

0280114

Remoli Simone

Basi di Dati



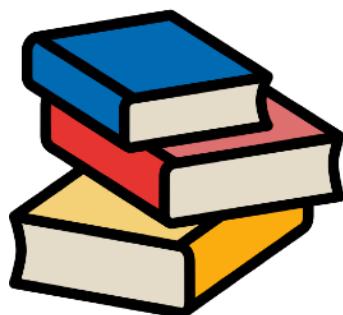
Basi di Dati

Progetto A.A. 2024/2025

Sistema Informativo di una Biblioteca

0280114

Simone Remoli



Descrizione del Minimondo

1. Una biblioteca mette a disposizione un servizio di prenotazione libri.
2. Si vuole realizzare un sistema **informativo** di gestione per tale biblioteca,
3. caratterizzata da indirizzo, numero di telefono, nome del responsabile ed orario di
4. apertura settimanale.
5. Il sistema gestisce libri, le cui copie sono disponibili in un certo numero.
6. Ciascun **libro** è associato allo stato “**in prestito**”/“disponibile”.
7. Gli utenti della biblioteca possono registrarsi fornendo tutte le loro informazioni
8. anagrafiche ed un numero arbitrario di contatti (**telefono**, cellulare, email),
9. specificando qual è il **mezzo** di **comunicazione** preferito con cui vogliono essere
10. contattati.
- 11.
12. All’atto di effettuare un prestito, i bibliotecari possono recuperare la
13. disponibilità delle copie del libro.
- 14.
15. Se il **libro** è disponibile, il sistema restituisce lo scaffale ed il ripiano in cui può
16. essere prelevato il **libro**. Poiché la biblioteca è convenzionata con un circuito di
17. altre biblioteche, se non ne è disponibile alcuna copia, il bibliotecario può **verificare**
18. in quali altre biblioteche del circuito è presente tale libro ed effettuare una richiesta
19. di trasferimento. In maniera analoga, le altre biblioteche possono effettuare
20. richieste di trasferimento. In questo caso, il **libro** viene segnalato come “prestato ad
21. altra biblioteca” e viene tenuto traccia di qual è la biblioteca di destinazione.
22. La gestione dei trasferimenti tra biblioteche viene effettuata telefonicamente tra i
23. bibliotecari.
24. All’atto della **consegna** della copia, l’utente del servizio può chiedere di
25. trattenere in **consultazione** il libro per 1, 2, o 3 mesi.
- 26.
27. I bibliotecari hanno la possibilità di generare un report indicante quali libri in
28. prestito non sono ancora stati **restituiti** e le informazioni dell’utente che
29. possiede attualmente la copia del libro.
30. I **recapiti** consentono al bibliotecario di mettersi in contatto per sollecitare la
31. **restituzione**.
32. Un libro che non è stato prestato nei passati 10 anni viene dismesso dalla
33. biblioteca. Tale operazione viene effettuata dagli amministratori del circuito. Il
34. record ad esso associato non viene eliminato, ma questo non potrà più essere
35. prestato agli utenti.
36. I prestiti in corso restano validi fino alla riconsegna.

Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili.

| Linea | Termine | Nuovo termine | Motivo correzione |
|----------|--------------------|--|--|
| 2 | Informativo | Informatico | Ciò che si andrà a realizzare è un sistema informatico, ovvero una componente tecnologica di supporto al sistema informativo. |
| 6 | In prestito | In prestito utente/In prestito biblioteca | I termini impiegati per rappresentare lo stato del libro risultano poco univoci, in quanto non è esplicitato se la dicitura "in prestito" comprenda anche le copie temporaneamente trasferite ad altre biblioteche appartenenti al circuito convenzionato. Si ritiene pertanto opportuno introdurre una distinzione semantica più rigorosa tra il prestito interno, ossia il prestito della copia a un utente registrato, e il prestito esterno, ovvero il trasferimento del volume a un'altra biblioteca. |

| Linea | Termine | Nuovo termine | Motivo correzione |
|--------------|-----------------|----------------------|--|
| 8 | Telefono | Numero | <p>L'indicazione dei mezzi di contatto include sia <i>telefono</i> sia <i>cellulare</i>, ma tale distinzione è ridondante, in quanto il termine <i>telefono</i> è generico e, nel linguaggio attuale, è comunemente associato proprio al dispositivo mobile.</p> <p>Per evitare ambiguità e semplificare la struttura del dato, si propone di eliminare il termine <i>telefono</i>, mantenendo unicamente <i>cellulare</i> come riferimento per il contatto telefonico, la gestione della email invece rimane invariata.</p> |
| 9 | Mezzo | Metodo | <p>Il termine <i>mezzo di comunicazione</i> può risultare ambiguo o eccessivamente generico, in quanto è spesso associato ai mass media (es. stampa, radio, TV). Nel contesto di un sistema informatico per la gestione di utenti e contatti, è più appropriato il termine <i>metodo di comunicazione</i>, che si riferisce in modo più preciso alla modalità scelta dall'utente per essere contattato individualmente (es. email, cellulare).</p> |

| Linea | Termine | Nuovo termine | Motivo correzione |
|---------------|---------------|---------------|--|
| 9 | Comunicazione | Contatto | Nella pratica, il sistema non gestisce il processo di comunicazione nel senso pieno del termine (scambio bidirezionale di messaggi), ma solo le modalità di contatto con l'utente. Pertanto, il termine <i>comunicazione</i> può generare confusione, facendo pensare a strumenti di dialogo o messaggistica, mentre l'intenzione reale è registrare il <i>mezzo di contatto preferito</i> . |
| 6, 15, 16, 20 | Libro | Copia | Il termine <i>libro</i> viene spesso utilizzato nel testo come entità prestata o consultata, ma da un punto di vista logico e fisico, è la <i>copia del libro</i> l'unità concreta che può essere prestata, trasferita o consultata. L'uso indistinto dei termini <i>libro</i> e <i>copia</i> può generare ambiguità nella definizione delle entità e delle relazioni del modello dati, inducendo a confondere l'opera editoriale astratta con le sue istanze fisiche. |
| 17 | Verificare | Consultare | In questo caso, l'intenzione è quella di descrivere un'interazione con il sistema informatico, per cui termini più precisi come <i>interrogare</i> , <i>consultare</i> o <i>ricercare</i> risultano maggiormente appropriati. |

| Linea | Termine | Nuovo termine | Motivo correzione |
|--------------|----------------------|---------------------------|--|
| 24 | Consegna | Ritiro | Il termine è ambiguo: potrebbe sembrare che si tratti della restituzione, ma dal contesto è la consegna iniziale all'utente. |
| 25 | Consultazione | Prestito | Poiché nel testo si fa esplicitamente riferimento alla possibilità per l'utente di trattenere il libro per uno o più mesi, è più appropriato il termine <i>prestito</i> , che chiarisce la natura dell'operazione come ritiro del volume per un periodo definito. |
| 28 | Restituiti | Riconsegnati | Il termine <i>restituiti</i> presenta un grado di ambiguità in quanto può riferirsi a un'azione generica di restituzione, senza specificare il soggetto destinatario dell'operazione. Nel contesto del sistema bibliotecario descritto, è invece opportuno utilizzare il termine <i>riconsegnati</i> , che sottolinea in modo più preciso l'atto di riconsegna della copia da parte dell'utente alla biblioteca. |
| 30 | Recapiti | Metodi di Contatto | Il termine <i>recapiti</i> è generico e poco specifico, in quanto può riferirsi a qualunque tipo di informazione per contattare una persona. |

| Linea | Termine | Nuovo termine | Motivo correzione |
|-------|--------------|---------------|---|
| 31 | Restituzione | Riconsegna | <p>Il termine <i>restituzione</i>, seppur comunemente utilizzato in ambito bibliotecario, presenta un certo grado di ambiguità, in quanto non specifica il destinatario dell'azione. Potrebbe riferirsi alla riconsegna della copia alla biblioteca presso cui è stata prelevata, oppure, in caso di prestito interbibliotecario, alla riconsegna della copia alla biblioteca proprietaria.</p> |

Specifiche Disambiguata.

Una biblioteca mette a disposizione un servizio di prenotazione libri.
Si vuole realizzare un sistema informatico di gestione per tale biblioteca.

Il sistema gestisce libri, le cui copie sono disponibili in un certo numero.
Ciascuna copia è associata allo stato “in prestito utente”/“in prestito biblioteca”/
“disponibile”.

Gli utenti della biblioteca possono procedere alla registrazione fornendo le proprie informazioni anagrafiche unitamente a uno specifico contatto (numero di telefono o indirizzo email), indicando il metodo di contatto preferito attraverso il quale desiderano essere contattati.

All’atto di effettuare un prestito, i bibliotecari possono recuperare la disponibilità delle copie del libro.

Se la copia è disponibile, il sistema restituisce l’indicazione dello scaffale e del ripiano presso cui la copia può essere prelevata e, al momento del ritiro, la copia viene contrassegnata con lo stato “in prestito utente”. Poiché la biblioteca è convenzionata con un circuito di altre biblioteche, se non ne è disponibile alcuna copia, il bibliotecario può consultare in quali altre biblioteche del circuito è presente tale libro ed effettuare una richiesta di trasferimento della copia.

In maniera analoga, le altre biblioteche possono effettuare richieste di trasferimento. In questo caso, la copia locale viene segnalata come “in prestito biblioteca” e viene tenuto traccia di qual è la biblioteca di destinazione.

All’atto del ritiro della copia, l’utente del servizio può chiedere di trattenere in prestito il libro per 1, 2, o 3 mesi.

I bibliotecari hanno la possibilità di generare un report indicante quali libri in prestito non sono ancora stati riconsegnati e le informazioni dell’utente che possiede attualmente la copia del libro.

I metodi di contatto consentono al bibliotecario di contattare l’utente per sollecitare la riconsegna.

Un libro che non è stato prestato nei passati 10 anni viene dismesso dalla biblioteca. Tale operazione viene effettuata dagli amministratori del circuito. Il record ad esso associato non viene eliminato, ma questo non potrà più essere prestato agli utenti.

I prestiti in corso restano validi fino alla riconsegna.

Glossario dei Termini.

| Termine | Descrizione | Sinonimi | Collegamenti |
|---------------------------|--|----------------------|-------------------------|
| Copia | Unità fisica di un libro, soggetta a prestito e trasferimento. | Esemplare | Stato, Prestito. |
| Stato | Stato operativo di una copia: disponibile, in prestito utente, in prestito biblioteca. | Disponibilità | Copia, Prestito. |
| Utente | Persona regolarmente registrata nel sistema bibliotecario, abilitata al prestito di una sola copia per volta, fino alla riconsegna della medesima. | Lettore | Prestito. |
| Bibliotecario | Operatore autorizzato all'utilizzo delle funzionalità gestionali del sistema, responsabile delle operazioni di prestito, restituzione, trasferimento delle copie e comunicazione con gli utenti e con le altre biblioteche del circuito. | Addetto | Prestito. |
| Prestito | Operazione tramite cui una copia viene assegnata temporaneamente a un utente o ad un'altra biblioteca. | Richiesta | Copia, Stato. |
| Metodo di Contatto | Modalità con cui l'utente preferisce essere contattato (telefono o email). | Modalità di contatto | Utente. |

| Termine | Descrizione | Sinonimi | Collegamenti |
|-----------------------|--|------------------|------------------------------|
| Amministratore | Figura incaricata della gestione centralizzata delle biblioteche appartenenti al circuito, con compiti di supervisione, coordinamento e attuazione di operazioni speciali, quali la dismissione delle copie non più oggetto di prestito. | Gestore Circuito | Copia, Bibliotecario. |

Raggruppamento dei Requisiti in Insiemi Omogenei.

Frasi di carattere generale

Una biblioteca mette a disposizione un servizio di prenotazione libri.
Si vuole realizzare un sistema informatico di gestione per tale biblioteca.

Frasi relative alle copie

Il sistema gestisce libri, le cui copie sono disponibili in un certo numero.

Frasi relative allo stato

Ciascuna copia è associata allo stato “in prestito utente”/“in prestito biblioteca”/
“disponibile”.

Frasi relative agli utenti

Gli utenti della biblioteca possono procedere alla registrazione fornendo le proprie informazioni anagrafiche unitamente a uno specifico contatto (numero di telefono o indirizzo email), indicando il metodo di contatto preferito attraverso il quale desiderano essere contattati.

Frasi relative ai bibliotecari

All’atto di effettuare un prestito, i bibliotecari possono recuperare la disponibilità delle copie del libro.

Poiché la biblioteca è convenzionata con un circuito di altre biblioteche, se non ne è disponibile alcuna copia, il bibliotecario può consultare in quali altre biblioteche del circuito è presente tale libro ed effettuare una richiesta di trasferimento della copia. I bibliotecari hanno la possibilità di generare un report indicante quali libri in prestito non sono ancora stati riconsegnati e le informazioni dell’utente che possiede attualmente la copia del libro.

Frasi relative al prestito

Se la copia è disponibile, il sistema restituisce l’indicazione dello scaffale e del ripiano presso cui la copia può essere prelevata e, al momento del ritiro, la copia viene contrassegnata con lo stato “*in prestito utente*”.

In maniera analoga, le altre biblioteche possono effettuare richieste di trasferimento. In questo caso, la copia locale viene segnalata come “*in prestito biblioteca*” e viene tenuto traccia di qual è la biblioteca di destinazione.

All’atto del ritiro della copia, l’utente del servizio può chiedere di trattenere in prestito il libro per 1, 2, o 3 mesi.

I prestiti in corso restano validi fino alla riconsegna.

Frasi relative ai metodi di contatto

I metodi di contatto consentono al bibliotecario di contattare l’utente per sollecitare la riconsegna.

Frasi relative agli amministratori

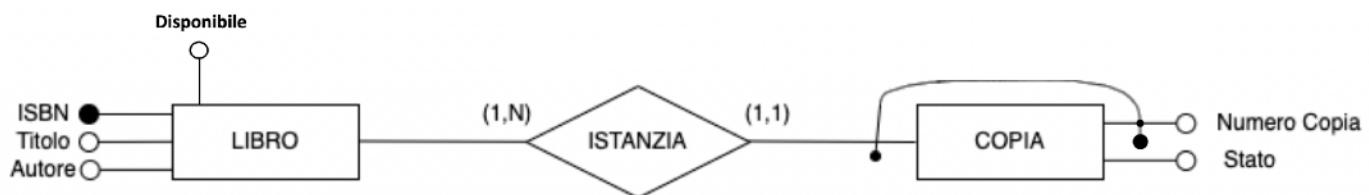
Un libro che non è stato prestato nei passati 10 anni viene dismesso dalla biblioteca. Tale operazione viene effettuata dagli amministratori del circuito. Il record ad esso associato non viene eliminato, ma questo non potrà più essere prestato agli utenti.

Progettazione Concettuale.

Costruzione dello schema E-R.

La costruzione del modello E-R è stata condotta adottando la strategia **inside-out**, ovvero un approccio che prevede l'individuazione iniziale di un nucleo ristretto di concetti fondamentali, dai quali si procede successivamente all'estensione progressiva del modello, secondo un criterio espansivo e graduale.

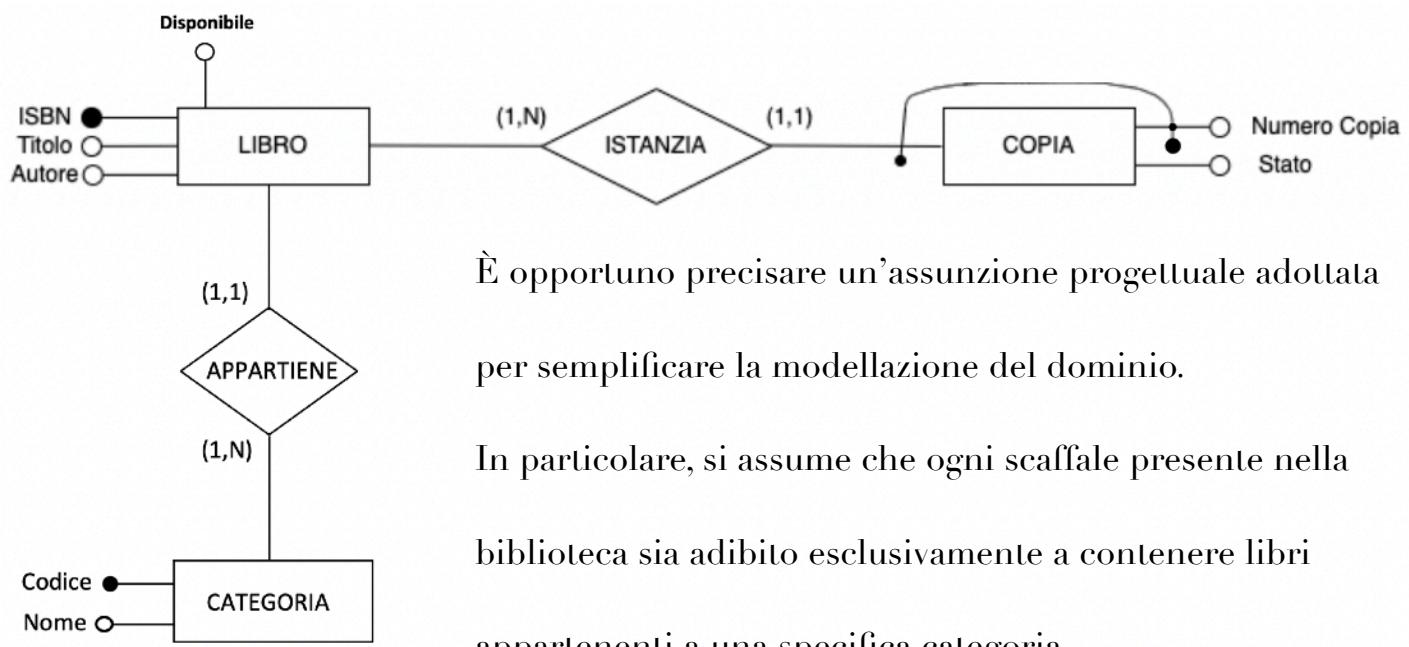
In fase iniziale, si è ritenuto opportuno rappresentare il concetto centrale ai fini della progettazione, ovvero la distinzione tra l'opera bibliografica astratta (*Libro*) e le sue manifestazioni fisiche (*Copia*). Tale scelta ha condotto all'introduzione di due entità distinte: *Libro* e *Copia*, collegate mediante la relazione *Istanzia*, che esprime il fatto che ciascuna copia rappresenta un'istanza materiale del corrispondente libro.



Nel contesto del modello concettuale, la relazione "ISTANZIA" lega le entità LIBRO e COPIA secondo una dipendenza esistenziale: una copia non può esistere senza il libro fisico da cui è istanziata. Pertanto, l'entità COPIA è identificata in modo univoco attraverso un identificatore esterno derivante dall'entità LIBRO, ovvero l'ISBN, e da un attributo proprio, il numero di copia.

Ragionando sulla natura del minimondo di riferimento, si osserva che ogni libro appartiene a una ben precisa categoria tematica o disciplinare. È importante sottolineare che non è la copia a essere classificata, in quanto la copia rappresenta una istanza fisica del libro stesso e ne eredita le caratteristiche concettuali.

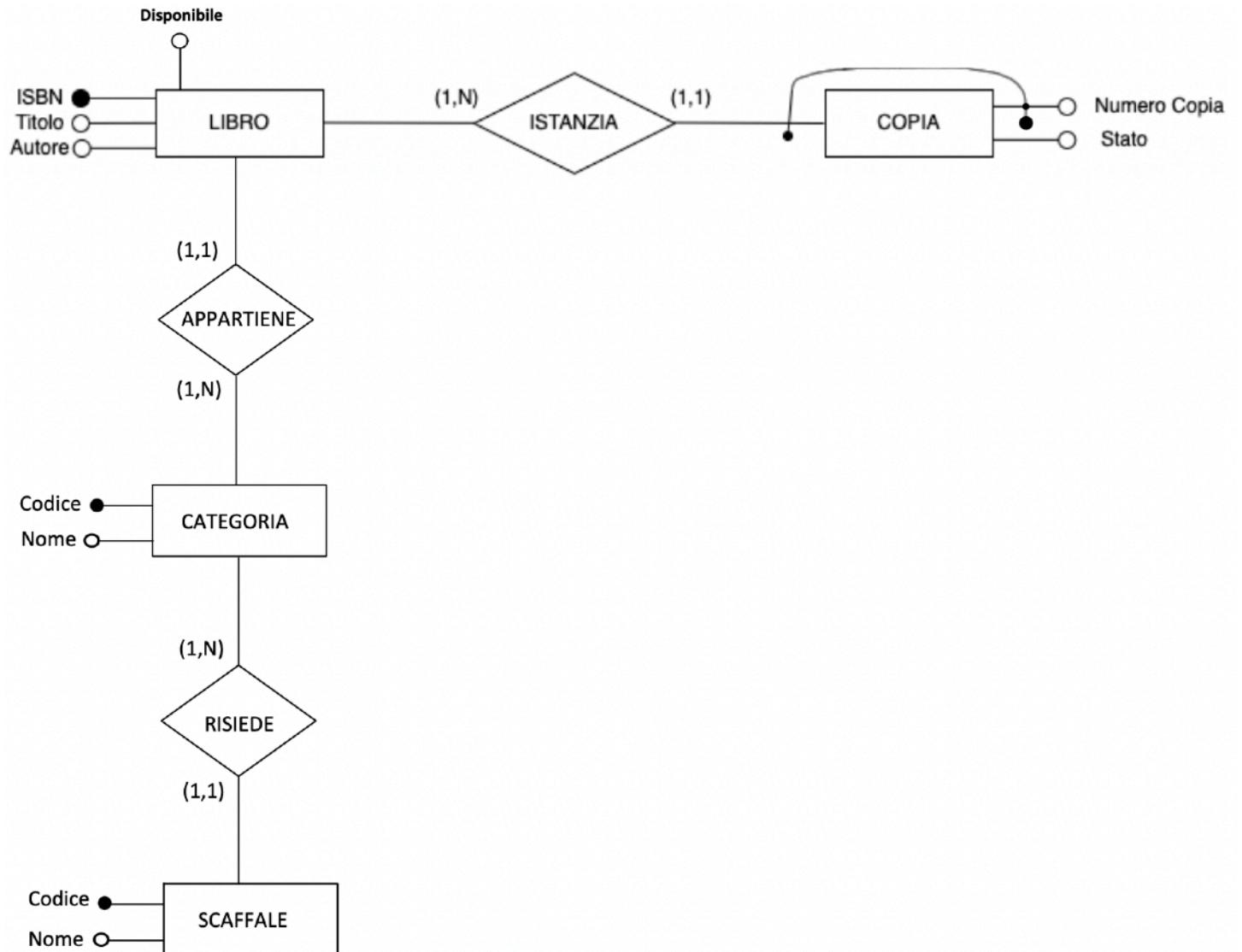
Per tale motivo, si introduce una relazione tra l'entità LIBRO e l'entità CATEGORIA, al fine di modellare l'appartenenza di ciascun libro a una determinata categoria. Tale relazione riflette una classificazione semantica, dove la categoria rappresenta un'informazione concettuale associata all'opera nel suo complesso, e non ai suoi singoli esemplari materiali.



In base a tale assunzione, si introduce la relazione "RISIEDE" tra le entità CATEGORIA e SCAFFALE, la quale esprime che una categoria “risiede” in un determinato scaffale. Questa relazione riflette una mappatura logistica tra classificazione concettuale (categoria) e collocazione fisica (scaffale) all'interno della biblioteca.

Ogni scaffale contiene esclusivamente copie di libri appartenenti alla medesima categoria.

