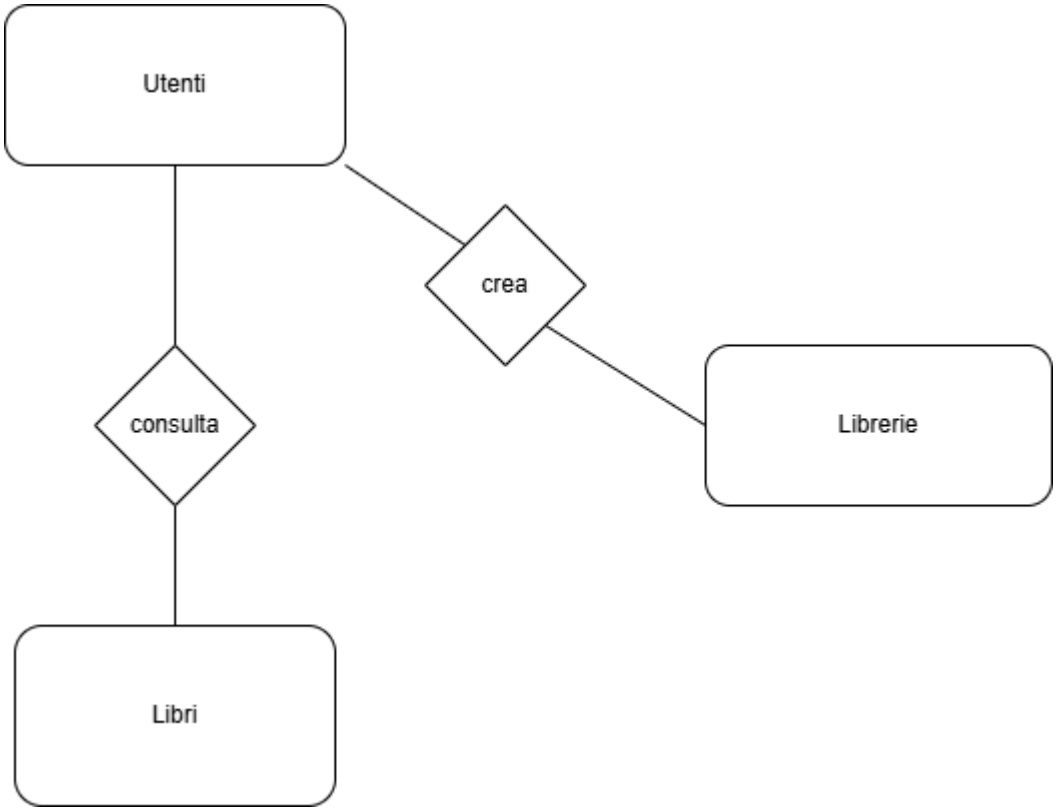
- **Librerie**: possono essere create solo da utenti registrati alla piattaforma e ad esse gli si può attribuire un nome. All'interno di esse possono essere inseriti un numero indefinito di libri e gli utenti possono aggiungere e togliere libri dalle librerie a loro piacimento.

E' richiesto nella homepage che l'utente quando apra l'applicazione può visualizzare alcuni libri presi in maniera casuale dal database; nel caso in cui dovesse scorrere questa lista di libri è necessario che i libri già visualizzati nell'interfaccia non vengano riproposti ulteriormente quindi è necessario dover memorizzare anche questi in maniere temporanea. La progettazione del database deve quindi assicurare che il sistema finale sia scalabile, al fine di gestire l'aumento del volume di dati e degli accessi simultanei garantendo prestazioni ottimali e tempi di risposta rapidi.

