



Una piattaforma per organizzare e condividere la propria libreria multimediale

Descrizione del progetto

Si vuole realizzare una base di dati a supporto di un sito web che permetta la creazione, gestione e condivisione della propria libreria multimediale, come è possibile fare su altri siti web, quali aNobii, MyAnimeList e The Backlogery.

Il sito sarà suddiviso in sezioni, ciascuna riguardante un media diverso:

- Libri e audiolibri
- Film
- Videogiochi

In futuro, si potrà pensare di espandere il sito web, includendo nuove sezioni, come ad esempio *Serie TV*.

Utenti

Chiunque può registrarsi al sito web scegliendo un username univoco e inserendo una password segreta (sarà hashata con l'algoritmo bcrypt prima che venga inserita nel database), creando così un utente.

Ogni utente avrà una sua raccolta multimediale, in cui potrà aggiungere, modificare e rimuovere *elementi* (descritti in seguito).

Amministratori

Esisterà una tipologia particolare di utente: l'utente **amministratore**.

Gli amministratori potranno modificare le raccolte multimediali di tutti gli utenti, in aggiunta alla propria.

Inoltre, potranno decidere di **bannare** utenti dal sito, impedendo loro di effettuare l'accesso e di conseguenza di interagire con la loro raccolta.

Elementi

Gli utenti potranno aggiungere *elementi* alla loro raccolta multimediale.

Un elemento rappresenta una copia di un libro, di un film o di un videogioco posseduta da un utente.

Ogni elemento avrà associato uno **stato** da una lista di opzioni diverse per ogni tipologia:

- Libro
 - Da iniziare
 - Iniziato

- Finito
- Abbandonato
- Non applicabile
- Film
 - Da vedere
 - Visto
- Videogioco
 - Da iniziare
 - Iniziato
 - Finito
 - Completato al 100%
 - Non applicabile

Inoltre, ogni elemento avrà associata una **provenienza** da un'altra lista:

- Libro
 - Acquistato (su supporto fisico)
 - Acquistato (su supporto digitale)
 - Preso in prestito (da restituire)
 - Perso / Venduto / Restituito / Non più posseduto
 - Altro
- Film
 - Acquistato (su supporto fisico)
 - Visto al cinema
 - Visto in televisione
 - Visto su un servizio di streaming
 - Preso in prestito (da restituire)
 - Perso / Venduto / Restituito / Non più posseduto
 - Altro
- Videogioco
 - Gratuito
 - Acquistato
 - Giocato in abbonamento
 - Preso in prestito (da restituire)
 - Perso / Venduto / Restituito / Non più posseduto
 - Altro

Un utente potrà creare un nuovo oggetto (edizione di libro, film...) di cui aggiungere poi un elemento nella sua libreria.

Per farlo dovrà selezionarne il tipo e inserirne la chiave (ISBN, EIDR...) e il titolo; tutti gli altri campi saranno facoltativi: possono essere compilati per completezza, oppure lasciati vuoti.

Un utente può inoltre modificare una pagina già esistente al fine di aggiungervi informazioni o correggere errori.

Un amministratore è in grado di eliminare le pagine, in caso esse rappresentino oggetti non realmente esistenti.

Recensioni

Un utente potrà lasciare una recensione ad ogni elemento presente nella sua raccolta.

La recensione sarà composta da una valutazione (tra 0 e 100, dove 100 è la valutazione migliore), un commento e la data di pubblicazione.

La media delle valutazioni delle recensioni relativa a un dato libro / film / serie TV / videogioco sarà poi visualizzata nella relativa pagina, assieme ad alcune recensioni selezionate casualmente.

Gli utenti potranno decidere in qualsiasi momento di eliminare una loro recensione.

Gli amministratori potranno eliminare le recensioni nel caso queste violino i termini di servizio del sito web.

Libri ed edizioni

Ogni libro avrà una sua pagina in cui sarà presente il titolo originale, gli autori, i generi, un breve riassunto della trama, l'elenco di tutte le sue edizioni (sia in formato libro sia in formato audiolibro) e opzionalmente una lista di opere correlate (sequel, prequel, libri ambientati nello stesso universo, etc).

Ciascuna edizione del libro avrà una seconda pagina con ulteriori informazioni, quali il suo titolo, la copertina, la casa editrice e il numero di pagine; ciascuna edizione sarà identificata da il relativo codice ISBN.