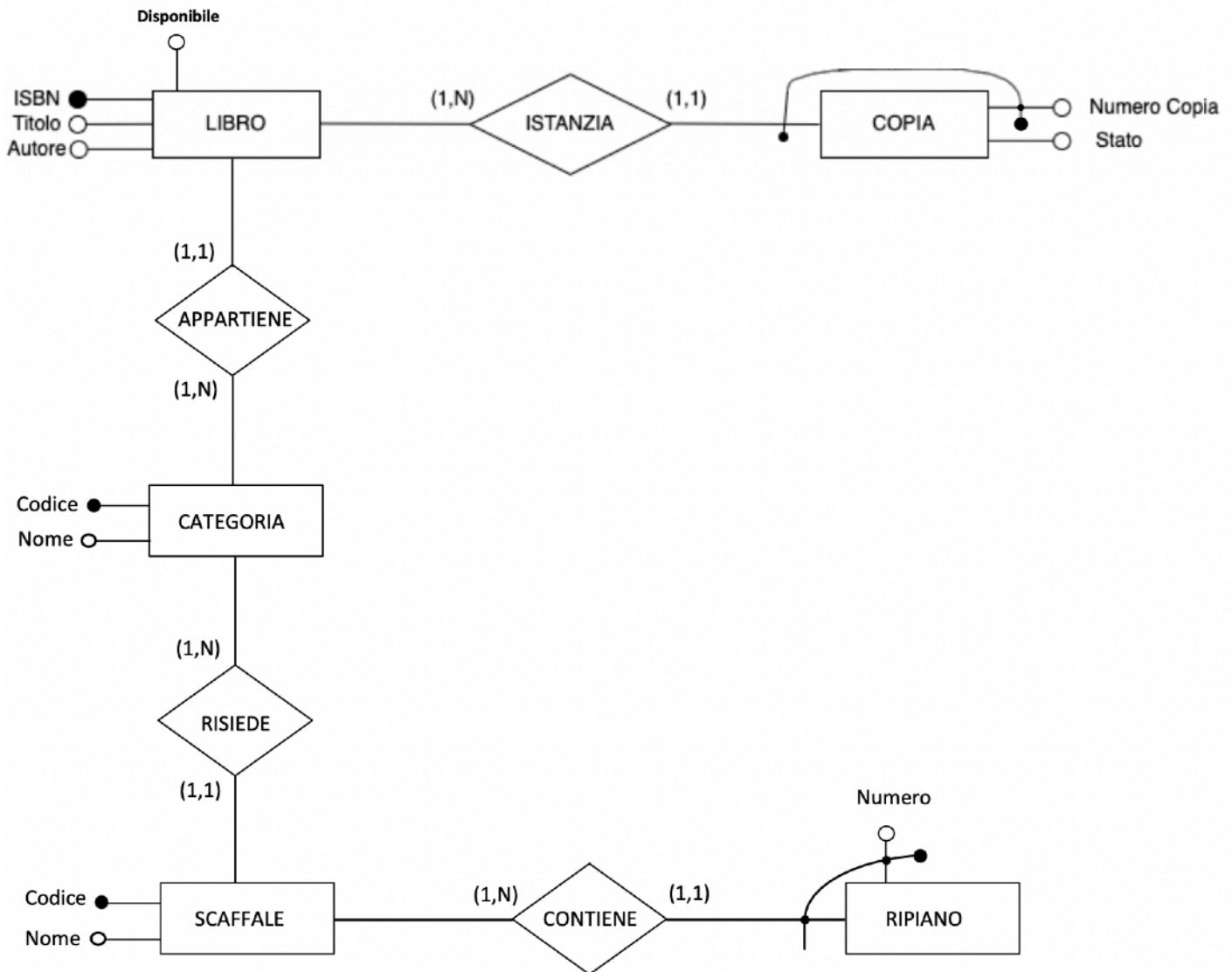
Però una categoria potrebbe risiedere in uno o più scaffali.



Un ulteriore livello di dettaglio nella modellazione prevede la rappresentazione della struttura fisica interna degli scaffali. A tal proposito, si osserva che ogni scaffale è composto da uno o più ripiani, mentre ciascun ripiano è contenuto in uno e un solo scaffale.

Sebbene sarebbe possibile assegnare un identificatore autonomo al ripiano (ad esempio un codice univoco), si è scelto, per coerenza e semplicità, di adottare una chiave esterna per l'identificazione. In particolare, l'identità di ciascun ripiano è determinata in modo univoco tramite:

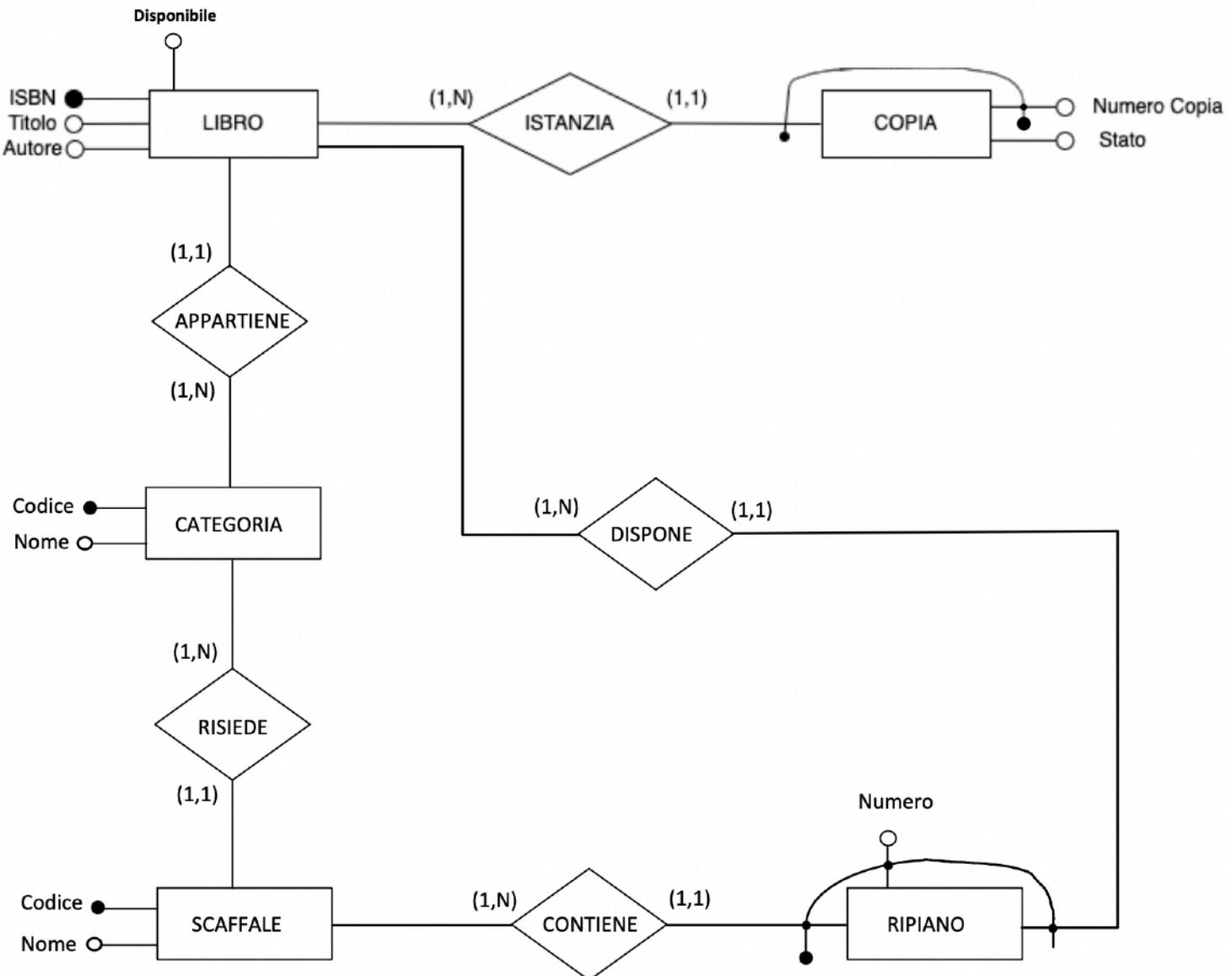
- il codice dello scaffale (identificatore esterno),
- e il codice del ripiano (attributo discriminante interno).



All'interno della modellazione concettuale della biblioteca, si introduce una relazione tra le entità RIPIANO e LIBRO, volta a rappresentare la collocazione fisica degli ISBN dei libri.

Sebbene la copia sia l'elemento materiale che occupa fisicamente il ripiano, è importante osservare che ciascuna copia è istanza di un determinato libro, e dunque è riconducibile a uno specifico ISBN.

Pertanto, la relazione di disposizione può essere interpretata anche a livello concettuale come una disposizione dei libri (intesi tramite ISBN) nei ripiani, attraverso le loro rispettive copie.



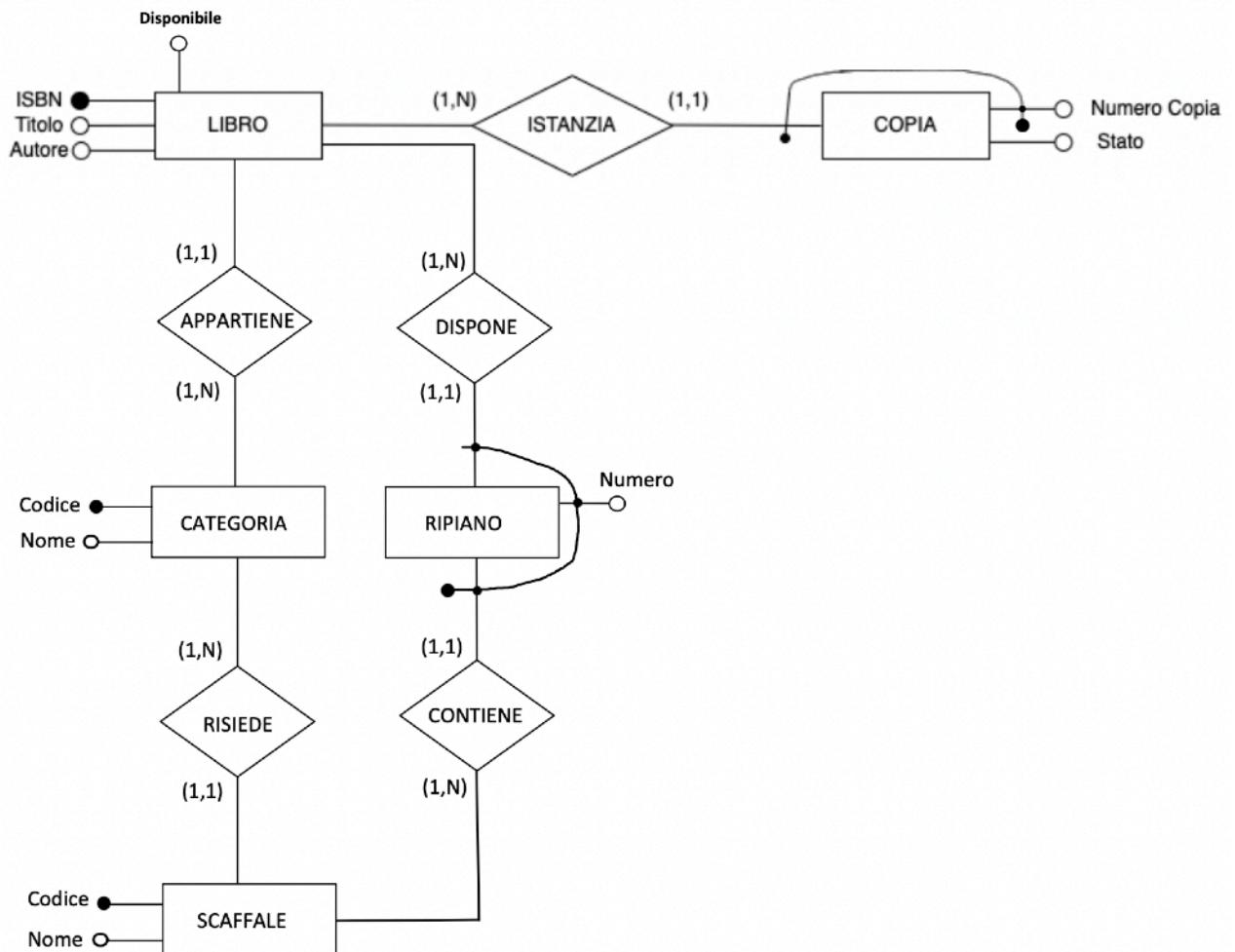
In realtà, il libro (inteso come concetto) non "risiede" in un ripiano, ma sono le copie a essere collocate fisicamente. Tuttavia, per esigenza di progetto si decide che ogni ISBN ha una collocazione univoca (quindi tutte le copie dello stesso libro stanno nello stesso ripiano). Dal punto di vista concettuale, un medesimo libro potrebbe teoricamente essere collocato in più ripiani dello stesso scaffale, qualora si ritenesse necessario suddividerne le copie per ragioni di spazio o organizzazione, e questo non è quello che vorremmo ottenere.

Tuttavia, per garantire **univocità**, dopo verrà effettuato un aggiustamento di queste entità.

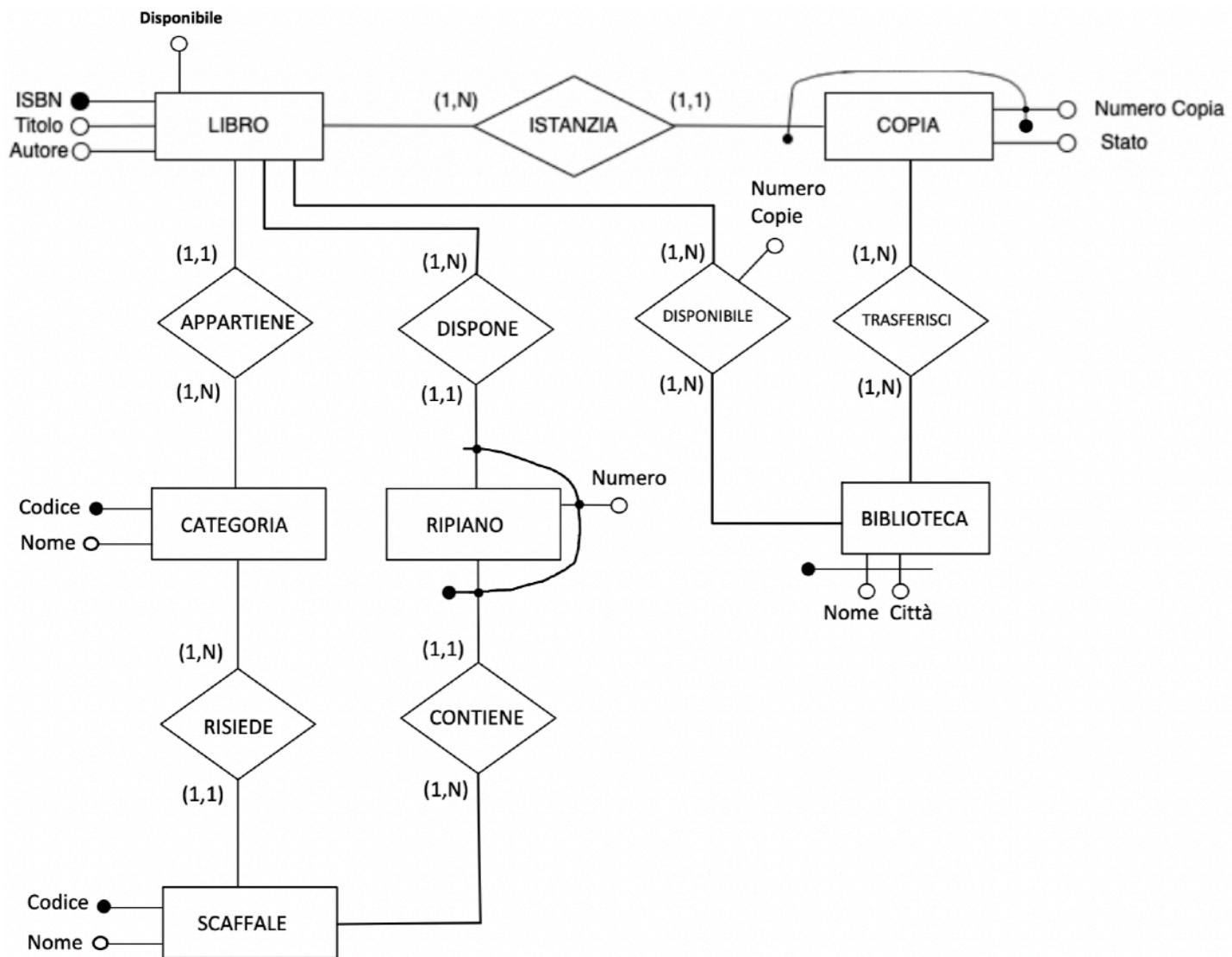
Si vuole consentire di mantenere l'associazione univoca tra ogni ISBN e un singolo scaffale.

L'adozione di un identificatore esterno di ripiano verso libro consente di individuare in maniera univoca un ripiano mediante il codice dello scaffale, il numero del ripiano e l'opera libraria che vi sarà collocata, definendo così una disposizione singolare e irripetibile.

Il modello E/R riscritto con sembianza grafica migliore è il seguente:



Ora la copia può essere prestata a un'altra biblioteca oppure essere disponibile per il prestito da un'altra biblioteca. Pertanto, aggiungiamo due relazioni che collegano i trasferimenti e le disponibilità.



È importante prestare attenzione a quanto accade nel nostro minimondo di riferimento, che è la **biblioteca**. All'interno di essa sono presenti le **copie dei vari libri**, disponibili anche per la vendita.

Quando si effettua il **trasferimento** di un determinato libro, la biblioteca deve tenere traccia delle **copie trasferite**. Pertanto, l'entità *Copia* deve essere in relazione con l'entità *Biblioteca*: **una copia può essere trasferita a una o più biblioteche, e una biblioteca può ricevere una o più copie**.

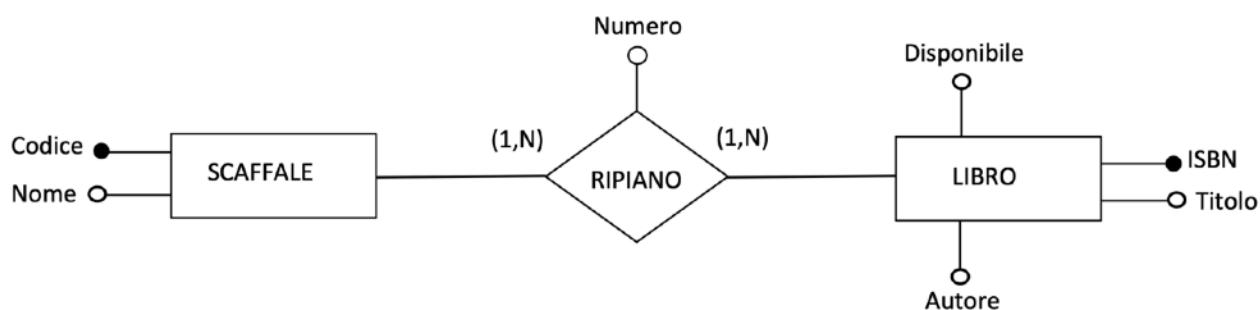
Nel caso in cui una copia non sia disponibile nella biblioteca del nostro minimondo, si verifica se il libro è reperibile presso le **altre biblioteche del circuito**. In tal caso, sono queste ultime a gestire la **propria copia fisica**, mentre la biblioteca che effettua la richiesta **richiede il prestito del libro, senza preoccuparsi di tracciare le copie appartenenti ad altre biblioteche**.

Per quanto riguarda l'entità *Biblioteca*, aggiungiamo un id univoco.

Avrei potuto tranquillamente utilizzare un identificativo univoco per l'utente, ma ho preferito adottare una chiave primaria composta, basata su nome, cognome e data di nascita, poiché si presume che non esistano utenti con lo stesso nome e cognome nati esattamente nello stesso giorno, mese e anno.

Si noti che l'entità Ripiano attuale rappresenta una **reificazione** della relazione Ripiano, poiché avrei potuto modellare la situazione utilizzando direttamente delle relazioni: Libro in relazione 1:N con Ripiano e Scaffale anch'esso in relazione 1:N con Ripiano.

In questo modo, avrei ottenuto il seguente schema:



In questo modo, un'occorrenza della relazione “RIPIANO” è un insieme di coppie (SCAFFALE,LIBRO) senza duplicati. Un libro può apparire una sola volta per ogni scaffale, indipendentemente dal ripiano.

E questo, se ci pensiamo, è **esattamente ciò che si vuole**: la **disposizione di un libro deve essere unica**. Una volta che un libro è stato collocato in uno **scaffale specifico e in uno specifico ripiano**, quell'**ISBN non deve comparire altrove, al massimo può comparire in un altro scaffale**.

Non vogliamo che un libro appaia più volte nello stesso scaffale, su ripiani diversi, non avrebbe senso.



Si osservi che è stata prevista anche una relazione finalizzata alla mappatura dei prestiti relativi ai libri richiesti dagli utenti presso una specifica biblioteca. Tale relazione è modellata mediante un'associazione ternaria di tipo molti-a-molti, denominata **Prestito_Altrove**.

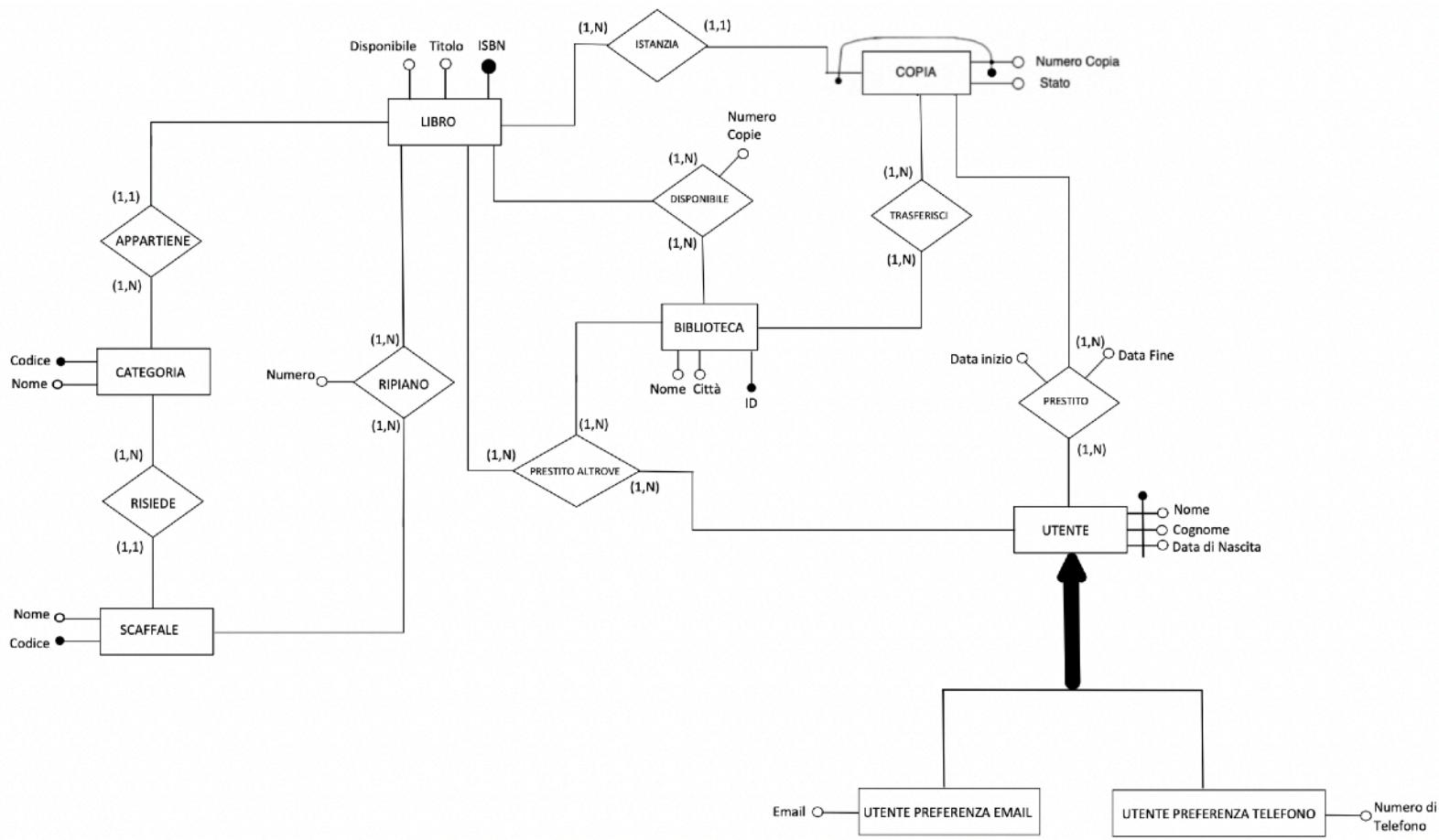
Integrazione finale.