In sintesi, il database dell’applicazione Book Recommender dovrà supportare un’ampia rete di utenti, gestendo efficacemente sia le operazioni di lettura che di scrittura per offrire un servizio affidabile e scalabile

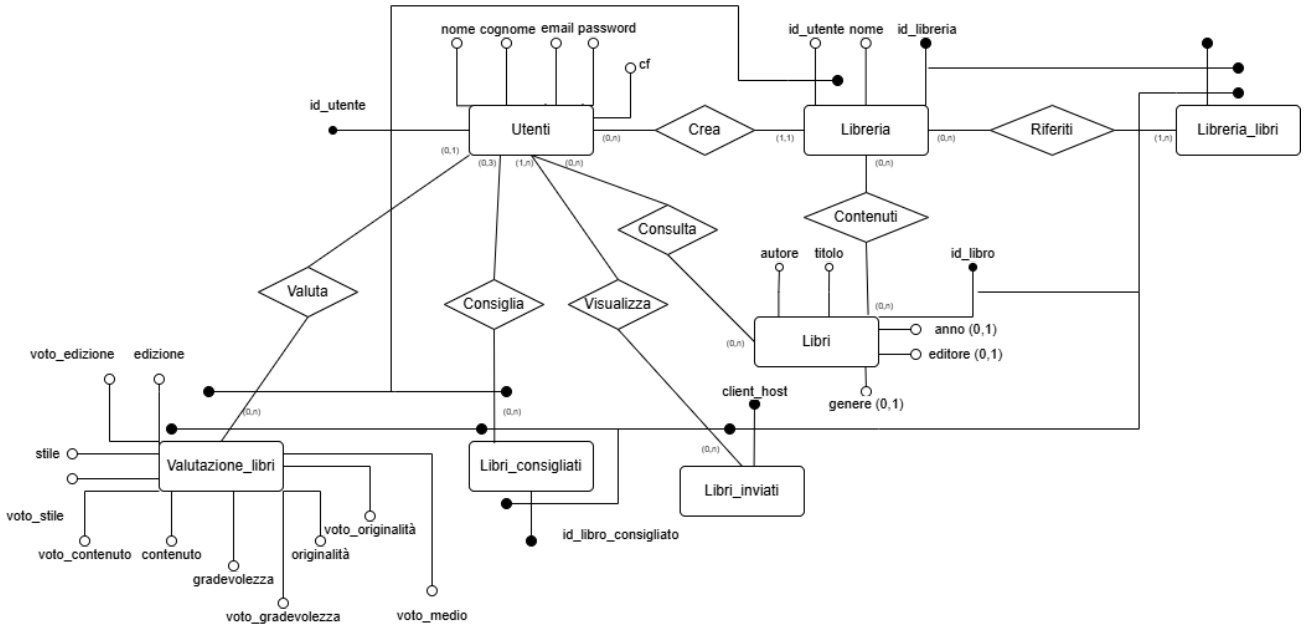
Schema scheletro

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



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:



Per quanto riguarda gli identificatori, si è convenuto di aggiungere un ID intero (auto-incrementante nel database finale) per le entità relative agli utenti e le librerie; mentre i libri al momento del loro inserimento nel database gli è stato attribuito in ID intero.
Inoltre sono stati dati gli attributi UNIQUE agli attributi cf e email dell'utente.
Infine gli altri vincoli non rappresentabili nello schema concettuale sono:

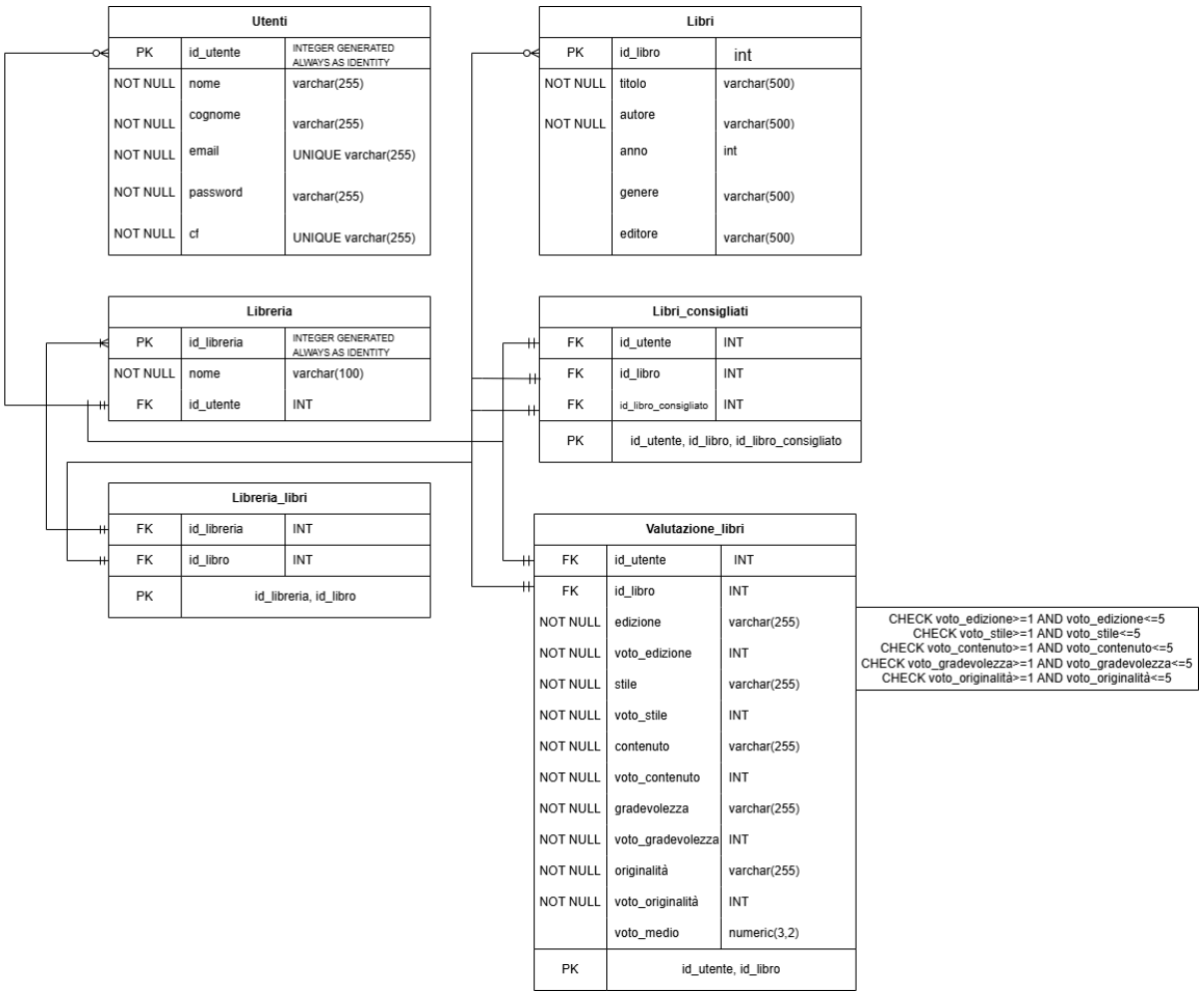
- un utente può valutare un libro solo se esso è presente in una delle sue librerie
- un utente può consigliare un libro solo se esso è presente in una delle sue librerie

Normalizzazione

Per quanto riguarda la normalizzazione e la ristrutturazione dello schema concettuale è stato ritenuto che non fosse opportuno in quanto non presenti costrutti non rappresentabili cioè: attributi multi-valore e gerarchie.