Le edizioni in formato audiolibro avranno attributi diversi: invece che avere il numero di pagine e la copertina, essi avranno la durata in minuti e secondi della registrazione e opzionalmente un'immagine che rappresenti l'audiolibro.

Recensioni e valutazione media saranno calcolate sia per libro, sia per edizione.

Film

Ogni film avrà una sua pagina in cui sarà presente il titolo originale, i titoli nelle varie lingue (identificati dal codice ISO 639 della lingua), una sinossi della trama, la durata, la casa produttrice, il cast, e, come per i libri, una lista opzionale di pellicole correlate.

I film saranno identificati dal loro codice EIDR, e per ciascuno di essi verrà calcolata la valutazione media dalle recensioni, che sarà visualizzata sulla pagina assieme a un campione di recensioni.

Videogiochi

Ogni videogioco avrà una sua pagina in cui sarà presente il titolo, lo sviluppatore, il publisher, una breve descrizione del gioco, l'elenco di tutte le piattaforme in cui esso è disponibile e, come per libri e film, un elenco di altri giochi correlati.

Per ogni piattaforma sarà disponibile una sottopagina, che conterrà la box art di quella versione, il nome dello studio che ha effettuato il porting ed eventualmente il titolo se diverso da quello principale.

Recensioni e valutazione media saranno disponibili sia per ogni singola piattaforma, sia per il gioco nel suo complesso.

Glossario

Scritta la descrizione, si è realizzato un glossario al fine di identificare le parole chiave in essa contenute.

Generale

Nome	Dati	Sinonimi	Collegamenti	Note
Utente	username, hash della password, email, è amministratore, è bannato	user, admin, amministratore	possiede Elementi	
Elemento	stato, provenienza		di una copia di un Libro / Film / Videogioco, posseduto da un Utente	
Recensione	valutazione (0-100), commento, data, è nascosto	valutazione, commento, post	riguardante un Elemento	

Libri

Nome	Dati	Sinonimi	Collegamenti	Note
Libro	titolo originale, sinossi	testo, saggio, romanzo, opuscolo	scritto da uno o più Autori, appartenente a uno o più Generi, con più Edizioni (libro, audio), correlato ad altri Libri	
Genere	nome		a cui appartengono i Libri	
Edizione (libro)	isbn , titolo localizzato, pagine, copertina	<i>Libro</i> (ambiguo), pubblicazione, stampa	di un Libro, posseduta da più Utenti (<i>Elemento</i>), pubblicata da un Editore	
Edizione (audio)	isbn , titolo localizzato, durata, immagine		di un Libro, posseduta da più Utenti (<i>Elemento</i>), pubblicata da un Editore, narrata da uno o più Narratori	
Autore	nome	scrittore	scrive dei Libri	
Editore	nome		pubblica delle Edizioni (libro, audio)	
Narratore	nome	voce narrante	narra delle Edizioni (audio)	

Film

Nome	Dati	Sinonimi	Collegamenti	Note
Film	eidr, titolo originale, sinossi, durata, locandina	pellicola	guardato da Utenti (<i>Elemento</i>), realizzato da Cast, prodotto da Studio, appartenente a Generi, scritto da Sceneggiatori, correlato ad altri Film	
Genere	nome		a cui appartengono i Film	
Cast	nome	"attore", "regista", "scenografo", "produttore esecutivo"	Prende parte al Film, ha un Ruolo	I ruoli sono specificati nella tabella Ruolo
Ruolo	nome		di Cast	Il ruolo è un attributo!
Studio	nome	azienda, casa produttrice	che ha prodotto un Film	
Localizzazione	titolo alternativo, lingua	titolo	relativo a un Film	

Giochi

Nome	Dati	Sinonimi	Collegamenti	Note
Gioco	titolo, descrizione	videogioco, videogame	sviluppato da uno o più Studio, pubblicato da uno o più Studio, appartenente a Generi, con più Edizioni	
Edizioni	piattaforma, box art, titolo alternativo		di un Gioco, giocata da Utenti (<i>Elemento</i>), portato da uno o più Studio	
Generi	nome		a cui appartengono Giochi	
Studio	nome		che ha sviluppato Giochi, che ha portato Edizioni	

Schema scheletro

Dal glossario, si è poi realizzato con il software diagrams.net uno schema scheletro della base di dati, contenente le principali entità e relazioni.