Per maggiore chiarezza, va precisato che l'attributo “Numero Copie” nella relazione DISPONIBILE potrebbe risultare ambiguo, poiché esiste già un'entità denominata COPIA che tiene traccia puntuale di ogni singola copia.

Tuttavia, questo attributo non è ridondante né non atomico, in quanto non si riferisce alle copie gestite direttamente dalla biblioteca del minimondo di riferimento, bensì indica il numero di copie disponibili presso una biblioteca esterna appartenente al circuito.

Si tratta, quindi, di un'informazione aggregata e dichiarativa, utile per sapere quante copie sono disponibili per il prestito da parte di altre biblioteche, senza dover tracciare nel dettaglio ciascuna singola copia fisica di queste ultime.

Di seguito lo schema E/R definitivo:



Si noti che questo schema E/R presenta numerosi cicli tra le diverse entità. Tuttavia, in questo caso, la presenza dei cicli è giustificata, poiché si è resa necessaria per rappresentare correttamente tutte le informazioni: libri disponibili in più biblioteche, copie trasferite, e libri appartenenti a categorie specifiche, collocati in uno scaffale preciso, su un determinato ripiano.

La costruzione del ripiano utilizzando un identificatore esterno è stata una scelta volontaria, pensata appositamente per evidenziare come la situazione potesse essere risolta senza la reificazione.

Ora una copia può essere richiesta da un utente, in particolare da un utente che ha specificato un metodo di contatto preferito tramite il quale desidera essere contattato. Per questo motivo, viene introdotta una generalizzazione totale: tutti gli utenti devono indicare una preferenza, scegliendo tra email o cellulare come metodo di contatto.

Regole Aziendali.

1. Una copia può trovarsi in un solo ripiano di un solo scaffale alla volta.
2. Una copia può essere assegnata ad uno specifico utente solo se è nello stato *Disponibile*.
3. Una copia può essere trasferita tra biblioteche solo se è nello stato *Disponibile*.
4. Un utente può richiedere in prestito solo libri non dismessi.
5. Un utente non può avere in prestito più copie in quanto ogni prestito è associato a una singola copia alla volta.
6. L'utente che riconsegna una copia deve effettuare nuovamente la registrazione per richiedere un nuovo prestito poiché le informazioni a lui attribuite vengono perse all'atto della consegna.
7. Prima di effettuare una nuova richiesta, l'utente deve **restituire la copia precedentemente assegnata**.
8. Un utente deve avere una preferenza di contatto obbligatoria, scelta tra email o telefono.
9. Un utente può richiedere una copia solo se ha indicato una preferenza di contatto valida (email o cellulare).
10. Solo le copie associate a libri non dismessi possono essere prestate.
11. Un libro è considerato **non disponibile** se è stato dismesso dagli amministratori.
12. Ogni biblioteca del circuito può comunicare il numero di copie disponibili per un determinato libro, anche se queste copie non sono gestite direttamente dalla biblioteca di riferimento del sistema.
13. Una copia può essere trasferita solo se disponibile.
14. Ogni copia è associata a un singolo **libro** e a una specifica **biblioteca**.
15. Lo **stato** di una copia può essere solo uno tra: "*Disponibile*", "*In prestito utente*", "*In prestito biblioteca*".
16. Ogni categoria può risiedere in uno e un solo scaffale, mentre più scaffali distinti possono ospitare la stessa categoria.
17. Un libro può apparire una sola volta per ogni scaffale, indipendentemente dal ripiano.
18. Non è possibile aggiungere una nuova copia se il libro corrispondente è stato dismesso.
19. Non si può eliminare una copia se è attualmente in prestito.
20. Il codice categoria segue la classificazione decimale Dewey e deve essere univoco.
21. L'ISBN è un identificatore univoco a livello globale e non può essere duplicato.
22. La disponibilità di una copia viene verificata preliminarmente, prima di richiedere all'utente l'inserimento di nome, cognome e data di nascita. Soltanto nel caso in cui la copia risulti effettivamente disponibile, l'utente potrà procedere con la registrazione. In caso contrario, la registrazione non è consentita.

23. Nel caso in cui un **libro risulti disponibile in più biblioteche del circuito**, l'utente ha la **facoltà di scegliere** presso quale di esse **effettuare il ritiro** della copia.

Tale scelta viene effettuata **al momento della richiesta**, in base alle disponibilità comunicate dalle biblioteche.

24. Nel momento in cui l'utente richiede una copia “*Disponibile*”, questa viene marcata come “*In prestito utente*”.

25. Nel momento in cui una biblioteca richiede la copia di un libro “*Disponibile*”, questa viene marcata come “*In prestito biblioteca*”.

26. Lo scaffale che contiene una determinata copia deve afferire ad una categoria.

27. Se una copia di un libro dismesso è già in prestito, quell'utente può tenerla fino alla restituzione.

28. Ogni libro (ISBN) afferisce ad una sola categoria.

29. Esistono più scaffali che possono contenere copie afferenti ad una sola categoria.

Dizionario dei Dati.

| ENTITÀ | DESCRIZIONE | ATTRIBUTI | IDENTIFICATORE |
|--------------------------------|---|--|--|
| <i>Libro</i> | Opera editoriale identificata da un codice univoco. | ISBN, Titolo, Autore, Disponibile. | <u>ISBN</u> |
| <i>Copia</i> | Singola copia fisica di un libro. | Numero Copia, Stato. | <u>Libro, Numero Copia.</u> |
| <i>Utente</i> | Persona che può effettuare prestiti di copie. | Nome, Cognome, Data di Nascita. | <u>Nome, Cognome, Data di Nascita.</u> |
| <i>Utente Preferenza Email</i> | Specifiche che l'utente desidera essere contattato tramite email. | Nome, Cognome, Data di Nascita, Email. | <u>Nome, Cognome, Data di Nascita.</u> |

| ENTITÀ | DESCRIZIONE | ATTRIBUTI | IDENTIFICATORE |
|-----------------------------------|---|---|--|
| <i>Utente Preferenza Telefono</i> | Specifiche che l'utente desidera essere contattato tramite telefono. | Nome, Cognome, Data di Nascita, Telefono. | <u>Nome, Cognome, Data di Nascita.</u> |
| <i>Biblioteca</i> | Sede fisica facente parte del circuito bibliotecario. | Nome, Città. | <u>Nome, Città.</u> |
| <i>Scaffale</i> | Contenitore fisico di ripiani. | Codice, Nome. | <u>Codice.</u> |
| <i>Categoria</i> | Classificazione tematica dei libri, secondo un criterio organizzativo standardizzato. | Codice, Nome. | <u>Codice.</u> |

| RELAZIONE | DESCRIZIONE | ENTITÀ COINVOLTE | ATTRIBUTI |
|--------------------|---|-----------------------------|----------------------|
| <i>Ripiano</i> | Associa un libro ad uno scaffale. | <i>Libro, Scaffale</i> | <i>Numero.</i> |
| <i>Appartiene</i> | Associa un libro alla sua categoria. | <i>Libro, Categoria.</i> | |
| <i>Disponibile</i> | Associa un libro alla biblioteca presso cui è disponibile. | <i>Libro, Biblioteca.</i> | <i>Numero Copie.</i> |
| <i>Risiede</i> | Associa una categoria ad un determinato scaffale. | <i>Categoria, Scaffale.</i> | |
| <i>Trasferisci</i> | Associa una copia trasferita ad una determinata biblioteca. | <i>Copia, Biblioteca</i> | |

| RELAZIONE | DESCRIZIONE | ENTITÀ COINVOLTE | ATTRIBUTI |
|-------------------------|---|----------------------------------|--------------------------------|
| <i>Prestito</i> | Associa una copia disponibile ad un utente che la ritira. | <i>Copia, Utente.</i> | <i>Data Inizio, Data Fine.</i> |
| <i>Prestito Altrove</i> | Prestito Esterno | <i>Biblioteca, Utente, Libro</i> | |

Progettazione Logica .

Per garantire un'adeguata progettazione del sistema bibliotecario, è stato necessario stimare il **volume atteso dei dati** che il sistema dovrà gestire. A tal fine, sono state definite **ipotesi ragionevoli** basate su un possibile contesto d'uso, in cui il sistema viene adottato da una **rete di 20 biblioteche** operanti in una città di medie dimensioni.

Impostiamo delle ipotesi operative.

Ogni biblioteca gestisce in media 500 libri: $500 \times 20 = 10.000$ libri nel catalogo.

Ogni libro ha in media **5 copie**: $10.000 \times 5 = 50.000$ copie.

La biblioteca del minimondo effettua circa **10 prestiti al giorno**.

Sistema attivo **300 giorni/anno** (tolti festivi e chiusure): $10 \times 300 = 3.000$ prestiti/anno.

Ogni biblioteca riceve o invia **15 copie al mese**: $15 \times 20 \times 12 = 3.600$ trasferimenti/anno.

Circa 2.000 utenti attivi nel circuito.

Il 60% preferisce **email**, il 40% **telefono**.

Il numero degli scaffali è 10 per ogni biblioteca.

Il numero dei ripiani per ogni scaffale è 5.

La tavola dei volumi è la seguente:

Volume dei dati.

| CONCETTO | TIPO | VOLUME ATTESO |
|----------------------------|------|---------------|
| LIBRO | E | 10.000 |
| CATEGORIA | E | 100 |
| SCAFFALE | E | 200 |
| BIBLIOTECA | E | 20 |
| COPIA | E | 50.000 |
| UTENTE | E | 2000 |
| UTENTE PREFERENZA EMAIL | E | 1200 |
| UTENTE PREFERENZA TELEFONO | E | 800 |
| APPARTIENE | R | 10.000 |
| DISPONIBILE | R | 15.000 |
| RISIEDE | R | 200 |
| RIPIANO | R | 1000 |
| PRESTITO | R | 3000 |
| TRASFERISCI | R | 250 |
| PRESTITO ALTROVE | R | 750 |

Piccole spiegazioni.

Se nel sistema sono presenti 100 categorie (es. secondo la classificazione Dewey), e 10.000 libri, dire che la relazione APPARTIENE ha 10.000 occorrenze non significa che ci sono 10.000 categorie, ma che ogni libro appartiene a una (e una sola) categoria.

Se nel sistema ci sono 50.000 copie totali gestite internamente (cioè Copia ha 50.000 istanze), Il 30% di queste (ossia 15.000) è una stima ragionevole del numero di copie che le altre biblioteche esterne al sistema dichiarano come disponibili per il prestito attraverso la relazione DISPONIBILE.

Se ogni categoria risiede in 2 scaffali in media, $100 \text{ categorie} \times 2 = 200$.

Se la biblioteca del minimondo effettua 10 prestiti al giorno, e il sistema è attivo 300 giorni, il numero di prestiti stimato è $300 \times 10 = 3.000$.

La relazione TRASFERISCI rappresenta le copie che la biblioteca del minimondo invia ad altre biblioteche del circuito. Ho 500 libri, ciascuno con 5 copie: $500 \times 5 = 2.500$ copie gestite nella biblioteca. Non tutte le copie verranno trasferite. Possiamo fare un'ipotesi realistica: ad esempio, il 10% delle copie potrebbe essere trasferito durante l'anno, $10\% \text{ di } 2.500 = 250$ copie, sono quelle che, secondo l'ipotesi, vengono trasferite ad altre biblioteche dal minimondo.

Ipotesi: 5% di 15.000 copie disponibili, ho 750 prestiti da altre biblioteche

Tavola delle Operazioni.

L'obiettivo della presente analisi è quello di porre l'attenzione unicamente sulle operazioni di maggiore rilevanza, escludendo deliberatamente quelle di natura routinaria o soggette ad automatizzazione. Per tale ragione, nella tavola delle operazioni sono state omesse tutte le attività ritenute di carattere elementare o trascurabile, ossia operazioni che riguardano la produzione di report relativi al personale, comprendendo sia i bibliotecari sia gli amministratori.

| CODICE | OPERAZIONE | FREQUENZA ATTESA | MOTIVAZIONE |
|-------------|---|---------------------|---|
| OP01 | Richiesta di prestito da parte dell'utente. | 200/Giorno | Operazione frequente da parte degli utenti. |
| OP02 | Inserimento di un nuovo libro nel catalogo. | 50/Mese | |
| OP03 | Dismissione di un libro. | 30/Mese | |
| OP04 | Trasferimento di una copia verso un'altra biblioteca. | 20/Settimana | |
| OP05 | Verifica disponibilità di una copia. | 200/Giorno | |

Le frequenze attese sono state stimate sulla base di un'ipotesi operativa che prevede una biblioteca con **2.000 utenti attivi, 10.000 libri e 50.000 copie.**

Nota: Se la stima di OP01 è 200/Giorno, allora ci sono **altrettante verifiche**, perché ogni prestito è preceduto da una sola verifica effettiva. La verifica avviene prima di ogni richiesta di prestito. Ogni utente, prima di registrarsi o di chiedere una copia, controlla se c'è disponibilità, ma appena la copia è disponibile il prestito viene effettuato.

| CODICE | OPERAZIONE | FREQUENZA ATTESA | MOTIVAZIONE |
|-------------|---|----------------------------|---|
| OP06 | Richiesta di copia da biblioteca esterna. | 30 richieste/giorno | Supponiamo che il 15% delle richieste di prestito si riferiscono a copie non disponibili in sede ma disponibili altrove. Ci sono 200 prestiti al giorno, il 15% è 30. |

Costo delle Operazioni.

Ipotizziamo che il costo in scrittura di un dato sia doppio rispetto a quello in lettura.

OP01 - Richiesta di prestito dell'utente.

Entità e relazioni coinvolte: *UTENTE, COPIA, PRESTITO E LIBRO.*

Utente: Per inserire un nuovo utente che richiede il prestito.

Copia: Viene letta per verificarne lo stato (“Disponibile”) e successivamente **modificata** per impostare lo stato a “In prestito” una volta che la copia è stata assegnata.

Prestito: Viene **creata una nuova istanza** di prestito, associata a un utente e a una copia, con indicazione della data di inizio (e successivamente, della data di fine).

Libro: Per conoscere i dati del libro richiesto (es. titolo, autore).

Numero accessi totali = (aggiornamento dello stato dell'utente che può avere al massimo un libro in prestito + verifica e modifica dello stato della copia + la registrazione dell'evento di prestito + ottenere info bibliografiche da libro) × 200 = [2+(1+2)+2+1] × 200 = 1600/Giorno.

OP02 - Inserimento di un nuovo libro nel catalogo.

Entità e relazioni coinvolte: *LIBRO, CATEGORIA, RIPIANO, SCAFFALE, APPARTIENE E RISIEDE*.

Libro: Inserimento dei dati del nuovo libro.

Categoria: Per selezionare la categoria a cui il libro appartiene.

Ripiano: Per scegliere su quale ripiano posizionare fisicamente la copia.

Scaffale: Per individuare lo scaffale di riferimento.

Appartiene: Per associare il libro alla categoria selezionata.

Risiede: Per verificare quali categorie possono risiedere nello scaffale scelto.

Si considera solo l'inserimento del **libro**, non delle **copie**.

Durante l'inserimento del libro si decide di collocarlo fisicamente su un ripiano specifico: l'operazione comporta una scrittura anche sull'entità **RIPIANO**.

Numero accessi totali = $(2+1+2+1+2+1) \times 50 = 450/\text{Mese}$.

L'accesso in scrittura avviene nel Libro, scrittura del Ripiano e scrittura della Categoria tra quelle disponibili.

L'operazione di inserimento di un nuovo libro comporta la scrittura dell'entità **LIBRO** e l'associazione a una categoria esistente .

Il bibliotecario seleziona anche uno scaffale e un ripiano, consultando le entità **CATEGORIA**, **SCAFFALE** e **RISIEDE** per garantire la coerenza tra classificazione e collocazione fisica.

Infine, viene registrata la posizione fisica nel **RIPIANO**, tramite scrittura diretta o inserimento in una tabella associativa.

L'operazione ha un costo unitario di 9 unità computazionali e una frequenza stimata di 50 al mese, per un costo totale mensile di 450 unità.

OP03 - Dismissione di un libro.

L'operazione **non elimina il libro fisicamente**, ma **ne aggiorna lo stato**, rendendolo **non più disponibile** al prestito.

Entità e relazioni coinvolte: *LIBRO*.

L'amministratore aggiorna lo stato del libro per indicare che non è più disponibile al prestito. Le copie esistenti non vengono eliminate né aggiornate.

Le copie restano intatte: non si modificano, ma diventano “**inutilizzabili**” perché il sistema nega nuove assegnazioni **in base allo stato del libro**.

Se una copia di un libro dismesso è già in prestito, quell'utente può tenerla fino alla restituzione.

Il controllo sullo stato del libro (disponibile: “No”) viene eseguito al momento di una nuova richiesta di prestito.

Pertanto, non è necessario accedere alla relazione PRESTITO durante la dismissione.

L'unica scrittura riguarda l'entità LIBRO, dove l'attributo Disponibile viene impostato su “No”.

Numero accessi totali = $2 \times 30 = 60/\text{Mese}$.

OP04 - Trasferimento di una copia verso un'altra biblioteca.

Il trasferimento di una copia comporta la modifica dell'entità COPIA, in particolare per aggiornare la disponibilità. Viene inoltre creata una nuova istanza della relazione TRASFERISCI, contenente le informazioni del trasferimento. L'operazione richiede anche l'accesso in lettura alla tabella BIBLIOTECA, per selezionare la biblioteca ricevente, e al LIBRO, per recuperare le informazioni descrittive collegate alla copia.

Entità e relazioni coinvolte: *COPIA, TRASFERISCI, BIBLIOTECA E LIBRO*.

Numero accessi totali = $(2+2+1+1) \times 20 = 120/\text{Settimana, ossia } 480/\text{Mese}$.

OP05 - Verifica disponibilità di una copia.

Questa è un'operazione interattiva molto frequente, eseguita dall'utente prima di effettuare un prestito.

Il sistema deve controllare se esistono copie disponibili per il libro richiesto, sia:

- nella biblioteca locale (minimondo),
- sia eventualmente in biblioteche esterne del circuito (tramite la relazione DISPONIBILE).

Entità e relazioni coinvolte: *LIBRO, COPIA E DISPONIBILE*.

L'operazione di verifica della disponibilità di una copia viene eseguita prima di ogni richiesta di prestito. Essa comporta l'accesso in lettura all'entità LIBRO per identificare il titolo cercato, alla tabella COPIA per verificare se esistono copie locali assegnabili, e alla relazione DISPONIBILE per controllare eventuali disponibilità in biblioteche esterne al minimondo.

Numero accessi totali = $(1+1+1) \times 200 = 600/\text{Giorno, pari a } 18.000/\text{Mese.}$

OP06 - Richiesta della copia da una biblioteca nel circuito.

Quando una copia non è disponibile presso la biblioteca locale, ma risulta reperibile in una delle biblioteche appartenenti al circuito, il sistema consente all'utente di inoltrare una richiesta interbibliotecaria.

Tale operazione comporta l'accesso in lettura alle entità LIBRO e DISPONIBILE, nonché alla tabella BIBLIOTECA per identificare una sede da cui sia possibile ottenere la copia.

Contestualmente, viene effettuata una scrittura sulla relazione DISPONIBILE per mappare o aggiornare la disponibilità della copia presso la biblioteca selezionata, al fine di tracciare la provenienza della richiesta e gestire il processo di prestito interbibliotecario.

Infine, aggiorno in scrittura Prestito_Altrove.

Entità e relazioni coinvolte: *LIBRO, DISPONIBILE, BIBLIOTECA E PRESTIVO_ALTROVE..*

Numero accessi totali = $[1+(1+2)+1+2] \times 30 = 210/\text{Giorno, } 6.300/\text{Mese.}$

RISTRUTTURAZIONE DELLO SCHEMA E/R

Lo schema E/R necessita ora di una fase di **ristrutturazione**, pur rimanendo evidente l'intento originario di **limitare al minimo le ridondanze**.

Va sottolineato, infatti, che nello schema attuale **non sono presenti dati derivati**, e tutte le informazioni risultano essere **rappresentate in forma atomica**, nel pieno rispetto dei principi di modellazione concettuale.

Un aspetto fondamentale di questa fase di ristrutturazione consisterà nella **rimozione delle generalizzazioni**, al fine di rendere il modello maggiormente compatibile con la successiva traduzione logica e fisica.

1. Analisi delle ridondanze.

Anche se l'attributo Stato può assumere valori come "Disponibile", "In prestito utente", "In prestito biblioteca", non viola l'atomicità, perché:

- È un solo valore per volta,
- Appartiene a un dominio predefinito (enumerato);

2. Partizionamento.

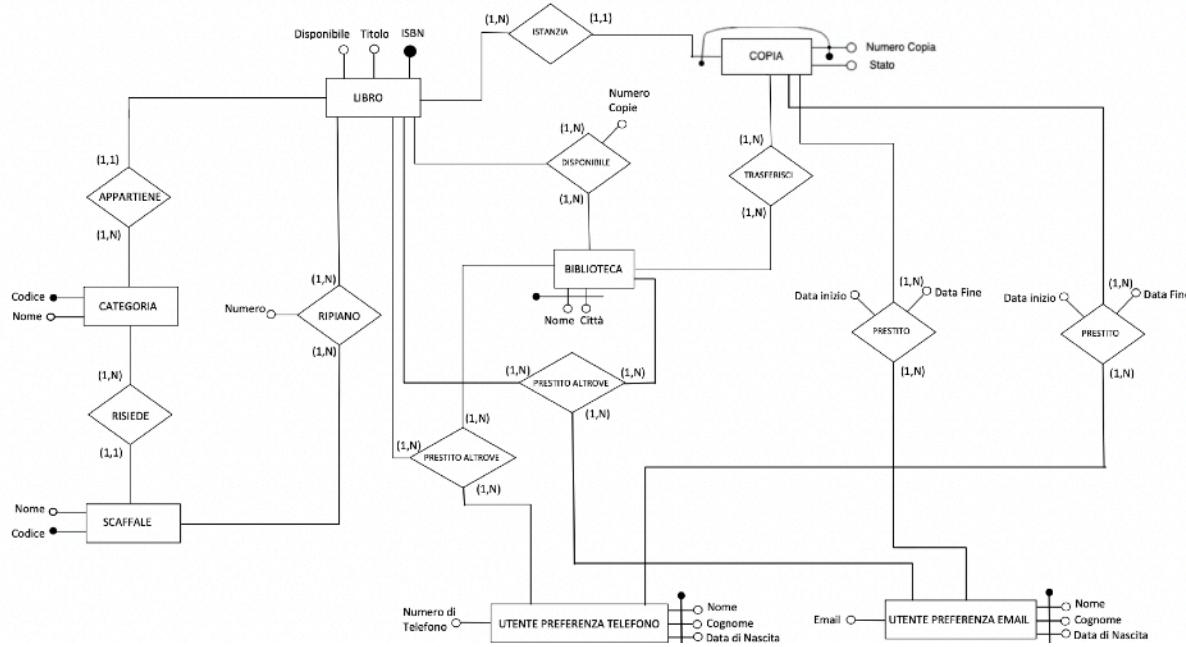
Non è necessario partizionare entità o associazioni.