Schema logico

Lo schema logico finale prevede la traduzione dello schema concettuale ristrutturato nelle effettive tabelle della base di dati.
Le associazioni uno-a-molti (molti-a-uno) sono state opportunamente tradotte in chiavi esterne come da prassi. Non erano presenti associazioni uno-a-uno. L'unica associazione molti-a-molti sarebbe stata tra utenti e librerie ma si è opportunamente creata una tabella di collegamento(libreria) la cui contiene l'id utente associato al propria l'libreria e la tabella libreria_libri contiene il collegamento tra id_libreria e id_libro:



Per gli identificatori aggiuntivi sono stati impiegati i tipi di dato specifici di PostgreSQL ovvero INTEGER GENERATED ALWAYS AS IDENTITY, che corrispondono ad indici auto-incrementali. Per gli altri attributi sono stati usati principalmente INT per numeri e VARCHAR per stringhe di dimensione adatta all'attributo. Solo per il valore voto_medio è stato scelto il tipo NUMERIC in quanto permette di inserire un numero con la virgola.

Query SQL

Di seguito verranno riportati i comandi SQL utilizzati:
CREAZIONE TABELLE:

Tabella libri:

```
create table Libri (
    id_libro int primary key,
    titolo varchar(500) not null,
    autore varchar(500) not null,
    anno int,
    genere varchar(500),
    editore varchar(500)
);
```

Tabella utenti:

```
create table Utenti (
    id_utente INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    nome varchar(255) not null,
    cognome varchar(255) not null,
    email varchar(255) unique not null,
    password varchar(255) not null,
    cf varchar(16) unique not null
);
```

Tabella Libreria:

```
CREATE TABLE libreria (
    id_libreria INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    id_utente INT not null references Utenti(id_utente) on delete cascade on update cascade,
    nome VARCHAR(100) NOT NULL
);
```

Tabella Libreria_libri:

```
CREATE TABLE libreria_libri (
    id_libreria INT NOT NULL  REFERENCES libreria(id_libreria) on delete cascade on
update cascade,
    id_libro INT NOT NULL REFERENCES libri(id_libro) on delete cascade on update
cascade,
    PRIMARY KEY (id_libreria, id_libro)
);
```

Tabella Libri_consigliati:

```
create table Libri_Consigliati (
    id_libro int references Libri(id_libro) on delete cascade on update cascade,
    id_utente int DEFAULT 1 references Utenti(id_utente) on delete set default on update
cascade,
    id_libro_consigliato int references Libri(id_libro),
    primary key (id_libro, id_utente, id_libro_consigliato)
);
```

Tabella Valutazione_libri:

```
create table Valutazioni_Libri (
    id_libro int references Libri(id_libro) on delete cascade on update cascade,
    id_utente int DEFAULT 1 references Utenti(id_utente) on delete set default on update
cascade,
    primary key (id_libro, id_utente),
    edizione varchar(255) not null,
    voto_edizione int check(voto_edizione >= 1 and voto_edizione <= 5) not null,
    stile varchar(255) not null,
    voto_stile int check(voto_stile >= 1 and voto_stile <= 5) not null,
    contenuto varchar(255) not null,
```

```
    voto_contenuto int check(voto_contenuto >= 1 and voto_contenuto <= 5) not null,
    gradevolezza varchar(255) not null,
    voto_gradevolezza int check(voto_gradevolezza >= 1 and voto_gradevolezza <= 5) not
null,
    originalita varchar(255) not null,
    voto_originalita int check(voto_ordiginalita >= 1 and voto_ordiginalita <= 5) not null,
    voto_medio numeric(3,2)

);
```

Tabella Libri_inviati:

```
CREATE TABLE libri_inviati (
    client_host VARCHAR(255) NOT NULL,
    id_libro INT NOT NULL,
    PRIMARY KEY (client_host, id_libro),
    FOREIGN KEY (id_libro) REFERENCES libri(id_libro)
);
```

Operazione CRUD

Di seguito le query CRUD utilizzate, raggruppate per relazione:

Utenti

Inserimento:

```
INSERT INTO utenti (nome, cognome, cf, email, password) VALUES (?, ?, ?, ?, ?)
```