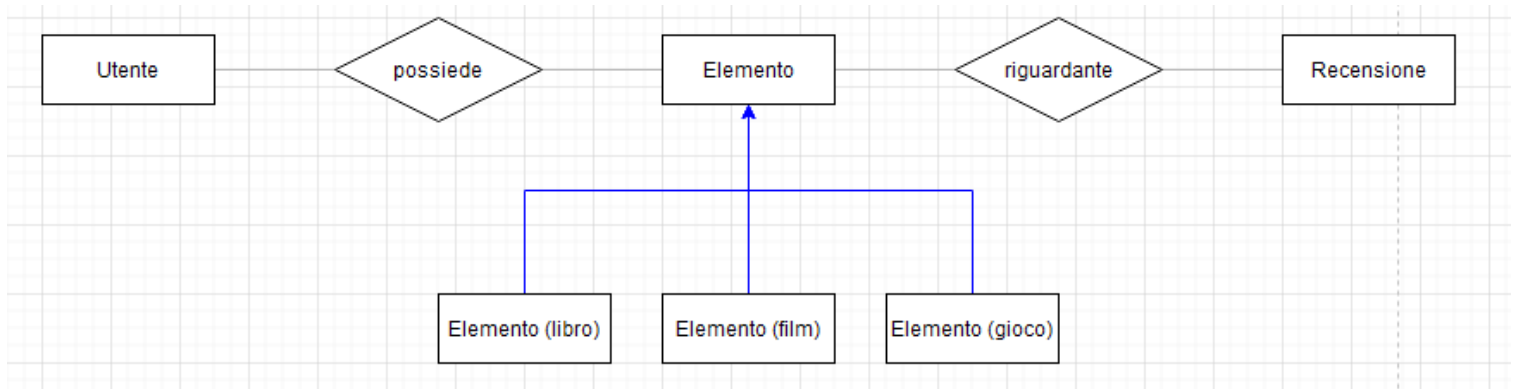
È allegato alla relazione il file `3-1-schema-scheletro.drawio` contenente l'intero schema scheletro in un formato modificabile; per comodità, si riportano qui sotto i suoi contenuti sotto forma di immagini.

Legenda

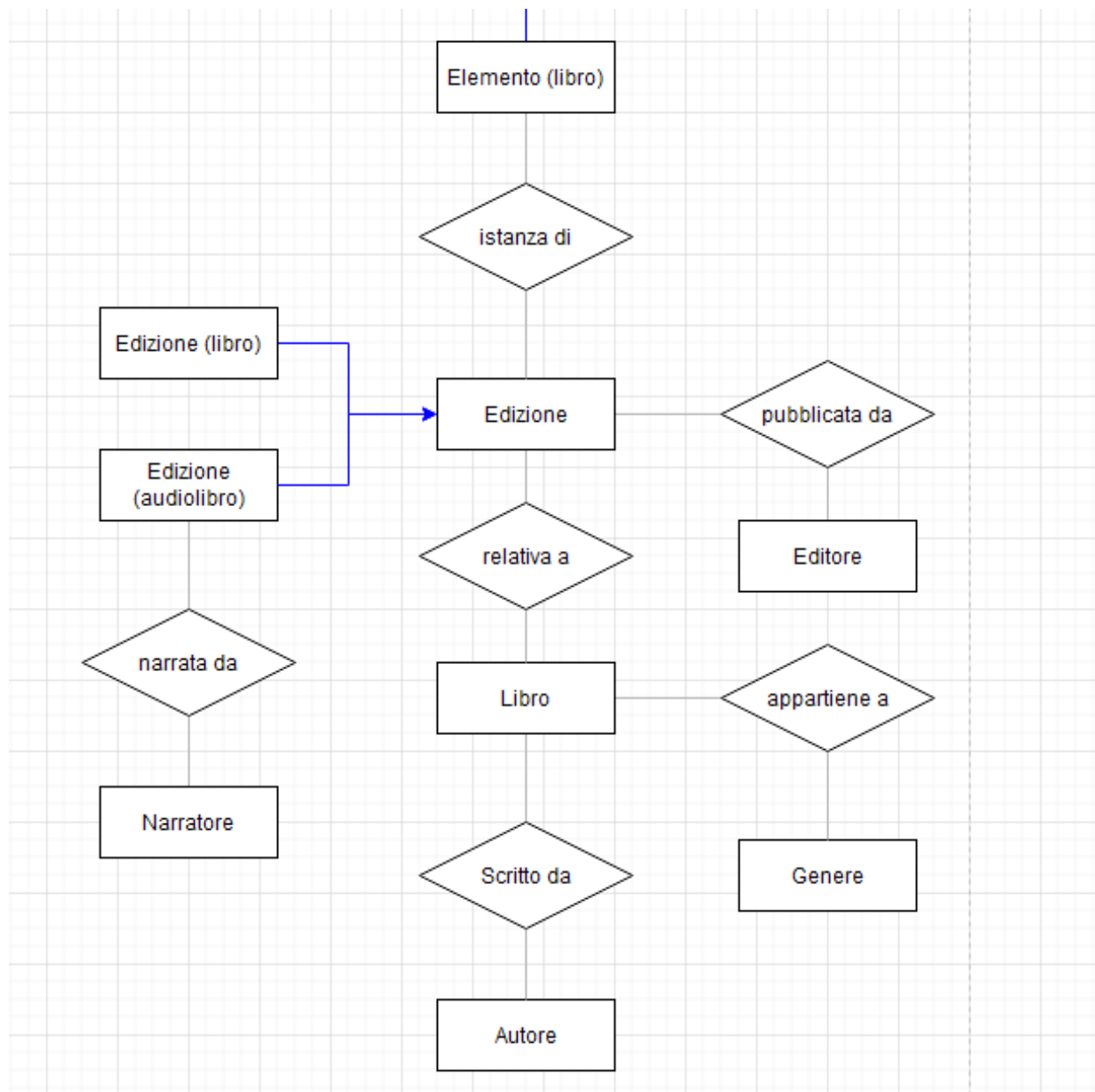
I *quadrati* rappresentano le **entità**, i *rombi* rappresentano le **relazioni**, le *linee continue grigie* rappresentano l'**appartenenza** di un'entità a una relazione e infine le *frecce blu* rappresentano l'esistenza di una **gerarchia IsA** tra le entità connesse.