Sebbene non si sia reso necessario un partizionamento esplicito delle entità o relazioni nella fase di ristrutturazione, si osserva che una forma di partizionamento concettuale è già stata realizzata in sede di progettazione del modello E/R.

In particolare, è stata operata una distinzione strutturale tra due dinamiche concettualmente differenti:

- da un lato, il prestito verso un utente appartenente al dominio del minimondo (modellato tramite la relazione PRESTITO);
- dall'altro, il trasferimento di una copia fisica verso una biblioteca esterna appartenente al circuito (modellato tramite la relazione TRASFERISCI).
- A ciò si aggiunge la presenza della relazione DISPONIBILE, che consente di **rappresentare la disponibilità di un libro presso biblioteche esterne** al minimondo, mantenendo così separata la gestione della disponibilità interna da quella esterna.

Lo schema E/R ristrutturato è il seguente:



In questa fase, si procede con un accorpamento delle entità figlie della generalizzazione, mediante l'eliminazione dell'entità genitore UTENTE.

Gli attributi, l'identificatore primario e tutte le relazioni alle quali l'entità UTENTE partecipava vengono ereditati ed assorbiti dalle due entità figlie:

UTENTE_PREFERENZA_EMAIL e UTENTE_PREFERENZA_TELEFONO.

Tale operazione consente di rimuovere la generalizzazione dal modello, semplificandone la struttura e rendendolo più aderente alla logica relazionale.

Il comportamento funzionale del sistema resta invariato, in quanto ogni utente continuerà a essere rappresentato integralmente, ma in forma specifica e disgiunta a seconda della preferenza di contatto dichiarata.

Inoltre, la scelta di **eliminare la generalizzazione** a favore dell'**accorpamento delle entità figlie** si rivela particolarmente conveniente quando le **operazioni applicative fanno riferimento esclusivamente a occorrenze specifiche di una delle entità figlie**, distinguendo in maniera netta i comportamenti o le funzionalità associate.

In scenari di questo tipo, mantenere una struttura generalizzata porterebbe a una maggiore complessità sia nella definizione delle operazioni sia nella loro implementazione, mentre la separazione netta delle entità consente una **modellazione più mirata e un'interazione più efficiente con i dati**.

Tuttavia, è fondamentale sottolineare che questa trasformazione è **corretta e consigliabile solo nel caso in cui la generalizzazione sia totale**, ovvero quando **ogni istanza del genitore appartiene necessariamente a una ed una sola delle entità figlie**.

In caso contrario, tale semplificazione introdurrebbe **perdita di generalità e ambiguità semantica**, compromettendo l'integrità del modello.

Nel modello concettuale non sono stati effettuati partizionamenti di entità, in quanto non si sono presentate situazioni in cui tale scelta progettuale fosse necessaria o funzionalmente vantaggiosa.

3. Identificatori principali.

Gli identificatori principali delle entità sono chiaramente visibili a livello grafico, e risultano immediati da individuare mediante un'adeguata lettura dello schema E/R, nel rispetto della notazione adottata.

Essi sono stati scelti nel pieno rispetto delle regole di integrità del modello relazionale, assicurando in particolare il vincolo di **non nullabilità**.

Si è data preferenza, ove possibile, a identificatori composti da un numero ridotto di attributi, in quanto tali soluzioni risultano generalmente più efficienti e leggibili rispetto a chiavi composte da molti campi.

All'interno dello schema è stato introdotto un solo identificatore esterno, in corrispondenza della relazione ISTANZIA tra LIBRO e COPIA.

Come si vedrà nella fase di traduzione al modello logico, anche gli identificatori esterni vengono correttamente trattati, traducendosi in chiavi primarie che includono gli identificatori delle entità coinvolte nella relazione di identificazione.

Scelte progettuali guidate dal costo delle operazioni

1. Separare TRASFERISCI e PRESTITO in due relazioni distinte.

Entrambi rappresentano una forma di movimento della copia, ma si è scelto di tenerli **ben distinti**. Le operazioni che coinvolgono solo l'utente o solo le biblioteche sono più semplici, mirate e ottimizzabili.

Le due operazioni rispondono a **logiche distinte**:

- PRESTITO coinvolge utenti finali.
- TRASFERISCI coinvolge solo biblioteche.

Inoltre, mantenere le due relazioni separate semplifica le query applicative e riduce il costo computazionale per ciascun tipo di operazione.

2. Modellazione della disponibilità esterna con una relazione autonoma DISPONIBILE.

La disponibilità delle copie nelle biblioteche del circuito non è calcolata dinamicamente (derivata), ma è **modellata esplicitamente** tramite la relazione DISPONIBILE, che associa una biblioteca a un libro con un attributo *Numer copie*.

L'operazione di verifica ha una frequenza elevatissima (6000/mese), ossia 200/Giorno, come abbiamo visto.

Avere la disponibilità già pronta evita calcoli complessi e migliora le prestazioni di lettura, a fronte di un minimo grado di ridondanza.

3. Relazione APPARTIENE modellata come FK e non come tabella.

Evita la creazione di una tabella inutile, semplifica la struttura logica e **riduce il numero di join** nelle operazioni di classificazione..

4. Attribuzione fisica del ripiano già in fase di inserimento libro.

Il libro viene direttamente collocato su un ripiano al momento della sua creazione nel catalogo, evitando operazioni separate per la collocazione fisica.

Anche se questo comporta un leggero aumento del costo in fase di inserimento (scrittura su RIPIANO), permette di evitare operazioni future e migliora l'efficienza nella gestione delle copie.

Traduzione di entità e associazioni nello schema relazionale.

Forma Int. delle Tabelle.

Libro(**ISBN**, Titolo, Disponibile, **Codice_Categoria**);

Categoria(**Codice_Categoria**, nome);

Scaffale(**Codice_Scaffale**, nome, **Codice_Categoria**);

Ripiano(**ISBN**, **Codice_Scaffale**, Numero_Ripiano);

Copia(**ISBN**, **Numero_Copia**, stato);

Utente_Teléfono(**Nome**, **Cognome**, **Data_Nascita**, Telefono);

Utente_Email(**Nome**, **Cognome**, **Data_Nascita**, Email);

Disponibilità_Copia(**ISBN**, **ID_Biblio** Numero_Copie_Disponibili);

Trasferimento_Copia(**ISBN**, **Numero_Copia**, **ID_Biblio**);

Prestito(**ISBN**, **Numero_Copia**, **Nome_Utente**, **Cognome_Utente**, **Data_Nascita**,

Data_Inizio, **Data_Fine**);

Biblioteca(ID, Nome,Città);

Utente(Nome, Cognome, Data_Nascita**):**

Prestito_Altrove(ISBN, Nome, Cognome, Data_Nascita, ID_BIBLIO, Data_Inizio, Data_Fine);

Nella tabella Trasferimento_Copia, è opportuno e corretto utilizzare entrambe le componenti della chiave primaria della relazione COPIA, ovvero ISBN e Numero_Copia, come foreign key composite.

In fase di progettazione logica è stata adottata una scelta implementativa strutturata, volta a coniugare chiarezza semantica. È stata infatti creata una tabella Utente principale, che contiene gli attributi identificativi comuni a tutti gli utenti (come Nome, Cognome, Data_Nascita), indipendentemente dalla tipologia di contatto preferita.

Le due tavole derivate, Utente_Email e Utente_Teléfono, sono state mantenute per rappresentare in maniera disgiunta e completa la scelta del canale comunicativo preferito da ciascun utente.

Ciascuna di esse include una foreign key verso la tabella Utente, garantendo l'integrità referenziale e consentendo l'accesso centralizzato alle informazioni anagrafiche, pur preservando la specializzazione funzionale nella gestione dei contatti.

Inoltre, se un'entità figlia è una specializzazione totale e disgiunta dell'entità padre, allora:

- Ogni riga della sottotabella è **un'estensione esatta** di una riga del padre.
- Deve esserci **una corrispondenza 1:1** tra la PK del padre e quella della figlia.

La chiave primaria delle tavole specializzate coincide con la foreign key verso la tabella Utente (Mia scelta progettuale consapevole).

Questo avviene perché tali tavole rappresentano **specializzazioni disgiunte e totali**, e quindi ogni occorrenza nelle tavole Utente_Email e Utente_Teléfono deve corrispondere univocamente a un'istanza presente nella tabella Utente.

È proprio per questo motivo che, nelle tavole specifiche relative all'utente, la **tripla (Nome, Cognome, Data_Nascita)** **funge sia da chiave esterna verso Utente, sia da chiave primaria della tabella stessa.**

Gli attributi Nome, Cognome, Data_Nascita sono condivisi da entrambe le specializzazioni: **mantenerli in una tabella unica evita ridondanza e ripetizione dei dati.**

La tabella Utente diventa il *riferimento solido* per tutte le entità che devono interagire con un utente (es. Prestito): in questo modo puoi collegarti sempre al nucleo anagrafico comune.

Normalizzazione del modello Relazionale.

Una relazione r è in forma normale di Boyce e Codd se per ogni dipendenza funzionale (non banale) $X \rightarrow A$ definita su di essa, X contiene una chiave K di r , cioè X è superchiave per r .

In altre parole: "Se qualcosa determina qualcos'altro, allora quel qualcosa deve essere in grado di identificare univocamente tutte le tuple della relazione."

Anomalie e ridondanze non si presentano per relazioni in forma normale di Boyce e Codd.

Si noti che le ridondanze sono causate dalle dipendenze funzionali $X \rightarrow A$ che permettono la presenza di più tuple fra loro uguali, tali che X non contiene una chiave.

Una dipendenza $X \rightarrow Y$ è non banale se Y non è incluso in X .

Nel caso delle tabelle specializzate **Utente_Email** e **Utente_Teléfono**, si osservano le seguenti dipendenze funzionali:

- $Email \rightarrow Nome, Cognome, Data_Nascita$
- $Telefono \rightarrow Nome, Cognome, Data_Nascita$

Poiché ogni utente è associato a un solo indirizzo email o a un solo numero di telefono, tali attributi fungono da superchiavi nelle rispettive tabelle.

Di conseguenza, entrambe le relazioni risultano essere in forma normale di Boyce-Codd (BCNF), in quanto tutte le dipendenze funzionali non banali hanno come determinante una superchiave.

Analizziamo ora correttamente la tabella **Prestito**, con le relative dipendenze.

ISBN, Numero_Copia → Data_Inizio

Una certa copia (con un certo ISBN e Numero_Copia) può essere prestata una sola volta alla volta, e ogni nuovo prestito avrà una riga diversa, con un nuovo Data_Inizio. Nel modello l'esemplare di una copia può essere prestata una sola volta, allora ISBN, Numero_Copia è effettivamente una superchiave, e quindi la dipendenza ISBN, Numero_Copia → Data_Inizio non viola la BCNF.

Poiché ogni copia (ISBN, Numero_Copia) può essere prestata una sola volta, tale coppia può essere considerata una **superchiave**.

Risultano quindi valide le seguenti dipendenze:

- $(ISBN, Numero_Copia) \rightarrow Data_Inizio, Data_Fine$
- $(ISBN, Numero_Copia) \rightarrow Nome, Cognome, Data_Nascita$
- $(ISBN, Numero_Copia) \rightarrow *$

Inoltre, dato che il sistema impedisce che un utente abbia più di una copia contemporaneamente, è valida anche la dipendenza:

- $(Nome, Cognome, Data_Nascita) \rightarrow ISBN, Numero_Copia$

Poiché tutte le dipendenze non banali hanno come determinante una **superchiave**, la relazione Prestito soddisfa i requisiti della forma normale di Boyce-Codd (BCNF).

La tabella Prestito è in forma normale di Boyce-Codd (BCNF).

Analizziamo la tabella **Disponibilità_Copia**.

La tabella, che associa ogni libro (ISBN) a una biblioteca (Nome, Città) con un certo numero di copie disponibili, presenta la seguente **dipendenza funzionale non banale**:

- $(ID_Biblioteca, ISBN) \rightarrow Numero_Copie$

Tale dipendenza ha come determinante la **chiave primaria** della relazione, e non esistono ulteriori dipendenze funzionali valide nel dominio.

$ISBN \rightarrow Numero_Copie$.

Allo stesso ISBN possono essere associate le stesse quantità di copie di biblioteche diverse.

Tuttavia la dipendenza non è valida: La dipendenza $ISBN \rightarrow Numero_Copie$ è valida se e solo se, in ogni possibile istanza della tabella, a ogni valore di ISBN corrisponde uno e un solo valore di $Numero_Copie$.

Non c'è nessuna regola che imponga che il numero di copie sia lo stesso in tutte le biblioteche.

La tabella Disponibilità_Copia è in forma normale di Boyce-Codd (BCNF).

Ora analizziamo la tabella **Trasferimento_Copia**.

$(ISBN, Numero_Copia) \rightarrow ID_Biblioteca$

Questa non è una dipendenza valida, non ho la garanzia che X fornisca attributi uguali per Y univocamente.

La stessa copia può essere trasferita più volte, anche verso biblioteche diverse.

Quindi a un valore fisso di (ISBN, Numero_Copia) possono corrispondere più biblioteche.

L'unica dipendenza funzionale non banale presente è quella indotta dalla chiave primaria completa:

$(ISBN, Numero_Copia, ID_Biblioteca) \rightarrow^* *$

Non esistono altre dipendenze funzionali non banali con lato sinistro non superchiave.

Quindi, la tabella rispetta pienamente la forma normale di Boyce-Codd (BCNF).

Ora analizziamo la tabella **Ripiano**.

L'unica dipendenza funzionale non banale è:

$(ISBN, Codice_Scaffale) \rightarrow Numero_Ripiano$

Essendo il lato sinistro una superchiave, la dipendenza non viola la BCNF.

Le altre dipendenze ipotizzabili non si verificano nel dominio.

La tabella **Ripiano** è quindi in forma normale di Boyce-Codd (BCNF).

Per lo stesso motivo, tutte le altre tabelle rispettano la BCNF.

CONCLUSIONE NORMALIZZAZIONE.

Tutte le tabelle del modello relazionale sono state progettate in modo da soddisfare i vincoli di **integrità semantica e le dipendenze funzionali** del dominio.

Dopo un'analisi approfondita, tutte risultano conformi alla forma normale di Boyce-Codd (BCNF), in quanto:

- Ogni dipendenza funzionale non banale ha come determinante una superchiave,
- E non vi sono dipendenze transitive o parziali.

Di conseguenza, **non sono necessarie ulteriori trasformazioni per la normalizzazione.**

Una struttura ben normalizzata **minimizza gli aggiornamenti multipli** dello stesso dato. **Ogni informazione è registrata in un solo punto.** La BCNF elimina tutte le **dipendenze funzionali non banali** non fondate su superchiavi. Questo riduce la duplicazione dei dati e evita anomalie durante le operazioni di inserimento, aggiornamento o cancellazione. Ci sono meno dati ridondanti da gestire e meno necessità di aggiornare dati in più punti, quindi operazioni più veloci e sicure.

PROGETTAZIONE FISICA

Si giunge ora al cuore pulsante della progettazione fisica del sistema informatico dedicato alla gestione bibliotecaria.

UTENTI E PRIVILEGI

Ogni utente che accede al sistema deve autenticarsi attraverso un modulo di **login**, composto da un **nome utente** e una **password**.

La gestione dei privilegi è basata sul **principio del minimo privilegio** (*Principle of Least Privilege*), il quale garantisce che ciascun attore possa eseguire **solo le operazioni strettamente necessarie** al proprio ruolo, contribuendo a mantenere la sicurezza, la coerenza e la tracciabilità delle operazioni nel sistema.

Sebbene non sia rappresentata nel modello concettuale E/R, è stata introdotta nella fase di progettazione fisica la tabella **Permessi**, finalizzata alla gestione delle credenziali di accesso. È importante sottolineare che la tabella **Permessi** è destinata esclusivamente alla gestione delle credenziali di **amministratori e bibliotecari**.

Gli **utenti finali**, ovvero coloro che effettuano richieste di prestito, **non sono tracciati** all'interno di questa tabella, in quanto il sistema prevede che ciascun utente possa avere **in prestito una sola copia per volta**.

Pertanto, le informazioni relative agli utenti comuni vengono gestite esclusivamente attraverso la tabella **Prestito**, nella quale sono registrati i dati anagrafici necessari al momento della richiesta, senza l'impiego di un meccanismo di login permanente.

Gli attori coinvolti nella struttura informatica del sistema bibliotecario sono i seguenti:

- **Amministratore** (Admin);
- **Bibliotecario**;
- **Utente**;
- **Login**;

Nella precedente descrizione della tavola delle operazioni sono state considerate unicamente le **operazioni più significative dal punto di vista funzionale**, omettendo intenzionalmente quelle di **routine** (quali il login o la generazione di report) per ragioni di chiarezza e sintesi.

In questa sezione, invece, si procederà con una **esplicitazione completa delle operazioni** che ciascuna categoria di utente può effettuare all'interno del sistema, distinguendo i permessi in base al **ruolo assegnato**, così da definire in modo chiaro le **responsabilità operative** di amministratori, bibliotecari e utenti finali.

• **Amministratore.**

Operazioni Permesse.

L'amministratore, una volta autenticato nel sistema, ha accesso a una serie di operazioni di gestione ordinaria, tra cui:

- l'aggiunta di nuovo personale, sia esso un altro amministratore o un bibliotecario;
- la rimozione del personale già presente nel sistema;
- la dismissione manuale di un libro qualora risulti non essere stato prestato negli ultimi dieci anni;
- la generazione di report amministrativi, relativi ai bibliotecari e agli amministratori attualmente attivi nel circuito bibliotecario.
- L'eliminazione fisica del libro;

Privilegi.

Concessione del permesso di esecuzione per le operazioni di aggiunta del personale, eliminazione del personale, generazione di report e dismissione dei libri.

• **Bibliotecario.**

Operazioni Permesse.

Il bibliotecario, una volta autenticato nel sistema, ha accesso a una serie di operazioni specifiche in linea con il proprio ruolo operativo. In particolare, può:

- generare report relativi a tutti e soli i bibliotecari attualmente presenti nel circuito bibliotecario;
- consultare le informazioni sugli utenti che detengono una determinata copia di un libro, indipendentemente dal fatto che la copia sia interna alla propria biblioteca o proveniente da una biblioteca esterna;

- produrre report sulle disposizioni dei libri, limitatamente a quelli per cui risultano copie disponibili per il prestito;
- effettuare l'inserimento di nuove copie per libri già presenti nel catalogo;
- eseguire l'inserimento di nuovi libri, purché accompagnati da almeno una copia iniziale.

Privilegi.

Il ruolo bibliotecario dispone dei seguenti permessi di esecuzione sulle procedure definite nel database:

- aggiornamento_altrove: consente di aggiornare informazioni relative a copie prese in prestito da biblioteche esterne del circuito;
- aggiungi_copia: permette l'inserimento di una nuova copia per un libro già esistente nel catalogo;
- aggiungi_libro: consente l'inserimento di un nuovo libro nel sistema, con almeno una copia associata;
- controlla_utente: consente di ottenere informazioni sugli utenti in possesso di una determinata copia;
- controllo_circuito: permette di verificare la disponibilità di un libro presso le biblioteche appartenenti al circuito;
- disponibilita: restituisce la disponibilità di copie prestabili per un determinato libro;
- effettua_riconsegna: abilita il bibliotecario a registrare la restituzione di una copia da parte dell'utente;
- modifica_copiac: consentono la modifica delle informazioni relative a una copia (es. stato, posizione, disponibilità);
- mostra_categorie: permette la consultazione dell'elenco delle categorie bibliografiche presenti nel sistema;
- registra_libro_e_copia: consente di inserire contestualmente un nuovo libro e almeno una copia associata;
- report_completo: consente di generare un report complessivo relativo alla disponibilità e collocazione dei libri;
- report_personale: abilita alla generazione di report sui bibliotecari del circuito;
- report_posesso: fornisce un report sulle copie in possesso degli utenti (sia interne che in prestito interbibliotecario).

Il bibliotecario non dispone di privilegi diretti sulle singole tabelle del database, in quanto tutte le operazioni che può effettuare sono mediate da apposite procedure memorizzate. Questo approccio garantisce un maggiore controllo sull'accesso ai dati, poiché l'utente può interagire solo attraverso le procedure autorizzate, evitando modifiche dirette e incontrollate alle strutture dati.

L'interazione avviene esclusivamente tramite GRANT EXECUTE sulle stored procedure, in conformità con i principi di sicurezza e isolamento dei ruoli. Se cambia la struttura interna delle tabelle, non è necessario modificare i privilegi, **basta aggiornare la procedura.**

• Utente.

Operazioni Permesse.

L'utente non accede al sistema tramite un vero e proprio meccanismo di autenticazione. Il suo unico compito consiste nel richiedere una copia di un libro, sia essa disponibile nella biblioteca di riferimento o in una delle biblioteche esterne appartenenti al circuito.

Il sistema prevede tuttavia che un utente possa detenere una sola copia alla volta. Qualora l'utente tenti di richiedere una seconda copia, sarà necessario che dismetta prima la copia precedentemente assegnata, in modo da garantire la coerenza e il corretto tracciamento dei prestiti.

Privilegi.

Il ruolo utente è stato progettato per non disporre di privilegi diretti né sulle tabelle né sulle procedure del database.

La sua interazione con il sistema è limitata esclusivamente alla richiesta di una copia, sia essa disponibile nella biblioteca di riferimento o presso una delle biblioteche del circuito.

Non è previsto un meccanismo di login o di autenticazione formale: l'utente viene identificato al momento della richiesta tramite nome, cognome e data di nascita, e può detenere al massimo una copia alla volta.

Di conseguenza, la configurazione GRANT USAGE ON *.* TO 'utente'@'%' è sufficiente per abilitarne l'esistenza nel sistema senza attribuirgli privilegi operativi.

• Login.

L'utente con ruolo "Login" rappresenta la figura abilitata ad accedere all'applicazione mediante l'inserimento delle proprie credenziali di autenticazione. Attraverso questa modalità, l'utente può essere riconosciuto e registrato come Bibliotecario oppure Amministratore, in base al profilo selezionato o assegnato durante il processo di registrazione. Possiede il permesso di esecuzione della sola ed unica procedura di login.