Ricerca tramite email e password:

```
SELECT id_utente FROM utenti WHERE email = ? AND password = ?
```

Ricerca tramite id_utente:

```
SELECT * FROM utenti WHERE id_utente = ?
```

Libri

Inserimento dei libri inviati:

```
INSERT INTO libri_inviati (client_host, id_libro) VALUES (?, ?) ON CONFLICT DO
NOTHING
```

Ricerca dei libri inviati tramite id_libro e riferimento clientHost:

```
 SELECT l.id_libro, l.titolo, l.autore, l.genere, l.editore, l.anno
        FROM libri l
        LEFT JOIN libri_inviati li
            ON l.id_libro = li.id_libro AND li.client_host = ?
        WHERE li.id_libro IS NULL
        ORDER BY l.id_libro
        LIMIT ?
```

Eliminazione dei libri inviati:
DELETE FROM libri_inviati WHERE client_host = ?

Ricerca di un libro tramite id_libro:
SELECT * FROM libri WHERE id_libro = ?

Ricerca di libri tramite titolo:
SELECT * FROM libri WHERE titolo ILIKE ?

Ricerca di libri tramite autore e anno:
SELECT * FROM libri WHERE autore ILIKE ? AND anno = ?

Ricerca di libri tramite autore:
SELECT * FROM libri WHERE autore ILIKE ?

Libreria
Creazione libreria:
INSERT into libreria(id_utente, nome) VALUES (?, ?)

Elimazione libreria:
DELETE FROM libreria WHERE id_libreria = ?

Inserimento di un libro nella libreria:
INSERT into libreria_libri(id_libro, id_libreria) VALUES (?, ?)

Elimazione di un libro da una libreria tramite id_libro:
DELETE FROM libreria_libri WHERE id_libro = ? AND id_libreria = ?

Ricerca di una libreria tramite id_libreria:
SELECT * FROM libreria L LEFT JOIN libreria_libri M on L.id_libreria = M.id_libreria
WHERE L.id_libreria = ?

Ricerca di librerie tramite id_utente:
SELECT id_libreria FROM libreria WHERE id_utente = ?

Libri consigliati
Creazione di un libro consigliato:
INSERT into Libri_consigliati (id_utente, id_libro, id_libro_consigliato) VALUES (?, ?, ?)

Ricerca numero di libri consigliati riguardati un libro tramite id_libro e id_utente:
SELECT count(*) FROM Libri_consigliati WHERE id_utente = ? AND id_libro = ?

Ricerca di libri consigliati tramite id_libro:
"SELECT * FROM libri_consigliati WHERE id_libro = ?

Ricerca libri consigliati tramite id_libro e id_utente:
SELECT * FROM libri_consigliati WHERE id_libro = ? AND id_utente = ?

Eliminazione di un libro consigliato:
DELETE FROM Libri_consigliati WHERE id_utente = ? AND id_libro = ? AND id_libro_consigliato = ?

Valutazione libri
Creazione valutazione di un libro:
INSERT into Valutazioni_Libri (id_utente, id_libro, edizione, voto_edizione, stile, voto_stile, contenuto, voto_contenuto, gradevolezza, voto_gradevolezza, originalita, voto_originalita, voto_medio) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? , ?)

Ricerca valutazioni tramite id_libro:
SELECT * FROM Valutazioni_Libri WHERE id_libro = ?

Ricerca valutazione tramtie id_libro e id_utente:
SELECT * FROM Valutazioni_Libri WHERE id_utente = ? AND id_libro = ?

Ricerca voto medio tramite id_libro:
SELECT AVG(voto_medio) FROM valutazioni WHERE id_libro = ?

Strumenti utilizzati

Al fine di realizzare questo manuale sono stati utilizzati:
draw.io per la progettazione dello schema sceheletro, diagramma E-R e schema logico

Contatti

Alessandro Grassi, Project Manager: agrassi6@studenti.uninsubria.it