Immagini

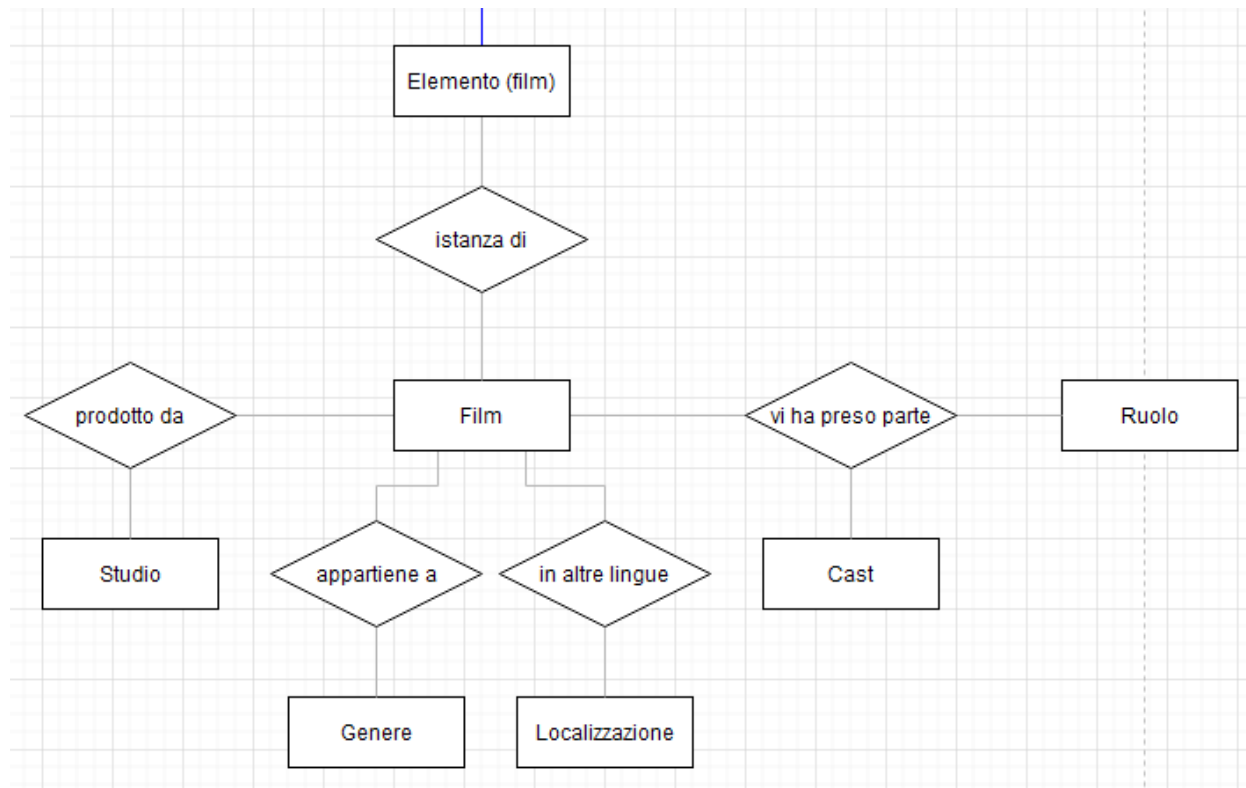
Generale



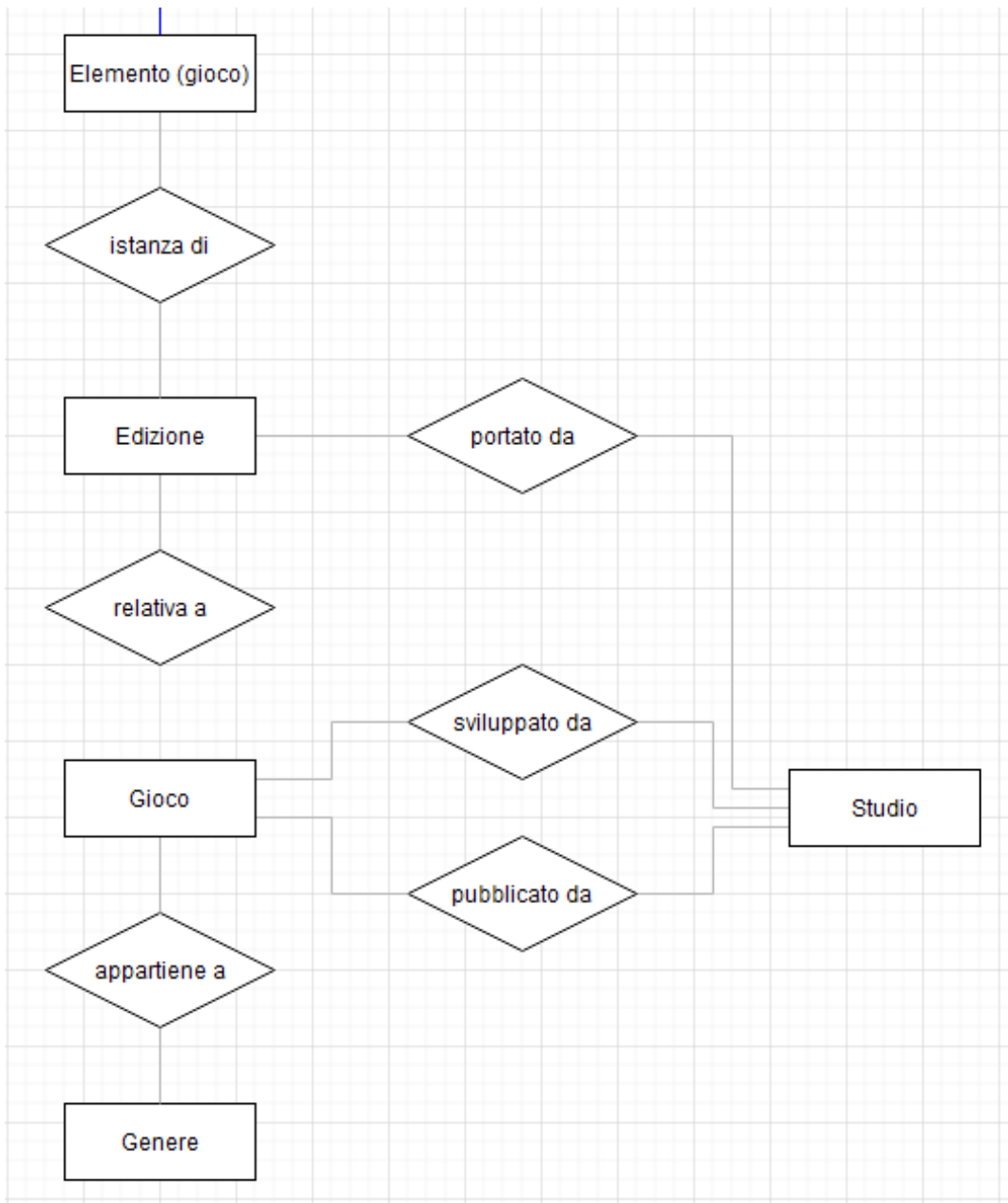
Libri



Film



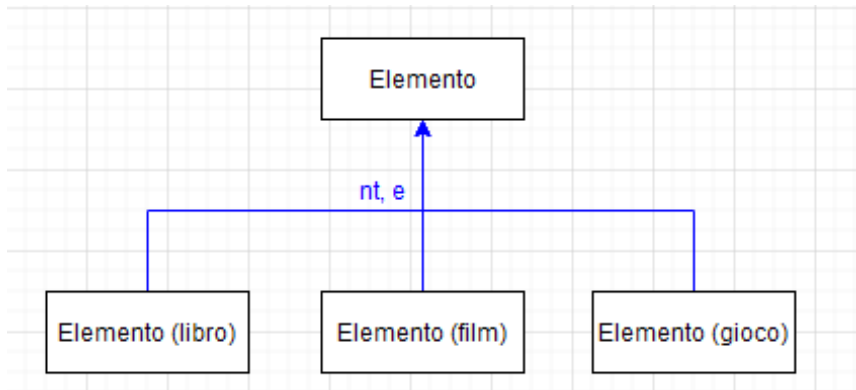
Giochi



Classificazione delle gerarchie

Nello schema scheletro di Alexandria compaiono **due gerarchie IsA**, rappresentate da frecce continue blu.

Gerarchia degli Elementi



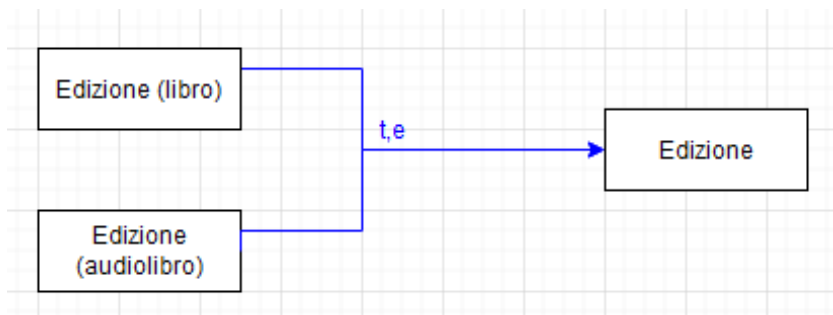
Il tipo di dato alla base di Alexandria è l'Elemento.

Un Elemento rappresenta una aggiunta da parte di un utente alla propria libreria di un libro, film o videogioco.

Nella descrizione si specifica che tutti gli Elementi devono avere uno *stato* e una *provenienza* specifici al tipo di Elemento; è quindi necessaria la distinzione dei vari tipi di Elemento e creando così una gerarchia **esclusiva** (un Elemento non può essere sia un libro sia un film allo stesso tempo).

Si è deciso di rendere **non totale** la gerarchia in modo da permettere l'introduzione di nuovi tipi di Elementi in futuro.