STRUTTURE DI MEMORIZZAZIONE

| TABELLA <Libro> | | |
|------------------------------|---------------------|--|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| ISBN | VARCHAR(100) | PRIMARY KEY, NOT NULL |
| Titolo | VARCHAR(100) | NOT NULL |
| Codice_Categoria | VARCHAR(100) | FOREIGN KEY (Categoria), NOT NULL, ON DELETE RESTRICT, ON UPDATE RESTRICT. |
| Disponibile | VARCHAR(100) | NOT NULL |

| TABELLA <Copia> | | |
|------------------------------|---------------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Numero_Copia | INT | PRIMARY KEY, NOT NULL |
| Stato | ENUM | NOT NULL |
| ISBN | VARCHAR(100) | PRIMARY KEY, FOREIGN KEY (Libro), NOT NULL, ON DELETE CASCADE, ON UPDATE CASCADE. |

| TABELLA <Categoria> | | |
|----------------------------------|---------------------|-----------------------|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Codice_Categoria | VARCHAR(100) | PRIMARY KEY, NOT NULL |
| Nome | VARCHAR(100) | NOT NULL |

| TABELLA <DisponibilitaCopia> | | |
|---|---------------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| ISBN | VARCHAR(100) | PRIMARY KEY, FOREIGN KEY(Libro), NOT NULL, ON DELETE RESTRICT, ON UPDATE RESTRICT. |
| ID_Biblioteca | INT | PRIMARY KEY, FOREIGN KEY(Biblioteca), NOT NULL, ON DELETE CASCADE, ON UPDATE CASCADE. |
| Numero_Copie | INT | NOT NULL |

| TABELLA <Permessi> | | |
|--------------------|--------------|-----------------------|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Identificativo | INT | PRIMARY KEY, NOT NULL |
| Nome | VARCHAR(100) | NOT NULL |
| Cognome | VARCHAR(100) | NOT NULL |
| Password (MD5) | VARCHAR(100) | PRIMARY KEY, NOT NULL |
| Ruolo | VARCHAR(100) | NOT NULL |

| TABELLA <Prestito> | | |
|----------------------|--------------|--|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| ISBN | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Copia), ON DELETE RESTRICT, ON UPDATE RESTRICT |
| Numero_Copia | INT | PRIMARY KEY, NOT NULL, FOREIGN KEY(Copia), ON DELETE RESTRICT, ON UPDATE RESTRICT |
| Nome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE CASCADE |
| Cognome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE CASCADE. |
| Data_Nascita | DATE | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE CASCADE. |
| Data_Inizio_Prestito | DATE | NOT NULL |
| Data_Fine_Prestito | DATE | NOT NULL |

| TABELLA <Prestito_Altrove> | | |
|----------------------------|--------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| ISBN | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(DisponibilitaCopia), ON DELETE RESTRICT, ON UPDATE RESTRICT. |
| ID_Biblioteca | INT | PRIMARY KEY, NOT NULL, FOREIGN KEY(DisponibilitaCopia), ON DELETE RESTRICT, ON UPDATE RESTRICT. |
| Nome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE CASCADE. |
| Cognome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE CASCADE. |
| Data_Nascita | DATE | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE CASCADE. |
| Data_Inizio_Prestito | DATE | NOT NULL |
| Data_Fine_Prestito | DATE | NOT NULL |

| TABELLA <Ripiano> | | |
|-------------------|--------------|--|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Numero_Ripiano | INT | NOT NULL |
| Codice_Scaffale | INT | PRIMARY KEY, NOT NULL, FOREIGN KEY(Scaffale), ON DELETE CASCADE, ON UPDATE CASCADE. |
| ISBN | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Libro), ON DELETE CASCADE, ON UPDATE CASCADE. |

| TABELLA <Scaffale> | | |
|--------------------|--------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Codice_Scaffale | INT | PRIMARY KEY, NOT NULL. |
| Nome | VARCHAR(100) | NOT NULL |
| Codice_categoria | VARCHAR(100) | NOT NULL, FOREIGN KEY(Categoria), ON DELETE CASCADE, ON UPDATE CASCADE. |

| TABELLA <TrasferimentoCopiaInBiblio> | | |
|--------------------------------------|--------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| ISBN | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Copia), ON DELETE RESTRICT, ON UPDATE RESTRICT |
| Numero_Copia | INT | PRIMARY KEY, NOT NULL, FOREIGN KEY(Copia), ON DELETE RESTRICT, ON UPDATE RESTRICT |
| ID_Biblioteca | INT | PRIMARY KEY, NOT NULL, FOREIGN KEY(Biblioteca) ON DELETE CASCADE, ON UPDATE CASCADE |

| TABELLA <Utente> | | |
|------------------|--------------|-----------------------|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Nome | VARCHAR(100) | PRIMARY KEY, NOT NULL |
| Cognome | VARCHAR(100) | PRIMARY KEY, NOT NULL |
| Data_Nascita | DATE | PRIMARY KEY, NOT NULL |

| TABELLA <UtenteEmail> | | |
|-----------------------|--------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Nome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE RESTRICT |

| TABELLA <UtenteEmail> | | |
|-----------------------|--------------|--|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Cognome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE RESTRICT. |
| Data_Nascita | DATE | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE RESTRICT. |
| Email | VARCHAR(100) | NOT NULL |

| TABELLA <UtenteTelefono> | | |
|--------------------------|--------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| Nome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE RESTRICT |
| Cognome | VARCHAR(100) | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE RESTRICT |
| Data_Nascita | DATE | PRIMARY KEY, NOT NULL, FOREIGN KEY(Utente), ON DELETE CASCADE, ON UPDATE RESTRICT |
| Telefono | VARCHAR(100) | NOT NULL |

L'autore del libro non è stato inserito come attributo nella tabella **Libro**, poiché ogni utente effettua la ricerca del libro in base al titolo.

Ogni volta che viene effettuato un prestito, viene aggiornata una tabella denominata **Log_prestiti**, la quale ha lo scopo di tracciare lo storico dei prestiti di ciascun libro. In questo modo, è possibile verificare se un libro non è stato prestato negli ultimi 10 anni e, di conseguenza, procedere alla sua dismissione. (Dopo questo aspetto verrà approfondito).

| TABELLA <Log_Prestiti> | | |
|------------------------|--------------|---|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| ID | INT | PRIMARY KEY, NOT NULL, AUTO INCREMENT |
| ISBN | VARCHAR(100) | NOT NULL, FOREIGN KEY(Libro), ON DELETE CASCADE, ON UPDATE CASCADE. |
| Data | DATE | NOT NULL |

| TABELLA <Biblioteca> | | |
|----------------------|--------------|--|
| COLONNA | TIPO DI DATO | ATTRIBUTI |
| NOME | VARCHAR(100) | NOT NULL |
| CITTÀ | VARCHAR(100) | NOT NULL |
| ID_Biblioteca | INT | PRIMARY KEY, NOT NULL, AUTO INCREMENT. |

È giunto ora il momento di esaminare in quale modo sono stati strutturati gli attributi delle tabelle, nonché di comprendere il funzionamento complessivo di tutti i **vincoli di integrità referenziale**.

SE**Cancello un'istanza di Libro.****Considera che**

La cancellazione di una copia non è consentita se esiste ancora un prestito attivo che fa riferimento a quella copia.

Infatti, la tabella **Prestito** contiene gli attributi esterni ISBN e NumeroCopia, che sono chiavi esterne riferite alla tabella **Copia**, con la clausola **ON DELETE RESTRICT**.

Questo significa che una copia può essere eliminata solo se non esiste alcun **prestito** che la coinvolga.

Allora**Si cancella un'istanza di Copia.****E****Si cancella un'istanza di Ripiano.****E****Tutto lo storico dei prestiti di quel Libro.****Inoltre**

Una copia identificata da un determinato ISBN e numero di copia **non può essere eliminata** se risulta attualmente in **prestito presso un'altra biblioteca del circuito**.

A tal fine, viene verificata l'esistenza di un'istanza nella tabella **TrasferimentoCopiaBiblio** che faccia riferimento alla copia in questione.

La presenza di tale riferimento impedisce la cancellazione della copia, grazie all'uso del vincolo di integrità referenziale con clausola **ON DELETE RESTRICT**, che garantisce la coerenza dei dati e l'impossibilità di eliminare copie ancora coinvolte in trasferimenti attivi.

Ma**Non si cancella un'istanza di DisponibilitàCopia.**

Bisogna impedire che un libro venga eliminato finché esistono disponibilità che lo usano.

È possibile stabilire che, nel momento in cui un utente viene eliminato dalla tabella Utente, le relative informazioni presenti in una tabella specifica, ad esempio UtenteTelefono (o email), vengano eliminate automaticamente.

Questo comportamento è reso possibile grazie all'uso di un vincolo di chiave esterna definito con la clausola **ON DELETE CASCADE**, che assicura la propagazione automatica della cancellazione ai record correlati. In questo modo si mantiene l'integrità referenziale evitando la presenza di dati orfani nel sistema.

Mentre la cancellazione della categoria fallisce indirettamente a causa della restrizione sulla disponibilità delle copie. Sebbene la cancellazione di una categoria comporti la rimozione in cascata dei libri associati, l'operazione viene bloccata se un libro fa riferimento a quella specifica categoria.

Nota che anche la copia del libro non viene eliminata.

Quando un utente viene eliminato dalla tabella Utente, tutti i record associati nelle tabelle dei prestiti vengono automaticamente rimossi.

Questo comportamento è ottenuto grazie all'utilizzo della clausola ON DELETE CASCADE nei vincoli di chiave esterna, che consente la cancellazione automatica delle entità correlate, liberando completamente i vincoli e garantendo la coerenza referenziale all'interno del database.

Prestito_Altrove **referenzia DisponibilitaCopia con ON DELETE RESTRICT.**

La tabella Prestito_Altrove contiene:

- ISBN e ID_Biblioteca
- che insieme sono una **chiave esterna** verso la tabella DisponibilitaCopia.

Non è possibile eliminare una riga da DisponibilitaCopia (cioè una disponibilità di una certa copia in una certa biblioteca) **se esiste un prestito attivo in Prestito_Altrove che la utilizza.**

In altre parole: Se un utente ha preso in prestito **altrove** una copia disponibile in una certa biblioteca, quella disponibilità non può essere eliminata dalla tabella DisponibilitaCopia finché il prestito è ancora registrato in Prestito_Altrove.

Tuttavia, se elimino una biblioteca, si eliminano tutti i record di TrasferimentoCopiaInBiblio e di DisponibilitaCopia che fanno riferimento a quella biblioteca.

INDICI.

Gli indici sono totalmente gestiti dal DBMS.

Il motore del database decide automaticamente quando e come usarli, in base alla query.
CREATE INDEX è un indice secondario, quindi UNCLUSTERED.

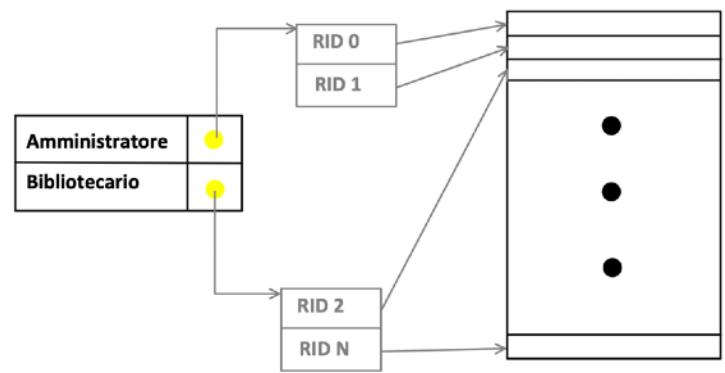
Una prima indicizzazione di tipo **unclustered** è stata applicata sulla tabella Permessi, in particolare sul campo Ruolo, al fine di ottimizzare le query che effettuano report filtrati per tipologia di ruolo (es. Bibliotecario o Amministratore).

Questa scelta è giustificata dal fatto che l'amministratore ha la necessità di consultare l'elenco dei bibliotecari e degli altri amministratori presenti nel circuito, e quindi vengono eseguite query selettive basate proprio su questo attributo. Il nome della colonna "Ruolo" non cambierà mai nome, quindi l'indice permane.

CREATE INDEX idx_permessi_ruolo ON BNCR.Permessi(Ruolo);

Tuttavia, attualmente la tabella “Permessi” è di dimensioni ridotte, per cui potrebbe risultare più efficiente una scansione completa.

Tuttavia, in uno scenario in cui la rete bibliotecaria sia composta da numerose biblioteche con un elevato numero di membri del personale, e quindi la tabella dei permessi cresca sensibilmente, l’utilizzo di un’indicizzazione unclustered può contribuire a migliorare le performance. La scelta dell’indice è stata presa consapevolmente.



Se un Indice Secondario (Non-Clustered) è costruito su una colonna con chiavi duplicate (es. Ruolo), il DBMS utilizza una lista di puntatori (RID) per collegare più record a un unico valore della chiave di ricerca.

Analizziamo ora l’operazione di report relativa al possesso, con particolare attenzione alla distinzione tra chi detiene una copia di un libro internamente e chi la detiene esternamente. Tali informazioni vengono restituite sotto forma di due insiemi di risultati distinti. È importante considerare che il report relativo al possesso dei libri viene eseguito con elevata frequenza nella pratica operativa, in quanto risulta sempre utile conoscere chi detiene una determinata copia di un libro oppure chi ha effettuato un prestito presso altre biblioteche del circuito.

In riferimento all’**Operazione OP01** precedentemente descritta, si evidenzia che il numero di prestiti da gestire è considerevole.

Di conseguenza, la base dati coinvolta è di dimensioni rilevanti, soggetta a frequenti aggiornamenti e consultazioni, in particolare tramite report utilizzati con regolarità.

Per quanto concerne il report relativo ai prestiti interni che risultano essere significativamente più numerosi rispetto ai prestiti esterni si è scelto di ottimizzarne la gestione mediante l’utilizzo di un indice solo.

CREATE VIEW BNCR.report_possesto_esterno AS**SELECT**

```

BNCR.Prestito_Altrove.Nome,
BNCR.Prestito_Altrove.Cognome,
BNCR.Libro.Titolo,
Contatti.Data_Nascita,
BNCR.Libro.ISBN,
Contatti.Contatto,
Contatti.Tipo_Contatto,
BNCR.Biblioteca.Nome AS NomeBiblio,
BNCR.Biblioteca.Citta
FROM BNCR.Prestito_Altrove, BNCR.Libro, BNCR.DisponibilitaCopia, BNCR.Biblioteca,
(SELECT BNCR.UtenteEmail.Nome, BNCR.UtenteEmail.Cognome, BNCR.UtenteEmail.Data_Nascita,
BNCR.UtenteEmail.Email AS Contatto, 'Email' AS Tipo_Contatto
FROM BNCR.UtenteEmail
UNION
SELECT BNCR.UtenteTelefono.Nome, BNCR.UtenteTelefono.Cognome, BNCR.UtenteTelefono.Data_Nascita,
BNCR.UtenteTelefono.Telefono AS Contatto, 'Telefono' AS Tipo_Contatto
FROM BNCR.UtenteTelefono) AS Contatti
WHERE BNCR.Prestito_Altrove.ISBN = BNCR.Libro.ISBN
AND BNCR.Prestito_Altrove.Nome = Contatti.Nome
AND BNCR.Prestito_Altrove.Cognome = Contatti.Cognome
AND BNCR.Prestito_Altrove.Data_Nascita = Contatti.Data_Nascita
AND BNCR.DisponibilitaCopia.ISBN = BNCR.Prestito_Altrove.ISBN
AND BNCR.DisponibilitaCopia.ID_Biblioteca = BNCR.Biblioteca.ID_Biblioteca
AND BNCR.DisponibilitaCopia.ID_Biblioteca = BNCR.Prestito_Altrove.ID_Biblioteca
AND BNCR.Biblioteca.ID_Biblioteca = BNCR.Prestito_Altrove.ID_Biblioteca;

```

END

È stata inoltre creata una vista dedicata ai prestiti interni, denominata report_possesto_interno, al fine di distinguerla chiaramente dal report relativo ai prestiti esterni.

CREATE VIEW BNCR.report_possesto_interno AS**SELECT**

```

BNCR.Prestito.NumeroCopia,
BNCR.Prestito.Nome,
BNCR.Prestito.Cognome,
BNCR.Libro.Titolo,
Contatti.Data_Nascita,
BNCR.Libro.ISBN,
Contatti.Contatto,
Contatti.Tipo_Contatto
FROM BNCR.Prestito, BNCR.Libro,
(SELECT BNCR.UtenteEmail.Nome, BNCR.UtenteEmail.Cognome, BNCR.UtenteEmail.Data_Nascita,
BNCR.UtenteEmail.Email AS Contatto, 'Email' AS Tipo_Contatto
FROM BNCR.UtenteEmail
UNION
SELECT BNCR.UtenteTelefono.Nome, BNCR.UtenteTelefono.Cognome, BNCR.UtenteTelefono.Data_Nascita.

```

Si segnala che questa vista viene richiamata all'interno della stored procedure report_posesso, relativa alla gestione dei prestiti interni.

È importante sottolineare che si è scelto di non utilizzare una vista materializzata, in quanto il report sui dati viene aggiornato con elevata frequenza e non si intende tollerare eventuali ritardi derivanti da operazioni di refresh programmato.

Quindi, in sostanza tutti gli indici necessari sono già presenti grazie alle PRIMARY KEY e non serve creare svariati indici manualmente per questa query.

In "Prestito": ISBN, Nome, Cognome, Data_Nascita sono già indicizzati in quanto sono PK.
In "Libro": ISBN è già indicizzato in quanto è PK.

In "UtenteEmail" e "UtenteTelefono": Nome, Cognome, Data_Nascita sono già indicizzati in quanto sono PK.

L'unico indice che si è ritenuto opportuno mantenere è **idx_prestito_utente** su (Nome, Cognome, Data_Nascita), in quanto vengono eseguite frequentemente interrogazioni con clausole WHERE che filtrano i dati proprio su tali attributi. Sebbene le stesse colonne siano incluse nella chiave primaria composta della tabella Prestito, è importante sottolineare che un indice viene utilizzato in modo efficace dal motore di interrogazione solo se le colonne specificate nella condizione di ricerca compaiono all'inizio della definizione dell'indice. Pertanto, l'indice derivante dalla PRIMARY KEY (che include ulteriori colonne precedenti) non risulta sfruttabile per queste specifiche interrogazioni.

CREATE INDEX idx_prestito_utente ON BNCR.Prestito(Nome, Cognome, Data_Nascita);

Nel contesto della gestione dei prestiti esterni, la tabella DisponibilitaCopia dispone di un indice sulla chiave esterna ID_Biblioteca, denominato fk_disp_biblioteca, il quale supporta efficientemente le operazioni di ricerca basate su tale attributo. Tuttavia, nella tabella Prestito_Altrove, le ricerche vengono effettuate anch'esse frequentemente su ID_Biblioteca, ma quest'ultimo risulta posizionato successivamente all'attributo ISBN all'interno della chiave primaria composta.

Per tale motivo, al fine di ottimizzare l'accesso diretto ai dati tramite ID_Biblioteca, si è scelto di definire un indice dedicato su tale colonna, così da migliorare le prestazioni nelle operazioni di join e filtraggio che coinvolgono questo campo specifico.

CREATE INDEX idx_prestito_altrove_biblioteca ON BNCR.Prestito_Altrove(ID_Biblioteca);

È opportuno segnalare che, qualora un libro non risulti disponibile presso la biblioteca interna, viene eseguito un controllo sull'intero circuito bibliotecario (vedi query successiva). Nella maggior parte dei casi, all'interno del circuito, qualora la copia venga effettivamente reperita, il numero di copie disponibili presso una biblioteca diversa risulta essere quasi sempre superiore a zero.

Alla luce di ciò, non appare giustificata l'introduzione di un indice sulla colonna numero_copie della tabella DisponibilitaCopia, poiché, nella stored procedure controllo_circuito, la ricerca su tale attributo risulterebbe scarsamente selettiva. In simili circostanze, il sistema di gestione del database potrebbe scegliere di ignorare l'indice e procedere con una scansione completa della tabella (full table scan), considerandola un'alternativa più efficiente dal punto di vista delle prestazioni:

```
SELECT BNCR.DisponibilitaCopia.ISBN, BNCR.Libro.Titolo,
BNCR.DisponibilitaCopia.numero_copie, BNCR.Biblioteca.Nome, BNCR.Biblioteca.Citta
FROM BNCR.DisponibilitaCopia, BNCR.Libro, BNCR.Biblioteca
WHERE BNCR.DisponibilitaCopia.ISBN = BNCR.Libro.ISBN
AND BNCR.Biblioteca.ID_Biblioteca =
BNCR.DisponibilitaCopia.ID_Biblioteca
AND BNCR.Libro.Titolo=titolo_var AND
BNCR.DisponibilitaCopia.numero_copie > 0;
```

È stata inoltre implementata una vista dedicata ai report dei prestiti esterni, successivamente integrata all'interno del report complessivo di possesso.

Viene ora analizzata la funzione incaricata della generazione del report_completo, relativa alla disposizione dei libri all'interno della biblioteca del minimondo. Nella stored procedure **report_completo**, la query prevede un'operazione di JOIN tra le tabelle Libro, Scaffale, Ripiano e Copia, al fine di ricostruire in maniera dettagliata la collocazione fisica dei volumi.

In tale contesto, l'aggiunta automatica di un indice sulla colonna Codice_Categoria della tabella Scaffale può apportare benefici in termini prestazionali, dal momento che la query prevede una clausola WHERE su tale campo. Inoltre, al fine di ottimizzare ulteriormente le prestazioni in fase di lettura, è stato introdotto un indice anche sulla colonna Stato della tabella Copia.

```
CREATE INDEX idx_copia_stato ON BNCR.Copia(Stato);
```

È stata definita una View anche per questo report.

CREATE VIEW BNCR.Disponi AS

```
select DISTINCT BNCR.Libro.Titolo, BNCR.Libro.ISBN, BNCR.Scaffale.Codice_Scaffale, BNCR.Scaffale.Nome,
BNCR.Ripiano.Numero_Ripiano
from BNCR.Libro, BNCR.Scaffale, BNCR.Ripiano, BNCR.Copia
where BNCR.Libro.Codice_categoria = BNCR.Scaffale.Codice_categoria
and BNCR.Ripiano.Codice_Scaffale = BNCR.Scaffale.Codice_Scaffale
and BNCR.Ripiano.ISBN = BNCR.Libro.ISBN
and BNCR.Libro.ISBN = BNCR.Copia.ISBN
and BNCR.Copia.stato = 'Disponibile'
and BNCR.Libro.Disponibile='si';
```

L'indice su Stato consente di saltare tutte le righe non rilevanti, evitando uno scan completo della tabella.

Ma attenzione: dipende dalla configurazione attuale della tabella. Se in un momento ho più della metà delle righe che hanno stato='Disponibile' il DB potrebbe NON usare l'indice, perché leggerà comunque quasi tutta la tabella.

Inoltre è stato creato l'indice sul seguente campo:

CREATE INDEX idx_libro_disponibile ON BNCR.Libro(Disponibile);

Va osservato che quanto detto per l'attributo Stato della tabella Copia è applicabile anche all'indice idx_libro_disponibile, creato sulla colonna Disponibile della tabella Libro. In presenza di una distribuzione fortemente sbilanciata dei valori ad esempio, se la maggior parte dei libri risulta ancora disponibile e solo una minima parte è stata dismessa il DBMS potrebbe decidere di non utilizzare tale indice, optando invece per una scansione completa della tabella.

Tuttavia, è opportuno specificare che l'indice potrebbe rivelarsi efficace in contesti specifici, come ad esempio nell'ambito di un report dedicato ai libri dismessi. In questo caso, poiché si presume che il numero di libri dismessi sia significativamente inferiore rispetto al totale, il filtro applicato nella clausola WHERE risulterebbe selettivo e l'indice verrebbe sfruttato dal motore di interrogazione per ottimizzare l'accesso ai dati.

Va osservato che l'operazione di aggiunta di una nuova copia relativa a un determinato libro interno alla biblioteca comporta, preliminarmente, un controllo sull'esistenza del libro stesso. Trattandosi di un'operazione effettuata con frequenza, risulta opportuno prevedere la creazione di un indice sulla colonna Titolo della tabella Libro, al fine di ottimizzare le prestazioni della query di verifica e migliorare l'efficienza complessiva dell'inserimento. Questo si ripercuote nella stored procedure denominata Aggiungi_Copia.

CREATE INDEX idx_libro_titolo ON BNCR.Libro(Titolo);

Considerato che il numero di biblioteche appartenenti al circuito risulta essere contenuto, si è scelto di non introdurre alcun indice secondario sulle colonne Nome e Città della tabella Biblioteca, ritenendo che il volume ridotto dei dati non giustifichi la necessità di un'ottimizzazione tramite indicizzazione.

Va inoltre sottolineato che la presenza di un indice composito sulla tabella Utente, generato automaticamente dal DBMS in corrispondenza della chiave primaria composta (Nome, Cognome, Data_Nascita), costituisce un importante fattore di ottimizzazione. Tale struttura consente al motore di interrogazione di eseguire le ricerche su queste colonne in modo efficiente, sfruttando direttamente l'indice senza ricorrere a costose operazioni di scansione completa della tabella. Questa scelta progettuale risulta particolarmente rilevante alla luce del volume di richieste previste: le operazioni precedentemente descritte, come OP01 e OP06, registrano complessivamente una frequenza attesa di circa 230 accessi giornalieri. Ne consegue che il numero di utenti registrati è significativo, con la possibilità concreta che si verifichino omonimie. In tali casi, l'indice consente di raggruppare i record per Nome, offrendo una struttura a lista puntata che facilita il recupero rapido delle informazioni associate.

È vero che tale meccanismo comporta un incremento dell'utilizzo di memoria, ma, in questo scenario, si ritiene preferibile accettare tale onere in favore di una maggiore efficienza, piuttosto che incorrere in una full table scan nella gestione di operazioni eseguite con elevata frequenza.

Anche nel caso della tabella Prestito_Altrove, all'atto della definizione della chiave esterna su (Nome, Cognome, Data_Nascita), il DBMS ha provveduto automaticamente alla creazione di un indice specifico su tali colonne. Tale indice consente di accelerare le operazioni di ricerca e confronto che coinvolgono l'identificazione dell'utente, ottimizzando le performance nelle operazioni di join o nelle interrogazioni basate su tali attributi.

Infine, al fine di ottimizzare le operazioni di ricerca basate sull'ISBN, si è ritenuto opportuno introdurre un indice su tale colonna sia nella tabella Prestito che in Prestito_Altrove. L'utilizzo dell'ISBN come chiave di accesso si configura come un'operazione particolarmente frequente, in quanto coinvolta non solo nella fase di prestito, ma soprattutto nel processo di riconsegna: quest'ultimo richiede l'individuazione puntuale della tupla relativa al prestito mediante la ricerca dell'ISBN corrispondente. Considerata la possibile elevata numerosità di ISBN registrati all'interno di tali tabelle, la presenza dell'indice risulta determinante per garantire prestazioni ottimali nelle interrogazioni ricorrenti.

CREATE INDEX idx_prestito_isbn ON BNCR.Prestito(ISBN);
CREATE INDEX idx_prestito_altrove_isbn ON BNCR.Prestito_Altrove(ISBN);

L'introduzione di indici dedicati sulla colonna ISBN nelle tabelle Prestito e Prestito_Altrove è stata prevista nell'ottica di un contesto applicativo in cui i prestiti raggiungano volumi considerevoli, nell'ordine di migliaia di registrazioni. Tuttavia, è opportuno precisare che, ai fini di un'applicazione a scopo didattico o in presenza di un numero limitato di prestiti, tali indici risultano ridondanti. Infatti, essendo ISBN già definito come chiave primaria in entrambe le tabelle, il DBMS provvede automaticamente alla creazione di un indice univoco, rendendo superflua l'aggiunta manuale di un ulteriore indice sulla medesima colonna.

Inoltre, in ambienti con un volume dati ridotto, mantenere indici non strettamente necessari comporta un inutile consumo di memoria e risorse, senza apportare benefici tangibili in termini di prestazioni. È pertanto fondamentale sottolineare che l'aggiunta di tali indici è stata effettuata consapevolmente, con l'obiettivo di simulare uno scenario realistico in cui l'ottimizzazione delle interrogazioni diventa rilevante a fronte dell'aumento del carico dati.

Si è adottata anche la vista sul report di tutte le categorie:

```
CREATE VIEW AS BNCR.categorie AS
select BNCR.Categoria.Codice_categoria, BNCR.Categoria.nome
from Categoria;
```

Di seguito uno schema riassuntivo degli indici:

| TABELLA<Libro> | |
|---|-------------|
| Indice <PRIMARY> | Tipo |
| ISBN | Primario. |
| Indice <idx_libro_disponibile> | Tipo |
| Disponibile | Secondario |
| Indice <idx_libro_titolo> | Tipo |
| Titolo | Secondario |
| Indice <Libro_Categoria_FK> | Tipo |
| Codice_categoria | Secondario |

| TABELLA<Scaffale> | |
|---|-------------------------|
| Indice <PRIMARY> | Tipo |
| Codice_Scaffale | Primario. |
| Indice <Scaffale_Categoria_FK> | Tipo |
| Codice_categoria | Secondario (Automatico) |

| TABELLA<Ripiano> | |
|---------------------------------------|-------------|
| Indice <PRIMARY> | Tipo |
| (ISBN, Codice_Scaffale) | Primario. |
| Indice <Codice_Scaffale> | Tipo |
| Codice_Scaffale | Secondario |

| TABELLA<Prestito> | |
|--|-------------|
| Indice <PRIMARY> | Tipo |
| (ISBN, NumeroCopia, Nome, Cognome, Data_Nascita) | Primario. |
| Indice <idx_prestito_isbn> | Tipo |
| ISBN | Secondario |
| Indice <idx_prestito_utente> | Tipo |
| (Nome,Cognome,Data_Nascita) | Secondario |

| TABELLA<Prestito_Altrove> | |
|---|-------------|
| Indice <PRIMARY> | Tipo |
| (ISBN, ID_Biblioteca, Nome, Cognome, Data_Nascita) | Primario. |
| Indice <idx_prestito_altrove_biblioteca> | Tipo |
| ID_Biblioteca | Secondario |
| Indice <idx_prestito_altrove_isbn> | Tipo |

| TABELLA<Prestito_Altrove> | |
|--|-------------------------|
| Indice <PRIMARY> | Tipo |
| ISBN | Secondario |
| Indice <fk_prestito_utenteee> | Tipo |
| (Nome, Cognome, Data_Nascita) | Secondario (Automatico) |

| TABELLA<Permessi> | |
|--|-------------|
| Indice <PRIMARY> | Tipo |
| Identificativo | Primario. |
| Indice <idx_permessi_ruolo> | Tipo |
| Ruolo | Secondario |

| TABELLA<Copia> | |
|---------------------------------------|-------------------------|
| Indice <PRIMARY> | Tipo |
| (Numero_Copia,ISBN) | Primario. |
| Indice <idx_copia_stato> | Tipo |
| stato | Secondario |
| Indice <copia_ibfk_1> | Tipo |
| ISBN | Secondario (Automatico) |

(Sono state riportate solo le tabelle dove sono stati aggiunti indici manualmente).

TRIGGER

Trigger_1: Inserimento_corretto.

```

CREATE DEFINER=`root`@`localhost` TRIGGER `Inserimento_corretto` BEFORE
INSERT ON `permessi` FOR EACH ROW BEGIN
    DECLARE buffer_nome VARCHAR(100);
    DECLARE buffer_cognome VARCHAR(100);

    -- Controlla se il nome e il cognome già esistono nella tabella Permessi
    SELECT Nome, Cognome INTO buffer_nome, buffer_cognome
    FROM BNCR.Permessi
    WHERE Nome = NEW.Nome AND Cognome = NEW.Cognome
    LIMIT 1;

    -- Se il record esiste già, genera un errore
    IF buffer_nome IS NOT NULL AND buffer_cognome IS NOT NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Nome e cognome già esistenti';
    END IF;
END

```

Questo trigger, attivato prima di ogni inserimento nella tabella Permessi, verifica se esiste già un record con lo stesso nome e cognome. In tal caso, impedisce l'inserimento e genera un errore personalizzato.

Ciò si rende necessario in quanto l'amministratore potrebbe voler inserire un nuovo bibliotecario o amministratore all'interno della tabella Permessi. In tale contesto, il controllo preventivo garantisce l'unicità delle anagrafiche già registrate, evitando duplicazioni indesiderate.

Trigger_2: aggiornamento_Log_Prestiti.

```

CREATE DEFINER=`root`@`localhost` TRIGGER `aggiornamento_Log_Prestiti` AFTER
INSERT ON `prestito` FOR EACH ROW BEGIN
    INSERT INTO BNCR.Log_Prestiti (ISBN, `Data`)
    VALUES (NEW.ISBN, NEW.DataInizioPrestito);
END

```

Il trigger aggiornamento_Log_Prestiti viene eseguito automaticamente dopo ogni inserimento nella tabella Prestito.

La clausola AFTER INSERT garantisce che l'operazione si attivi solo dopo che la tupla è stata effettivamente inserita.

Questo trigger è stato implementato con l'obiettivo di mantenere uno storico dei prestiti associati a ciascun libro, identificato tramite il codice ISBN. La registrazione automatica di ogni nuovo prestito nella tabella Log_Prestiti consente di ricostruire, a posteriori, la cronologia dei prestiti effettuati per un determinato volume. Tale meccanismo si rivela particolarmente utile per analizzare, ad esempio, se un libro sia stato effettivamente consultato o prestato negli ultimi dieci anni, al fine di valutarne l'eventuale dismissione dal catalogo.

Trigger_3: controlli_libro.

```

CREATE DEFINER=`root`@`localhost` TRIGGER `controlli_libro` BEFORE DELETE ON
`Libro` FOR EACH ROW BEGIN

    declare var1 int;
    declare var2 int;
    declare var3 int;
    declare msg TEXT DEFAULT '';

    SELECT count(*) into var1
    FROM BNCR.Copia
    WHERE BNCR.Copia.ISBN=OLD.ISBN AND BNCR.Copia.stato='In prestito
utente';

    SELECT count(*) into var2
    FROM BNCR.Copia
    WHERE BNCR.Copia.ISBN=OLD.ISBN AND BNCR.Copia.stato='In prestito altra
biblioteca';

    SELECT count(*) into var3
    FROM BNCR.DisponibilitaCopia
    WHERE BNCR.DisponibilitaCopia.ISBN=OLD.ISBN AND
BNCR.DisponibilitaCopia.numero_copie > 0;

    if var1 > 0 then
        SET msg=CONCAT(msg, 'Libro in prestito presso utente!');
    end if;

    if var2 > 0 then
        SET msg=CONCAT(msg, 'Libro in prestito presso biblioteche!');
    end if;

    if var3 > 0 then
        SET msg=CONCAT(msg, 'Il libro è nel circuito');
    end if;

    if msg!="" THEN
        SIGNAL SQLSTATE '45988' SET MESSAGE_TEXT=msg;
    end if;
END

```

Il trigger controlli_libro viene attivato prima della cancellazione di un libro dalla tabella Libro e verifica tre condizioni di coerenza:

1. Se almeno una copia del libro risulta in prestito presso un utente.
2. Se almeno una copia è in prestito presso un'altra biblioteca.
3. Se il libro risulta ancora disponibile nel circuito bibliotecario tramite altre sedi.

In presenza di una o più di queste condizioni, l'operazione di cancellazione viene interrotta.

Trigger_4:controllo_copia_esemplare.

```
CREATE DEFINER='root'@'localhost' TRIGGER `controllo_copia_esemplare` BEFORE
DELETE ON `copia` FOR EACH ROW BEGIN

declare var int;
select count(*) into var
from BNCR.Copia
where BNCR.Copia.Numero_Copia=OLD.Numero_Copia and
BNCR.Copia.ISBN=OLD.ISBN and BNCR.Copia.stato!='Disponibile';

if var > 0 then
SIGNAL SQLSTATE '45999' SET MESSAGE_TEXT='Non puoi eliminare una copia se
questa è in trasferimento presso utente o biblioteca';
end if;
END
```

Il trigger controllo_copia_esemplare viene attivato prima dell'eliminazione di una riga dalla tabella Copia e verifica che la copia interessata non sia attualmente coinvolta in operazioni di prestito o trasferimento.

Trigger_5:aggiorna_copia.

```
CREATE DEFINER='root'@'localhost' TRIGGER `aggiorna_copia` AFTER INSERT ON
`prestito` FOR EACH ROW BEGIN

UPDATE BNCR.Copia
SET BNCR.Copia.stato = 'In prestito utente'
WHERE BNCR.Copia.Numero_Copia = NEW.NumeroCopia
and BNCR.Copia.ISBN = NEW.ISBN;

END
```

Il trigger aggiorna_copia viene eseguito dopo l'inserimento di un nuovo prestito.

Esso aggiorna automaticamente la tabella Copia, modificando lo stato della copia coinvolta in 'In prestito utente', per riflettere correttamente l'avvenuta assegnazione della copia a un utente.

Trigger_6:riconsegna_copia.

```
CREATE DEFINER='root'@'localhost' TRIGGER `riconsegna_copia` AFTER DELETE ON `prestito` FOR EACH ROW BEGIN

    UPDATE BNCR.Copia
    SET BNCR.Copia.stato = 'Disponibile'
    WHERE BNCR.Copia.Numero_Copia = OLD.NumeroCopia
    and BNCR.Copia.ISBN = OLD.ISBN;

END
```

Il trigger riconsegna_copia viene attivato dopo l'eliminazione di un record dalla tabella Prestito. Esso aggiorna automaticamente la tabella Copia, impostando lo stato della copia corrispondente a 'Disponibile', per riflettere il fatto che la copia è stata restituita e può essere nuovamente prestata.

Trigger_7:reg_prestito_esterno.

```
CREATE DEFINER='root'@'localhost' TRIGGER `reg_prestito_esterno` AFTER INSERT ON `prestito_altrove` FOR EACH ROW BEGIN

    UPDATE BNCR.DisponibilitaCopia
    SET BNCR.DisponibilitaCopia.numero_copie =
BNCR.DisponibilitaCopia.numero_copie - 1
    WHERE BNCR.DisponibilitaCopia.ISBN=NEW.ISBN
    AND BNCR.DisponibilitaCopia.ID_Biblioteca = NEW.ID_Biblioteca;

END
```

Questo trigger viene eseguito automaticamente dopo l'inserimento di una nuova riga nella tabella Prestito_Altrove.

(cioè: quando un utente prende in prestito un libro da un'altra biblioteca del circuito).

Aggiorna la tabella DisponibilitaCopia, sottraendo 1 copia disponibile dalla biblioteca che ha prestato il libro.

Trigger_8:riconsegna_prestito_esterno.

```
CREATE DEFINER='root'@'localhost' TRIGGER `riconsegna_prestito_esterno` AFTER DELETE ON `prestito_altrove` FOR EACH ROW BEGIN

    UPDATE BNCR.DisponibilitaCopia
    SET BNCR.DisponibilitaCopia.numero_copie =
BNCR.DisponibilitaCopia.numero_copie + 1
    WHERE BNCR.DisponibilitaCopia.ISBN=OLD.ISBN
    AND BNCR.DisponibilitaCopia.ID_Biblioteca = OLD.ID_Biblioteca;

END
```

Il trigger riconsegna_prestito_esterno è attivato automaticamente dopo l'eliminazione di un record dalla tabella Prestito_Altrove.

Tale evento rappresenta la restituzione di un libro precedentemente richiesto da un utente a una biblioteca diversa da quella locale.

Il trigger ha la funzione di ripristinare la disponibilità del libro presso la biblioteca prestante, incrementando di una unità il campo numero_copie nella tabella DisponibilitaCopia, in corrispondenza del libro (ISBN) e della biblioteca (ID_Biblioteca) coinvolti nel prestito.

In questo modo si mantiene coerenza tra il numero effettivo di copie disponibili nel circuito bibliotecario e le operazioni di prestito e restituzione effettuate, automatizzando l'aggiornamento dei dati.

Trigger_9:richiesta_da_biblioteca_di_copia.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `richiesta_da_biblioteca_di_copia`
AFTER INSERT ON `trasferimentocopiainbiblio` FOR EACH ROW BEGIN
```

```
UPDATE BNCR.Copia
SET stato = 'In prestito altra biblioteca'
WHERE Numero_Copia = NEW.Numero_Copia AND ISBN = NEW.ISBN;
```

END

Il trigger richiesta_da_biblioteca_di_copia viene attivato automaticamente dopo l'inserimento di un record nella tabella TrasferimentoCopiaInBiblio.

Tale tabella rappresenta una richiesta di trasferimento di una copia fisica di un libro da una biblioteca a un'altra nel circuito bibliotecario.

Quando viene registrata una richiesta di questo tipo, il trigger si occupa di aggiornare lo stato della copia coinvolta, nella tabella Copia, impostandolo a: "In prestito altra biblioteca"

Questa operazione automatica consente di riflettere in tempo reale la situazione della copia, indicando che essa è attualmente fuori sede e non disponibile per altri prestiti, poiché impegnata in un trasferimento inter-bibliotecario.

Trigger_10:blocca_eliminazione_ripiano.

```

CREATE DEFINER='root'@'localhost' TRIGGER `blocca_eliminazione_ripiano`
BEFORE DELETE ON `Ripiano`
FOR EACH ROW
BEGIN

    declare var int;

    select count(*) into var
    from BNCR.Ripiano
    where BNCR.Ripiano.ISBN=OLD.ISBN and
BNCR.Ripiano.Numero_Ripiano=OLD.Numero_Ripiano
    and BNCR.Ripiano.Codice_Scaffale=OLD.Codice_Scaffale;

    if var > 0 then
        SIGNAL SQLSTATE '45956' SET MESSAGE_TEXT='Non puoi eliminare una tupla
che rappresenta la disposizione';
        end if;

END

```

Il trigger blocca_eliminazione_ripiano viene eseguito prima dell'operazione di cancellazione (BEFORE DELETE) su un record della tabella Ripiano, che rappresenta la disposizione fisica di un libro all'interno di uno scaffale in biblioteca. Il suo scopo è impedire l'eliminazione di una riga se essa rappresenta una disposizione registrata nel sistema.

Si osserva che, nella forma attuale, il trigger rileva anche la riga stessa in fase di eliminazione, risultando in un blocco sistematico dell'operazione di DELETE. Questo comportamento può essere considerato intenzionale per impedire qualsiasi tipo di eliminazione diretta di disposizioni registrate.

Trigger_11:reimposta_valore_disponibile_da_trasferimento.

```

CREATE DEFINER='root'@'localhost' TRIGGER
`reimposta_valore_disponibile_da_trasferimento` AFTER DELETE ON
`trasferimentocopiainbiblio` FOR EACH ROW BEGIN

    UPDATE BNCR.Copia
    SET BNCR.Copia.stato='Disponibile'
    WHERE BNCR.Copia.ISBN=OLD.ISBN AND
BNCR.Copia.Numero_Copia=OLD.Numero_Copia;
END

```

Il trigger reimposta_valore_disponibile_da_trasferimento viene attivato automaticamente a seguito dell'eliminazione di una riga dalla tabella TrasferimentoCopiaInBiblio, che rappresenta il termine di una fase di prestito o trasferimento inter-bibliotecario di una copia di libro.

In particolare, il trigger si occupa di ripristinare lo stato della copia coinvolta, aggiornando il relativo campo stato nella tabella Copia, impostandolo a:

Disponibile.

Questo consente al sistema di segnalare correttamente che la copia è nuovamente presente e utilizzabile nella biblioteca di origine, assicurando così la coerenza tra lo stato fisico delle copie e le operazioni gestionali del circuito bibliotecario.

Il trigger agisce sulla copia identificata tramite l'ISBN e il Numero_Copia precedentemente presenti nella riga cancellata di TrasferimentoCopiaInBiblio.

EVENTI

Per simulare il comportamento di una biblioteca del circuito che richiede in prestito una copia di un libro non ancora presente nel proprio inventario, si è scelto di utilizzare un evento pianificato (scheduled event). Tale evento agisce con una frequenza di esecuzione regolare (nell'esempio implementativo, ogni 10 minuti) e ha la funzione di identificare copie disponibili presso la biblioteca denominata "Minimondo", selezionandone alcune che non risultano ancora presenti tra le disponibilità. A seguito dell'individuazione, l'evento simula una richiesta automatica di trasferimento di tali copie da parte di una biblioteca remota, inserendo opportuni record nella tabella TrasferimentoCopiaInBiblio. Questa operazione attiva, a sua volta, il trigger 9 "richiesta_da_biblioteca_di_copia", che aggiorna lo stato della copia selezionata, segnalandola come: "In prestito altra biblioteca".

```

CREATE EVENT arrivo_richieste_biblioteche
ON SCHEDULE EVERY 10 MINUTE
STARTS '2025-03-21 15:40:49.000'
ON COMPLETION PRESERVE
DISABLE
DO BEGIN
    DECLARE isbn_scelto VARCHAR(100);
    DECLARE disponibilita VARCHAR(100);
    DECLARE num_copia INT;
    DECLARE id_biblio INT;
    DECLARE nomebiblio_log VARCHAR(100);
    DECLARE citta_log VARCHAR(100);

    -- Seleziona un ISBN disponibile casuale
    SELECT c.ISBN INTO isbn_scelto
    FROM BNCR.Copia c
    JOIN BNCR.Libro l ON c.ISBN = l.ISBN
    WHERE c.stato = 'Disponibile' AND l.Disponibile = 'si'
    ORDER BY RAND()
    LIMIT 1;

    -- Seleziona la copia disponibile corrispondente
    SELECT c.stato, c.Numero_Copia INTO disponibilita, num_copia
    FROM BNCR.Copia c
    WHERE c.ISBN = isbn_scelto AND c.stato = 'Disponibile'
    LIMIT 1;

    -- Seleziona una biblioteca casuale (diversa da quella dell'ISBN
    selezionato)
    SELECT dc.ID_Biblioteca INTO id_biblio
    FROM BNCR.DisponibilitaCopia dc
    WHERE dc.ISBN != isbn_scelto
    ORDER BY RAND()
    LIMIT 1;

```

```
-- Recupera i dati della biblioteca per log
SELECT b.Nome, b.Citta INTO nomebiblio_log, citta_log
FROM BNCR.Biblioteca b
WHERE b.ID_Biblioteca = id_biblio;

-- Inserisce log evento
INSERT INTO BNCR.Log_Eventi (messaggio, data_evento)
VALUES (
    CONCAT('Eseguito evento: ISBN scelto = ', isbn_scelto,
           ', Stato = ', disponibilita,
           ', Numero copia = ', num_copia,
           ', nome biblio = ', nomebiblio_log,
           ', città = ', citta_log),
    NOW()
);

-- Inserisce il trasferimento
INSERT INTO BNCR.TrasferimentoCopiaInBiblio (ISBN, Numero_Copia,
ID_Biblioteca)
VALUES (isbn_scelto, num_copia, id_biblio);
-- Occhio, quà parte er trigger.
```

END

Ogni operazione automatica generata dall'evento di simulazione del prestito inter-bibliotecario viene registrata all'interno della tabella Log_Eventi, che ha la funzione di mantenere tracciabilità e storicizzazione delle attività generate dal sistema.

In particolare, ad ogni attivazione dell'evento pianificato:

- Viene annotato il timestamp esatto dell'operazione,
- Vengono indicati i dettagli dell'ISBN, della copia coinvolta e delle biblioteche interessate.

Al fine di simulare la **restituzione automatica di un prestito inter-bibliotecario**, ovvero una copia precedentemente richiesta da una biblioteca esterna del circuito, è stato implementato un **evento schedulato** che agisce periodicamente aggiornando lo stato delle copie.

In particolare, l'evento si occupa di **selezionare una copia attualmente in stato 'In prestito altra biblioteca'** e di eliminarne la relativa riga dalla tabella TrasferimentoCopiaInBiblio, la quale rappresenta il prestito inter-bibliotecario in corso.

L'eliminazione di tale riga attiva automaticamente il **Trigger 11 (reimposta_valore_disponibile_da_trasferimento)**, il quale aggiorna lo stato della copia nella tabella Copia, ripristinandolo a 'Disponibile'.

Questo meccanismo simula in modo efficace e trasparente la dinamica di **ritorno alla disponibilità di una copia prestata nel circuito**, ed evidenzia l'interazione tra **eventi temporizzati e trigger reattivi**, fondamentali per garantire la coerenza del sistema.

```

CREATE EVENT reimposta_disponibile_richiesta
ON SCHEDULE EVERY 2 MINUTE
STARTS '2025-04-02 11:11:12.000'
ON COMPLETION PRESERVE
DISABLE
DO BEGIN

    declare isbn varchar(100);
    declare numero_copia int;
    select BNCR.Copia.ISBN, BNCR.Copia.Numero_Copia into isbn, numero_copia
    from BNCR.Copia
    where BNCR.Copia.stato='In prestito altra biblioteca'
    ORDER BY RAND()
    LIMIT 1;

    DELETE FROM BNCR.TrasferimentoCopiaInBiblio
    WHERE BNCR.TrasferimentoCopiaInBiblio.ISBN=isbn
    AND BNCR.TrasferimentoCopiaInBiblio.Numero_Copia=numero_copia;

    -- Let's gooo triggerr

    INSERT INTO BNCR.Log_Eventi (messaggio, data_evento)
    VALUES (
        CONCAT('Eseguito evento: ISBN scelto per tornare disponibile = ',
isbn,
        ', Numero copia = ', numero_copia),
        NOW()
    );

END

```

Non si è ritenuto opportuno implementare un evento periodico per la cancellazione dei dati presenti nella tabella Log_Prestiti, in quanto essa riveste un ruolo fondamentale nella **tracciabilità storica delle operazioni di prestito**.

Il mantenimento permanente di tali informazioni consente infatti di **ricostruire in qualsiasi momento lo storico dei prestiti effettuati**, permettendo la dismissione del libro se questo non risulta prestato nei passati 10 anni.

La scelta progettuale è quindi orientata a **garantire la persistenza completa dello storico**.

VISTE

```
-- BNCR.report_possesso_interno source
```

```
CREATE OR REPLACE
ALGORITHM = UNDEFINED VIEW `bncr`.`report_possesso_interno` AS
select
    `bncr`.`prestito`.`NumeroCopia` AS `NumeroCopia`,
    `bncr`.`prestito`.`Nome` AS `Nome`,
    `bncr`.`prestito`.`Cognome` AS `Cognome`,
    `bncr`.`libro`.`Titolo` AS `Titolo`,
    `contatti`.`Data_Nascita` AS `Data_Nascita`,
    `bncr`.`libro`.`ISBN` AS `ISBN`,
    `contatti`.`Contatto` AS `Contatto`,
    `contatti`.`Tipo_Contatto` AS `Tipo_Contatto`
from
    ((`bncr`.`prestito`
join `bncr`.`libro`)
join (
    select
        `bncr`.`utenteemail`.`Nome` AS `Nome`,
        `bncr`.`utenteemail`.`Cognome` AS `Cognome`,
        `bncr`.`utenteemail`.`Data_Nascita` AS `Data_Nascita`,
        `bncr`.`utenteemail`.`Email` AS `Contatto`,
        'Email' AS `Tipo_Contatto`
    from
        `bncr`.`utenteemail`
union
    select
        `bncr`.`utentetelefono`.`Nome` AS `Nome`,
        `bncr`.`utentetelefono`.`Cognome` AS `Cognome`,
        `bncr`.`utentetelefono`.`Data_Nascita` AS `Data_Nascita`,
        `bncr`.`utentetelefono`.`Telefono` AS `Contatto`,
        'Telefono' AS `Tipo_Contatto`
    from
        `bncr`.`utentetelefono`) `Contatti`)
where
    ((`bncr`.`prestito`.`ISBN` = `bncr`.`libro`.`ISBN`)
    and (`bncr`.`prestito`.`Nome` = `contatti`.`Nome`)
    and (`bncr`.`prestito`.`Cognome` = `contatti`.`Cognome`)
    and (`bncr`.`prestito`.`Data_Nascita` = `contatti`.`Data_Nascita`));
```

La vista report_possesto_interno è stata progettata per offrire una visualizzazione aggregata e immediata dei prestiti attualmente in corso da parte di utenti interni alla biblioteca.

La parte centrale della vista è costruita con una subquery UNION tra le due modalità di contatto possibili, normalizzate in un'unica struttura logica denominata Contatti.

```
-- BNCR.report_possesto_esterno source
```

```
CREATE OR REPLACE
ALGORITHM = UNDEFINED VIEW `bngr`.`report_possesto_esterno` AS
select
    `bngr`.`prestito_altrove`.`Nome` AS `Nome`,
    `bngr`.`prestito_altrove`.`Cognome` AS `Cognome`,
    `bngr`.`libro`.`Titolo` AS `Titolo`,
    `contatti`.`Data_Nascita` AS `Data_Nascita`,
    `bngr`.`libro`.`ISBN` AS `ISBN`,
    `contatti`.`Contatto` AS `Contatto`,
    `contatti`.`Tipo_Contatto` AS `Tipo_Contatto`,
    `bngr`.`biblioteca`.`Nome` AS `NomeBiblio`,
    `bngr`.`biblioteca`.`Citta` AS `Citta`
from
    (((`bngr`.`prestito_altrove`
join `bngr`.`libro`)
join `bngr`.`disponibilitacopia`)
join `bngr`.`biblioteca`)
join (
    select
        `bngr`.`utenteemail`.`Nome` AS `Nome`,
        `bngr`.`utenteemail`.`Cognome` AS `Cognome`,
        `bngr`.`utenteemail`.`Data_Nascita` AS `Data_Nascita`,
        `bngr`.`utenteemail`.`Email` AS `Contatto`,
        `Email` AS `Tipo_Contatto`
    from
        `bngr`.`utenteemail`
union
    select
        `bngr`.`utentetelefono`.`Nome` AS `Nome`,
        `bngr`.`utentetelefono`.`Cognome` AS `Cognome`,
        `bngr`.`utentetelefono`.`Data_Nascita` AS `Data_Nascita`,
        `bngr`.`utentetelefono`.`Telefono` AS `Contatto`,
        `Telefono` AS `Tipo_Contatto`
    from
        `bngr`.`utentetelefono` ) `Contatti`)
where
    ((`bngr`.`prestito_altrove`.`ISBN` = `bngr`.`libro`.`ISBN`)
        and (`bngr`.`prestito_altrove`.`Nome` = `contatti`.`Nome`)
            and (`bngr`.`prestito_altrove`.`Cognome` = `contatti`.`Cognome`)
                and (`bngr`.`prestito_altrove`.`Data_Nascita` = `contatti`.`Data_Nascita`)
                    and (`bngr`.`disponibilitacopia`.`ISBN` =
`bngr`.`prestito_altrove`.`ISBN`)
                        and (`bngr`.`disponibilitacopia`.`ID_Biblioteca` =
`bngr`.`biblioteca`.`ID_Biblioteca`)
                            and (`bngr`.`disponibilitacopia`.`ID_Biblioteca` =
`bngr`.`prestito_altrove`.`ID_Biblioteca`)
                                and (`bngr`.`biblioteca`.`ID_Biblioteca` =
`bngr`.`prestito_altrove`.`ID_Biblioteca`));
```

La vista report_possesto_esterno fornisce una rappresentazione strutturata dei prestiti attualmente in corso da parte di biblioteche esterne nel circuito bibliotecario. Questa struttura è pensata per facilitare la generazione di report.

```
-- BNCR.categorie source
```

CREATE OR REPLACE

```
ALGORITHM = UNDEFINED VIEW `bncr`.`categorie` AS
select
  `bncr`.`categoria`.`Codice_categoria` AS `Codice_categoria`,
  `bncr`.`categoria`.`nome` AS `nome`
from
  `bncr`.`categoria`;
```

La vista categorie è una semplice rappresentazione della tabella Categoria, dalla quale estrae tutti i codici e i nomi associati alle categorie presenti nel sistema.

In particolare, la vista fornisce:

- Codice_categoria: identificatore univoco della categoria
- nome: denominazione della categoria

```
-- BNCR.disponi source
```

CREATE OR REPLACE

```
ALGORITHM = UNDEFINED VIEW `bncr`.`disponi` AS
select
  distinct `bncr`.`libro`.`Titolo` AS `Titolo`,
  `bncr`.`libro`.`ISBN` AS `ISBN`,
  `bncr`.`scaffale`.`Codice_Scaffale` AS `Codice_Scaffale`,
  `bncr`.`scaffale`.`Nome` AS `Nome`,
  `bncr`.`ripiano`.`Numero_Ripiano` AS `Numero_Ripiano`
from
  (((`bncr`.`libro`
join `bncr`.`scaffale`)
join `bncr`.`ripiano`)
join `bncr`.`copia`)
where
  ((`bncr`.`libro`.`Codice_categoria` =
`bncr`.`scaffale`.`Codice_categoria`)
  and (`bncr`.`ripiano`.`Codice_Scaffale` =
`bncr`.`scaffale`.`Codice_Scaffale`)
  and (`bncr`.`ripiano`.`ISBN` = `bncr`.`libro`.`ISBN`)
  and (`bncr`.`libro`.`ISBN` = `bncr`.`copia`.`ISBN`)
  and (`bncr`.`copia`.`stato` = 'Disponibile')
  and (`bncr`.`libro`.`Disponibile` = 'si'));
```

La vista disponi è stata progettata per fornire una panoramica sintetica e filtrata dei libri attualmente disponibili al prestito, indicando al contempo la posizione fisica di ciascuna copia nel sistema bibliotecario.

In particolare, la vista integra i dati provenienti da:

- Libro: per titolo, ISBN, categoria e disponibilità logica
- Copia: per verificare lo stato fisico della disponibilità (Disponibile)

- Scaffale e Ripiano: per determinare la collocazione fisica del volume

I criteri di selezione garantiscono che vengano mostrati solo i libri effettivamente disponibili, cioè:

- il cui stato nella tabella Copia è 'Disponibile'
- e il cui flag Disponibile nella tabella Libro è impostato a 'si'.

STORED PROCEDURE E TRANSAZIONI.

In questa sezione vengono illustrate e commentate le stored procedure e le transazioni implementate all'interno del sistema.

Particolare attenzione è stata riservata all'**uso delle transazioni (All-or-Nothing)**, adottate in tutte quelle procedure che richiedono coerenza tra operazioni multiple. Tali transazioni sono state configurate con appropriati livelli di isolamento.

Tutte le stored procedure sono state definite con questo preambolo:

CREATE DEFINER='root'@'localhost'

La clausola DEFINER = 'root'@'localhost' specifica che la procedura è stata creata dall'utente root connesso dal server locale (localhost) e che, di conseguenza, verrà eseguito con i privilegi associati a tale utente, indipendentemente da chi lo invoca.

Questo meccanismo consente un controllo granulare dei permessi: anche un utente con privilegi limitati può invocare un oggetto definito da un utente con privilegi più elevati, purché abbia l'autorizzazione all'esecuzione.

Prima di iniziare è opportuna una piccola precisazione: in questo progetto non sono stati utilizzati parametri di output (OUT o INOUT) all'interno delle stored procedure. Tale scelta è stata presa in modo consapevole.

Nel presente progetto, alcune stored procedure sono state progettate per restituire **molteplici result set** e, in determinati casi, è stato adottato l'ausilio del cursore.

Alcune procedure presenti nel progetto sono state dichiarate con la clausola **READS SQL DATA**, indicando che esse **accedono a dati presenti nel database** tramite operazioni di lettura (es. SELECT), ma **non eseguono modifiche sui dati stessi**.

Altre procedure, invece, sono state definite con la clausola **MODIFIES SQL DATA**, che specifica che la routine **può modificare i dati presenti nelle tabelle**, attraverso istruzioni come INSERT, UPDATE o DELETE.

1) login_iniziale

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`login_iniziale`(IN identificativo_var INT, IN password_var VARCHAR(100))
READS SQL DATA
BEGIN

declare done int default false;
declare buffer VARCHAR(100);
declare var_nome VARCHAR(100);
declare var_cognome VARCHAR(100);

declare cur cursor for

SELECT BNCR.Permessi.Nome, bncr.Permessi.Cognome FROM BNCR.Permessi
WHERE BNCR.Permessi.Identificativo = identificativo_var
AND BNCR.Permessi.Password = md5(password_var);

declare exit handler for sqlexception
begin
    rollback;
    resignal;
end;
declare continue handler for not found set done = true;
drop temporary table if exists `BNCR`.`appoggio`;
create temporary table `appoggio`(

`nome` varchar(100),
`cognome` varchar(100)
);

set TRANSACTION ISOLATION LEVEL REPEATABLE READ;
set TRANSACTION READ ONLY;
START TRANSACTION;

open cur;
read_loop:loop
    fetch cur into var_nome, var_cognome;
    if done then
        leave read_loop;
    end if;
    insert into `appoggio` values(var_nome, var_cognome);
end loop;
close cur;

select * from `appoggio`;

open cur;
set done=false;
read_loop:loop
    fetch cur into var_nome, var_cognome;
    if done then
        leave read_loop;
    end if;

```

Per questa procedura ho adottato questo principio: Se una transazione legge un dato, quel dato non può essere modificato da altre transazioni fino alla fine della transazione corrente.

```

select BNCR.Permessi.Ruolo into buffer from BNCR.Permessi
where BNCR.Permessi.Nome=var_nome and BNCR.Permessi.Cognome=var_cognome;
end loop;
close cur;

COMMIT;

if buffer='Amministratore' then SELECT 1 AS Ruolo;
elseif buffer='Bibliotecario' then SELECT 2 AS Ruolo;
else SELECT 3 AS Ruolo;
end if;
END

```

Livello di isolamento: REPEATABLE READ.

Esiste la possibilità che qualcuno possa eliminare (o modificare) una tupla dalla tabella *Permessi* (committando) mentre la procedura *login_iniziale* è in esecuzione.

READ UNCOMMITTED sarebbe davvero una buona scelta pratica?

Non proprio, permette di leggere dati non ancora committati: **NON BLOCCANTE**.

In compenso potrebbe essere più veloce, questo è vero, però immagina che un amministratore stia aggiornando la password di un utente: nel frattempo parte la *login_iniziale* con *READ UNCOMMITTED* e potrebbe leggere la password "nuova" prima del commit.

Poi, se l'admin fa *ROLLBACK*, ho autenticato un utente con una password mai confermata.

(Si presume che nessuno modifichi la tabella *permessi* = livello isolamento basso).

2) validazione

```

CREATE DEFINER='root'@'localhost' PROCEDURE `BNCR`.`validazione`() BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE isbn VARCHAR(100);
    DECLARE dat DATE;

    DECLARE cur CURSOR FOR
        SELECT BNCR.Log_Prestiti.ISBN, MAX(BNCR.Log_Prestiti.`Data`)
        FROM BNCR.Log_Prestiti
        GROUP BY BNCR.Log_Prestiti.ISBN;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    SET TRANSACTION READ WRITE;
    START TRANSACTION;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO isbn, dat;
        IF done THEN
            LEAVE read_loop;
        END IF;

        UPDATE BNCR.Libro
        SET BNCR.Libro.Disponibile = 'no'
        WHERE BNCR.Libro.ISBN = isbn AND dat <= DATE_SUB(CURDATE(), INTERVAL
10 YEAR);
    END LOOP;

    CLOSE cur;

    COMMIT;

END

```

Livello di isolamento: REPEATABLE READ.

Con READ COMMITTED: Se un nuovo prestito viene inserito mentre la procedura è in esecuzione, la MAX(Data) per quell'ISBN potrebbe cambiare da una riga all'altra.

Funzionalità operativa della procedura.

La procedura validazione verifica, per ciascun libro, se l'ultimo prestito registrato risale a più di 10 anni fa. In tal caso, imposta il campo Disponibile a 'no', segnalando la rimozione dal circuito.

3) report_completo

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`report_completo`() 
    READS SQL DATA
BEGIN
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    SET TRANSACTION READ ONLY;
    START TRANSACTION;

    SELECT * FROM BNCR.disponi;

    COMMIT;
END;

```

4) report_personale

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`report_personale`(IN
ruolo_var VARCHAR(100))
BEGIN

declare exit handler for sqlexception
begin
    rollback;
    resignal;
end;

set transaction isolation level READ COMMITTED;
set transaction read write;

start transaction;

IF (SELECT COUNT(*) FROM BNCR.Permessi WHERE Ruolo = ruolo_var) = 0 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Nessun personale trovato per il ruolo specificato';
END IF;

SELECT BNCR.Permessi.Nome, BNCR.Permessi.Cognome,
BNCR.Permessi.Identificativo
from BNCR.Permessi
where BNCR.Permessi.Ruolo = ruolo_var;

COMMIT;

END

```

Livello di isolamento: READ COMMITTED.

Se nel frattempo qualcuno aggiunge o rimuove un bibliotecario, **non è critico**: la procedura mostra solo lo stato attuale. Con il livello di isolamento READ COMMITTED, tra la SELECT COUNT(*) e la SELECT finale, qualcuno potrebbe modificare la tabella Permessi, ad esempio cancellare l'unico utente con quel ruolo, ma sto supponendo che tale azione non avvenga mai.

Funzionalità operativa della procedura.

La procedura report_personale restituisce l'elenco del personale associato a un determinato ruolo, mostrando nome, cognome e identificativo. Se non viene trovato alcun utente con il ruolo specificato, viene sollevato un errore.

5) registra_libro_e_copia

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`registra_libro_e_copia`(IN
isbn VARCHAR(100), IN titolo VARCHAR(100), IN codice_scaffale_var INT, IN
codice_ripiano_var INT, IN codice_categoria_var INT)
BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Errore durante la
registrazione del libro e delle copie';
    END;

    START TRANSACTION;

    INSERT INTO BNCR.Libro (Titolo, ISBN, Codice_categoria, Disponibile)
    VALUES (titolo, isbn, codice_categoria_var, 'si');

    INSERT INTO BNCR.Copia (Numero_Copia, stato, ISBN)
    VALUES (1, 'Disponibile', isbn);

    INSERT INTO BNCR.Ripiano (Numero_Ripiano, Codice_Scaffale, ISBN)
    VALUES (codice_ripiano_var, codice_scaffale_var, isbn);

    COMMIT;
END

```

La procedura `registra_libro_e_copia` esegue solo operazioni di inserimento (INSERT) e non effettua letture concorrentiali, per cui non ho ritenuto necessario specificare un livello di isolamento esplicito. Però ho ritenuto necessario che tutti gli inserimenti fossero eseguiti tutti insieme o nessuno, motivo per il quale si trovano all'interno della transazione.

Funzionalità operativa della procedura.

La procedura `registra_libro_e_copia` inserisce un nuovo libro nel catalogo, ne registra la prima copia e assegna la collocazione fisica su uno specifico ripiano.

6) elimina_personale

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`elimina_personale`(IN
nome_var VARCHAR(100), IN cognome_var VARCHAR(100), IN ruolo VARCHAR(100))
BEGIN

declare nome_appoggio varchar(100);
declare cognome_appoggio varchar(100);

declare exit handler for sqlexception
begin
    rollback;
    resignal;
end;

set transaction isolation level repeatable read;
set transaction READ WRITE;
START TRANSACTION;

select BNCR.Permessi.Nome, BNCR.Permessi.Cognome into nome_appoggio,
cognome_appoggio
from BNCR.Permessi
where BNCR.Permessi.Ruolo=ruolo AND BNCR.Permessi.Nome=nome_var AND
BNCR.Permessi.Cognome=cognome_var;

if nome_appoggio is null and cognome_appoggio is null THEN
    SIGNAL SQLSTATE '45002' set message_text="Non puoi eliminare un personale
inesistente";
end if;

DELETE FROM BNCR.Permessi where BNCR.Permessi.Nome = nome_appoggio and
BNCR.Permessi.Cognome = cognome_appoggio;
COMMIT;

END

```

Livello di isolamento: REPEATABLE READ.

Quello che sto per dire ha un rischio basso, però può accadere: se qualcun altro modifica o elimina il record subito dopo la SELECT INTO ma prima del DELETE, potrei non eliminare nulla o agire su dati non più validi. Quindi READ COMMITTED non è molto consigliato.

Funzionalità operativa della procedura.

La procedura elimina_personale rimuove un membro del personale dalla tabella Permessi, verificando prima, all'interno di una transazione, che esista un utente con il nome, cognome e ruolo specificati.

Se il personale non esiste, viene sollevata un'eccezione e l'operazione viene annullata.

7) elimina_fisicamente_libro

```

CREATE DEFINER=`root`@`localhost` PROCEDURE
`BNCR`.`elimina_fisicamente_libro`(IN nome_libro VARCHAR(100))
BEGIN
    DECLARE isbn_check VARCHAR(100);

    SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
    SET TRANSACTION READ WRITE;
    START TRANSACTION;

    -- Controllaa se il libro esiste
    SELECT ISBN INTO isbn_check
    FROM BNCR.Libro
    WHERE Titolo = nome_libro
    LIMIT 1; -- mi fermo ad un risultato

    -- Se non esiste, segnalo errore ed esco
    IF isbn_check IS NULL THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45591' SET MESSAGE_TEXT = 'Impossibile eliminare: il
libro specificato non esiste.';
    END IF;

    -- Altrimenti eseguo la delete
    DELETE FROM BNCR.Libro
    WHERE Titolo = nome_libro;

    COMMIT;
END

```

Livello di isolamento: READ COMMITTED.

È stato adottato il livello di isolamento READ COMMITTED nella procedura elimina_fisicamente_libro in quanto l'operazione esegue una semplice eliminazione di un record.

DELETE blocca la riga da eliminare. Se un'altra transazione prova a modificare la stessa riga nello stesso istante, verrà messa in attesa del COMMIT/ROLLBACK. Una volta che la transazione termina, la riga sarà eliminata, e il tentativo di update dell'altra transazione non troverà più la riga, risultando in un update di “0 rows”.

Funzionalità operativa della procedura.

La procedura elimina_fisicamente_libro rimuove in modo permanente un libro dalla tabella Libro, in base al titolo fornito come parametro.

8) disponibilità

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`disponibilità` (IN
disponibilita_var VARCHAR(100), IN nome_libro_var VARCHAR(100))
BEGIN
    declare stato_copia varchar(100);
    declare titolo_copia varchar(100);
    declare isbn varchar(100);
    declare done int default false;

    declare cur cursor for
        select BNCR.Libro.Titolo, BNCR.Copia.Stato, BNCR.Libro.ISBN
        from BNCR.Libro, BNCR.Copia
        where BNCR.Copia.stato=disponibilita_var and
BNCR.Libro.Titolo=nome_libro_var
        and BNCR.Libro.ISBN=BNCR.Copia.ISBN and BNCR.Libro.Disponibile='si';

        declare continue handler for not found set done = true;

        declare exit handler for sqlexception
        begin
            rollback;
            resignal;
        end;
drop temporary table if exists `BNCR`.`appoggio`;

create temporary table `appoggio`(
    `titolo` varchar(100),
    `stato` varchar(100),
    `isbn` varchar(100)
);

set TRANSACTION ISOLATION LEVEL SERIALIZABLE;
set TRANSACTION READ ONLY;
START TRANSACTION;

open cur;
read_loop:loop
    fetch cur into titolo_copia, stato_copia, isbn;
    if done then
        leave read_loop;
    end if;
    insert into `appoggio` values(titolo_copia, stato_copia, isbn);
end loop;
close cur;

IF NOT EXISTS (SELECT 1 FROM appoggio) THEN
SIGNAL SQLSTATE '45004' set message_text='Nessuna copia disponibile';
else select count(*) from `appoggio`;
end if;

select isbn from `appoggio` limit 1;
COMMIT;
END

```

Livello di isolamento: SERIALIZABLE.

Ho usato un cursore che legge da tabelle collegate (JOIN tra Libro e Copia) e poi ho popolato una tabella temporanea, l'isolamento SERIALIZABLE mi garantisce che nessun'altra transazione possa modificare, aggiungere o eliminare righe che soddisfano la condizione durante l'esecuzione della procedura.

Funzionalità operativa della procedura.

La procedura BNCR.disponibilita ha il compito di verificare se esistono copie disponibili di un libro con un titolo specificato dall'utente

9) aggiornamento_altrove

```

CREATE DEFINER=`root`@`localhost` PROCEDURE
`BNCR`.`aggiornamento_altrove`(IN ISBN VARCHAR(100), IN TITOLO VARCHAR(100),
IN DISPONIBILITA INT, IN CITY VARCHAR(100), IN NOME_BIBLIO VARCHAR(100), IN
nome_var VARCHAR(100), IN cognome_var VARCHAR(100), IN data_var DATE, IN
contatto_email VARCHAR(100), IN numero_var VARCHAR(100), IN scelta INT, IN
data_fine INT)
BEGIN

    declare contatore INT;
    declare fine DATE;
    declare exit handler for sqlexception
    begin
        rollback;
        resignal;
    end;

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    SET TRANSACTION READ WRITE;
    START TRANSACTION;

    SELECT COUNT(*) INTO contatore
    FROM BNCR.Utente
    WHERE BNCR.Utente.Nome=nome_var
    AND BNCR.Utente.Cognome = cognome_var
    AND BNCR.Utente.Data_Nascita = data_var
    LIMIT 1;

    IF CONTATORE > 0 THEN
        SIGNAL SQLSTATE '45099'
        SET MESSAGE_TEXT = 'Non puoi richiedere un secondo libro, hai già un
libro in prenotazione';
    END IF;

    SET fine = DATE_ADD(CURDATE(), INTERVAL data_fine MONTH);

    INSERT IGNORE INTO BNCR.Utente (Nome, Cognome, Data_Nascita)

```

```

VALUES (nome_var, cognome_var, data_var);

if scelta = 1 then
INSERT INTO BNCR.UtenteEmail(BNCR.UtenteEmail.Nome,
BNCR.UtenteEmail.Cognome, BNCR.UtenteEmail.Data_Nascita,
BNCR.UtenteEmail.Email)
VALUES (nome_var, cognome_var, data_var, contatto_email);
end if;

if scelta = 2 then
INSERT INTO BNCR.UtenteTelefono (BNCR.UtenteTelefono.Nome,
BNCR.UtenteTelefono.Cognome, BNCR.UtenteTelefono.Data_Nascita,
BNCR.UtenteTelefono.Telefono)
VALUES (nome_var, cognome_var, data_var, numero_var);
end if;

INSERT INTO BNCR.Prestito_Altrove(BNCR.Prestito_Altrove.ISBN,
BNCR.Prestito_Altrove.ID_Biblioteca, BNCR.Prestito_Altrove.Nome,
BNCR.Prestito_Altrove.Cognome, BNCR.Prestito_Altrove.Data_Nascita,
BNCR.Prestito_Altrove.DataInizioPrestito,
BNCR.Prestito_Altrove.DataFinePrestito)
VALUES (ISBN,
(SELECT BNCR.Biblioteca.ID_Biblioteca
FROM BNCR.Biblioteca
WHERE BNCR.Biblioteca.Nome = NOME_BIBLIO AND BNCR.Biblioteca.Citta =
CITY), nome_var, cognome_var, data_var,CURDATE(), fine);

COMMIT;

END

```

Livello di isolamento: REPEATABLE READ.

In particolare, il caso in cui possono essere aggiunti più utenti (lettura fantasma e inserimenti simultanei) è rilevante se due transazioni operano contemporaneamente sullo stesso utente, non su utenti diversi. Siccome l'applicazione garantisce che:

- Ciascun utente è identificato univocamente da nome, cognome e data di nascita.
- Non c'è la possibilità che due transazioni contemporanee riguardino lo stesso utente.

Allora va bene REPEATABLE READ.

Funzionalità operativa della procedura.

La procedura verifica se l'utente ha già un prestito attivo, eventualmente lo registra o ne aggiunge i contatti (email o telefono) e inserisce un nuovo prestito presso un'altra biblioteca. Se l'utente risulta già in prenotazione, solleva un errore per evitare duplicati.

10) effettua_riconsegna

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`effettua_riconsegna`(
    IN isbn VARCHAR(100),
    IN numcpy INT,
    IN nome VARCHAR(100),
    IN cognome VARCHAR(100),
    IN dat DATE,
    IN colonne INT,
    IN nome_biblio_var VARCHAR(100),
    IN bibliocitta_var VARCHAR(100))
BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45088' SET MESSAGE_TEXT='Errore durante la riconsegna,
operazione annullata.';
    END;

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    START TRANSACTION;

    IF colonne = 5 THEN

        DELETE FROM BNCR.Prestito
        WHERE BNCR.Prestito.ISBN = isbn AND BNCR.Prestito.NumeroCopia = numcpy
        AND BNCR.Prestito.Nome = nome AND BNCR.Prestito.Cognome = cognome AND
BNCR.Prestito.Data_Nascita = dat;

        DELETE FROM BNCR.Utente
        WHERE BNCR.Utente.Nome = nome AND BNCR.Utente.Cognome = cognome AND
BNCR.Utente.Data_Nascita = dat;

    END IF;

    IF colonne = 6 THEN

        DELETE FROM BNCR.Prestito_Altrove
        WHERE BNCR.Prestito_Altrove.ISBN = isbn
        AND BNCR.Prestito_Altrove.Nome = nome
        AND BNCR.Prestito_Altrove.Cognome = cognome
        AND BNCR.Prestito_Altrove.Data_Nascita = dat;

        DELETE FROM BNCR.Utente
        WHERE BNCR.Utente.Nome = nome AND BNCR.Utente.Cognome = cognome AND
BNCR.Utente.Data_Nascita = dat;

    END IF;

    COMMIT;

END

```

Livello di isolamento: REPEATABLE READ.

Una volta che ho letto i dati (in questo caso, individuato un utente da cancellare), nessun'altra transazione concorrente potrà modificare quelle stesse righe prima che io abbia completato l'operazione, evitandomi di eliminare righe "ormai diverse" o inesistenti. Questo garantisce la coerenza interna della transazione. Non voglio che nessun commit precedente che insiste sulla stessa transazione modifichi la tabella utente, altrimenti cancello un utente non esistente.

Però se viene modificata la riga? Non è possibile aggiornare la riga del database fin tanto che una transazione che ha letto quella riga non va in commit o non va in abort.

Quindi non si verificheranno letture fantasma tali da creare inconsistenze quando cancello un utente già "agganciato" dalla transazione.

Funzionalità operativa della procedura.

La procedura ritorna due result set a seconda della consegna effettuata: consegna per un prestito interno oppure consegna per un prestito esterno.

11) Aggiungi_Copia

```

CREATE DEFINER='root'@'localhost' PROCEDURE `BNCR`.`Aggiungi_Copia`(IN nome_libro
varchar(100))
BEGIN
    DECLARE buffer VARCHAR(100);
    DECLARE maximo INT;
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    SET TRANSACTION READ WRITE;
    START TRANSACTION;

    SET buffer = NULL;

    SELECT ISBN INTO buffer
    FROM BNCR.Libro
    WHERE Titolo = nome_libro AND Disponibile = 'si'
    FOR UPDATE;

    IF buffer IS NULL THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45010' SET MESSAGE_TEXT = 'Errore aggiunta copia: libro non
vendibile o dismesso!';
    END IF;

    SELECT MAX(Numer_Copia) INTO maximo
    FROM BNCR.Copia
    WHERE ISBN = buffer;

    IF maximo IS NULL THEN
        SET maximo = 1;
    ELSE
        SET maximo = maximo + 1;
    END IF;

    INSERT INTO BNCR.Copia (Numer_Copia, stato, ISBN)
    VALUES (maximo, 'Disponibile', buffer);

    COMMIT;
END

```

Livello di isolamento: REPEATABLE READ.

In questo caso, con la select non stiamo imponendo un blocco, se qualcuno esegue un'update per una determinata tupla di libro, può farlo tranquillamente.

Cioè quando si tratta di semplici letture (SELECT “qualcosa”), non si blocca di per sé la riga per le scritture di altre transazioni che insistono su di essa;

Il problema è che questo aggiornamento potrà essere effettuato, quello sicuramente, ma la transazione continuerà a “vedere” lo stato al momento in cui è iniziata: ed è giusto che sia così. Sono le operazioni di scrittura(UPDATE/DELETE/INSERT) che creano un **row lock**.

Ora io voglio che nessuno possa modificare la riga del libro mentre sto aggiungendo la nuova copia, quindi creo un blocco a livello di riga fino al commit.

Con FOR UPDATE, la transazione rimane bloccata sino al COMMIT, assicurando che lo ISBN e il flag Disponibile non cambino durante l'incremento di Numero_Copia.

Funzionalità operativa della procedura.

La procedura aggiunge una copia di un libro già esistente nella biblioteca interna.

12) controllo_circuito

```
CREATE DEFINER='root'@'localhost' PROCEDURE `BNCR`.`controllo_circuito`(IN
titolo_var VARCHAR(100))
```

```
BEGIN
```

```
declare done int default false;
declare i varchar(100);
declare t varchar(100);
declare n int;
declare ci varchar(100);
declare na varchar(100);
declare conta int;
```

```
declare cur cursor for
```

```
SELECT BNCR.DisponibilitaCopia.ISBN, BNCR.Libro.Titolo,
BNCR.DisponibilitaCopia.numero_copie, BNCR.Biblioteca.Nome,
BNCR.Biblioteca.Citta
FROM BNCR.DisponibilitaCopia, BNCR.Libro, BNCR.Biblioteca
WHERE BNCR.DisponibilitaCopia.ISBN = BNCR.Libro.ISBN
AND BNCR.Biblioteca.ID_Biblioteca =
BNCR.DisponibilitaCopia.ID_Biblioteca
AND BNCR.Libro.Titolo=titolo_var AND
BNCR.DisponibilitaCopia.numero_copie > 0;
```

```
declare exit handler for sqlexception
```

```
begin
```

```
    rollback;
    resignal;
```

```
end;
```

```
declare continue handler for not found set done = true;
```

```

drop temporary table if exists `appoggio`;
create temporary table `appoggio`(
  `isbn` varchar(100),
  `titolo` varchar(100),
  `num_copy` int,
  `city` varchar(100),
  `na` varchar(100)
);

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET TRANSACTION READ ONLY;
START TRANSACTION;

open cur;

read_loop:loop
  fetch cur into i,t,n,ci,na;
  if done then
    leave read_loop;
  end if;
  insert into `appoggio` values (i,t,n,ci,na);
END loop;
close cur;

select count(*) into conta from `appoggio`;
if conta = 0 then
  SIGNAL SQLSTATE '45078' SET MESSAGE_TEXT='La copia non è disponibile in
nessuna biblioteca convenzionata';
end if;
if conta > 0 then
  select * from `appoggio`;
end if;
COMMIT;

END

```

Livello di isolamento: SERIALIZABLE

Nella range query voglio evitare che qualcuno modifichi i valori delle tuple interessate nel range. Questo accade perché, per implementare l'isolamento SERIALIZABLE e prevenire i “phantom read”, MySQL blocca non solo le righe lette, ma anche la possibilità di inserire (o modificare) altre righe che soddisferebbero la stessa condizione.

Funzionalità operativa della procedura.

La procedura controlla se i libri sono disponibili in biblioteca e, in caso negativo, controlla la disponibilità nel circuito.

13) aggiungi_libro

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`aggiungi_libro`(
    IN titolo VARCHAR(100),
    IN isbn VARCHAR(100),
    IN categoria INT
)
BEGIN
    DECLARE nomeS VARCHAR(100);
    DECLARE CodS INT;
    DECLARE buffered VARCHAR(100);
    DECLARE buffered_isbn VARCHAR(100);
    DECLARE done INT DEFAULT FALSE;

    DECLARE cur CURSOR FOR
        SELECT BNCR.Scaffale.Nome, BNCR.Scaffale.Codice_Scaffale
        FROM BNCR.Scaffale
        WHERE BNCR.Scaffale.Codice_categoria = categoria;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    SET TRANSACTION READ WRITE;
    START TRANSACTION;

    -- Verifico che la categoria esista
    SELECT BNCR.Categoria.nome INTO buffered
    FROM BNCR.Categoria
    WHERE BNCR.Categoria.Codice_categoria = categoria;

    IF buffered IS NULL THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45012' SET MESSAGE_TEXT = 'La categoria specificata
non esiste';
    END IF;

    -- Verifico che il libro non esista già
    SELECT BNCR.Libro.ISBN INTO buffered_isbn
    FROM BNCR.Libro
    WHERE BNCR.Libro.ISBN = isbn;

    IF buffered_isbn IS NOT NULL THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45013' SET MESSAGE_TEXT = 'Il libro esiste, devi
aggiungere solo una copia';
    END IF;

    -- Creo na tabella temporanea per i risultati
    DROP TEMPORARY TABLE IF EXISTS `appoggio`;
    CREATE TEMPORARY TABLE `appoggio` (
        `nome` VARCHAR(100),
        `codice_scaffale` INT
    );

```

-- Riempio la tabella temporanea con gli scaffali della categoria e fetcho cursore

```
OPEN cur;
SET done = FALSE;
read_loop: LOOP
    FETCH cur INTO nomeS, CodS;
    IF done THEN
        LEAVE read_loop;
    END IF;
    INSERT INTO `appoggio` VALUES (nomeS, CodS);
END LOOP;
CLOSE cur;
```

-- Mostro scaffali disponibili per quella categoria

```
SELECT * FROM `appoggio`;
```

-- Mostro quante copie disponibili esistono già con quel ISBN

```
SELECT COUNT(*) AS CopieDisponibili
FROM BNCR.Copia
WHERE BNCR.Copia.stato = 'Disponibile' AND BNCR.Copia.ISBN = isbn;
```

```
COMMIT;
```

```
END
```

```
END
```

Livello di isolamento: REPEATABLE READ

Se un'altra transazione modifica il record della tabella **Categoria** o **Libro** mentre la tua procedura è in corso, continuo a vedere lo stato precedente (al momento in cui ho fatto START TRANSACTION o la prima SELECT). Solo dopo il mio COMMIT e l'apertura di un'eventuale nuova transazione, vedrò quelle modifiche.

Nota: In questo caso specifico, sto facendo letture su chiavi specifiche (ISBN e Codice_categoria), quindi non ho la classica lettura “range” e di conseguenza non c’è tanto pericolo di “lettura fantasma”.

Funzionalità operativa della procedura.

La procedura crea un’istanza di Libro e ne aggiunge una copia.

14) report_posesso

```

CREATE DEFINER='root'@'localhost' PROCEDURE `BNCR`.`report_posesso`() BEGIN
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    START TRANSACTION READ ONLY;

    SELECT * FROM BNCR.report_posesso_interno;
    SELECT * FROM BNCR.report_posesso_esterno;

    COMMIT;
END

```

Livello di isolamento: REPEATABLE READ

Se un'altra transazione modifica e fa COMMIT sulla stessa riga mentre tu sto leggendo, non vedrò immediatamente quella modifica; continuerò a vedere la versione precedente fino a quando la mia query (o transazione) non termina.

Funzionalità operativa della procedura.

Report dei prestiti interni ed esterni (tramite due apposite view) dei libri all'interno del circuito.

15) controlla_utente

```

CREATE DEFINER='root'@'localhost' PROCEDURE `BNCR`.`controlla_utente`(
    IN nome VARCHAR(100),
    IN cognome VARCHAR(100),
    IN data_var DATE
)
BEGIN
    DECLARE conta INT DEFAULT 0;
    DECLARE conta2 INT DEFAULT 0;

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    SET TRANSACTION READ ONLY;
    START TRANSACTION;

    -- Controlla se l'utente esiste nella tabella Utente
    IF NOT EXISTS (
        SELECT 1 FROM BNCR.Utente
        WHERE BNCR.Utente.Nome = nome
        AND BNCR.Utente.Cognome = cognome
        AND BNCR.Utente.Data_Nascita = data_var
    ) THEN
        SIGNAL SQLSTATE '45024'
        SET MESSAGE_TEXT = 'Utente non esistente';
    END IF;

```

```
-- Conta i prestiti
SELECT COUNT(*) INTO conta
FROM BNCR.Prestito p
WHERE p.Nome = nome
AND p.Cognome = cognome
AND p.Data_Nascita = data_var;

-- Conta i prestiti altrove
SELECT COUNT(*) INTO conta2
FROM BNCR.Prestito_Altrove pa
WHERE pa.Nome = nome
AND pa.Cognome = cognome
AND pa.Data_Nascita = data_var;

-- Se ci sono prestiti, restituiamo la lista
IF conta > 0 THEN
  SELECT p.Nome, p.Cognome, l.Titolo, p.ISBN, p.NumeroCopia
  FROM BNCR.Prestito p
  JOIN BNCR.Libro l ON p.ISBN = l.ISBN
  WHERE p.Nome = nome
  AND p.Cognome = cognome
  AND p.Data_Nascita = data_var;
END IF;

-- Se ci sono prestiti altrove, restituiamo la lista
IF conta2 > 0 THEN
  SELECT
    pa.Nome,
    pa.Cognome,
    l.Titolo,
    pa.ISBN,
    ba.Nome,
    ba.Citta
  FROM BNCR.Prestito_Altrove pa
  JOIN BNCR.Libro l ON pa.ISBN = l.ISBN
  JOIN BNCR.Biblioteca ba ON pa.ID_Biblioteca = ba.ID_Biblioteca
  WHERE pa.Nome = nome
  AND pa.Cognome = cognome
  AND pa.Data_Nascita = data_var;
END IF;

COMMIT;
```

END

Livello di isolamento: REPEATABLE READ

Se qualcuno modifica la tabella utente comunque le modifiche verranno riflesse alla fine della transazione.

Inoltre sto effettuando delle query selettive e non di range, pertanto la probabilità che ci siano letture fantasma è bassa, motivo per il quale non ho adottato il più alto livello di isolamento.

Funzionalità operativa della procedura.

La procedura mostra le informazioni di una copia (o di un libro) presa/o in prestito da un utente.

16)modifica_copia

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `BNCR`.`modifica_copia`(IN
var_isbn VARCHAR(100), IN nome_var VARCHAR(100), IN cognome_var VARCHAR(100),
IN data_var DATE, IN contatto_email VARCHAR(100), IN numero_var VARCHAR(100),
IN scelta INT, IN data_fine INT)
BEGIN

declare copia_disponibile INT;
declare fine DATE;
declare contatore INT;
declare buffer_nome varchar(100);
declare buffer_cognome varchar(100);

declare exit handler for sqlexception
begin
    rollback;
    resignal;
end;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION READ WRITE;
START TRANSACTION;

SELECT Numero_Copia
INTO copia_disponibile
FROM BNCR.Copia
WHERE BNCR.Copia.ISBN = var_isbn and BNCR.Copia.stato='Disponibile'
ORDER BY BNCR.Copia.Numero_Copia ASC
LIMIT 1;

if copia_disponibile is null then
SIGNAL SQLSTATE '45008' SET MESSAGE_TEXT='Nessuna copia disponibile';
end if;

SELECT COUNT(*) INTO contatore
FROM BNCR.Utente
WHERE BNCR.Utente.Nome=nome_var
AND BNCR.Utente.Cognome = cognome_var
AND BNCR.Utente.Data_Nascita = data_var
LIMIT 1;

IF CONTATORE > 0 THEN
SIGNAL SQLSTATE '45098'
    SET MESSAGE_TEXT = 'Non puoi richiedere un secondo libro, hai già un
libro in prenotazione';
END IF;

SET fine = DATE_ADD(CURDATE(), INTERVAL data_fine MONTH);

INSERT IGNORE INTO BNCR.Utente (Nome, Cognome, Data_Nascita)

```

```

VALUES (nome_var, cognome_var, data_var);

if scelta = 1 then
INSERT INTO BNCR.UtenteEmail(BNCR.UtenteEmail.Nome,
BNCR.UtenteEmail.Cognome, BNCR.UtenteEmail.Data_Nascita,
BNCR.UtenteEmail.Email)
VALUES (nome_var, cognome_var, data_var, contatto_email);
end if;

if scelta = 2 then
INSERT INTO BNCR.UtenteTelefono (BNCR.UtenteTelefono.Nome,
BNCR.UtenteTelefono.Cognome, BNCR.UtenteTelefono.Data_Nascita,
BNCR.UtenteTelefono.Telefono)
VALUES (nome_var, cognome_var, data_var, numero_var);
end if;

INSERT INTO BNCR.Prestito (BNCR.Prestito.ISBN, BNCR.Prestito.NumeroCopia,
BNCR.Prestito.Nome, BNCR.Prestito.Cognome, BNCR.Prestito.Data_Nascita,
BNCR.Prestito.DataInizioPrestito, BNCR.Prestito.DataFinePrestito)
VALUES (var_isbn, copia_disponibile, nome_var, cognome_var, data_var,
CURDATE(), fine);

select BNCR.Scaffale.Codice_Scaffale, BNCR.Scaffale.Nome,
BNCR.Ripiano.Numero_Ripiano
from BNCR.Libro, BNCR.Scaffale, BNCR.Ripiano
where BNCR.Libro.Codice_categoria = BNCR.Scaffale.Codice_categoria
and BNCR.Ripiano.Codice_Scaffale = BNCR.Scaffale.Codice_Scaffale
and BNCR.Ripiano.ISBN = BNCR.Libro.ISBN
and BNCR.Ripiano.ISBN = var_isbn;

COMMIT;
END

```

Livello di isolamento: REPEATABLE READ

Non è possibile inserire più utenti uguali, quindi comunque vivrà un solo utente con nome, cognome e data di nascita univoci.

Inoltre se qualcuno aggiorna la copia comunque l'aggiornamento verrebbe bloccato.

Leggendo la prima copia disponibile con SELECT ... LIMIT 1 dentro una transazione a livello REPEATABLE READ, evito di “perdere” quella copia se un’altra transazione prova a modificarne lo stato (“Disponibile” a “Non disponibile”) durante la lettura.

Grazie all’inserimento con INSERT IGNORE, se esiste già un utente con quei dati non viene creato un duplicato.

Inoltre, se qualcuno provasse a rubare la stessa copia con un’altra transazione, non potrebbe farlo, questa operazione è atomica.

Funzionalità operativa della procedura.

La funzione aggiorna lo stato della copia, partìrà un apposito trigger.

17) mostra_categoria

```

CREATE DEFINER='root'@'localhost' PROCEDURE `BNCR`.`mostra_categoria`() BEGIN
    DECLARE total INT;

    -- Conto quante categorie esistono
    SELECT COUNT(*) INTO total
    FROM BNCR.categorie;

    -- Se non esistono categorie, segnalo un errore
    IF total = 0 THEN
        SIGNAL SQLSTATE '45099'
        SET MESSAGE_TEXT = 'Nessuna categoria esistente nel sistema.';
    END IF;

    -- Altrimenti mostro tutto ambaradam
    SELECT * FROM BNCR.categorie;
END

```

18) aggiungi_personale

```

CREATE DEFINER='root'@'localhost' PROCEDURE
`BNCR`.`aggiungi_personale`(IN nome_arg VARCHAR(100), IN
cognome_arg VARCHAR(100), IN password_arg VARCHAR(100), IN
privilegio_arg VARCHAR(100))
BEGIN
    declare done int default false;
    declare var_max INT;
    declare crypt VARCHAR(100);

    declare exit handler for sqlexception
    begin
        rollback;
        resignal;
    end;

    SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
    SET TRANSACTION READ WRITE;
    START TRANSACTION;

    select MAX(BNCR.Permessi.Identificativo) into var_max
    from BNCR.Permessi
    where BNCR.Permessi.Ruolo=privilegio_arg
    FOR UPDATE;

    IF var_max IS NULL THEN

```

```
SIGNAL SQLSTATE '45001'  
SET MESSAGE_TEXT = 'Errore: Il ruolo specificato non esiste nel  
database.';  
END IF;  
  
SET var_max = var_max + 1;  
-- Cripta la password  
SET crypt = MD5(password_arg);  
  
INSERT INTO BNCR.Permessi VALUES(var_max, nome_arg, cognome_arg, crypt,  
privilegio_arg);  
  
COMMIT;  
  
END
```

Livello di isolamento: REPEATABLE READ + FOR UPDATE

Finché la transazione non termina (con COMMIT o ROLLBACK), nessun'altra transazione può alterare o inserire dati in BNCR.Permessi che possano influire su MAX(Identificativo) di quel ruolo. Verrà costretta ad attendere.

In altre parole, sono protetto proprio grazie alla clausola FOR UPDATE: eventuali modifiche in parallelo sulla stessa chiave (stesso ruolo) dovranno aspettare.

Funzionalità operativa della procedura.

La procedura aggiungi un personale, che sia esso un bibliotecario oppure un amministratore.