Gerarchia delle Edizioni

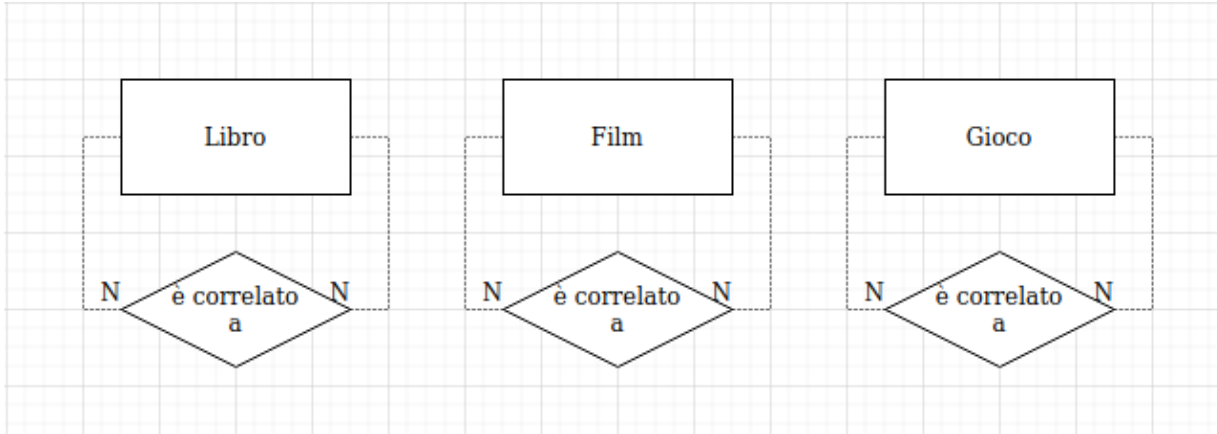


La descrizione prevede che le Edizioni dei libri e degli audiolibri abbiano attributi diversi: i libri hanno il *numero di pagine* e l'immagine della loro *copertina*, mentre gli audiolibri hanno la *durata* e la *cover art* ad essi associata.

Si viene a creare così una gerarchia **totale** (un'Edizione è o libro o audiolibro) ed **esclusiva** (una edizione non può essere sia libro sia audiolibro). # Classificazione delle autoassociazioni

Alexandria prevede la possibilità di marcare come correlate due opere in modo tale da far apparire sulla pagina di una un collegamento all'altra.

Si è deciso di sviluppare questa funzionalità utilizzando delle **autoassociazioni**: ogni `Libro`, `Film` o `Videogioco` sarà correlato ad altre entità del suo stesso tipo tramite la relazione `è correlato a`.



Inizialmente, si è considerato di aggiungere una unica autoassociazione alla entità `Elemento`, ma si è poi giunti alla conclusione che questo approccio sarebbe stato sbagliato: si sarebbero dovuti associare tutti i singoli `Elementi` creati da ciascun `Utente`, creando una quantità di correlazioni a crescita esponenziale!

Una possibile estensione a funzionalità si potrebbe ottenere aggiungendo un attributo `Tipo` alla relazione `è correlato a`: esso permetterebbe agli `Utenti` di descrivere come due opere sono correlate tra loro ("sequel", "prequel", "stesso universo", etc...).

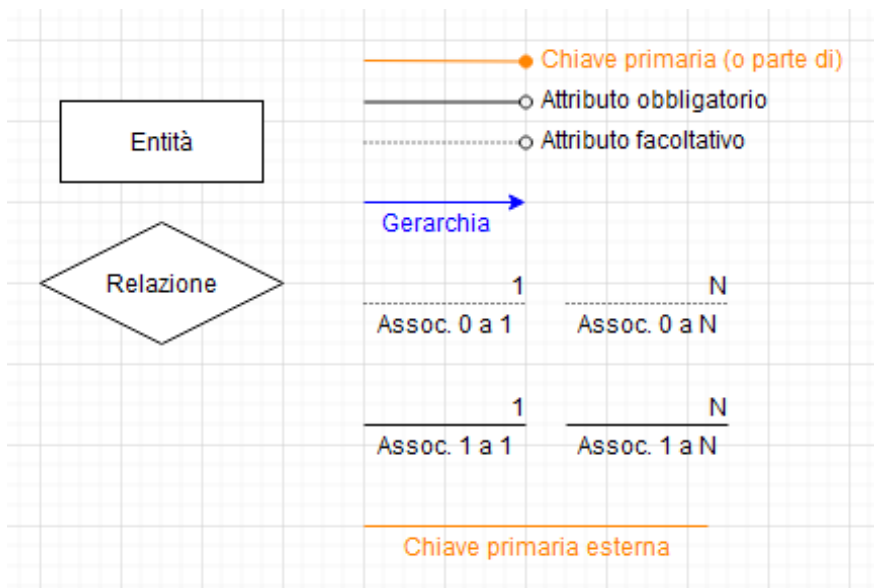
Schema scheletro finale

Dopo aver costruito lo schema scheletro, identificato le gerarchie e le autoassociazioni, si è passati a ultimare lo schema relazionale, aggiungendoci **attributi**, **identificatori** e **cardinalità delle relazioni**.

Per farlo, ci si è basati sui contenuti della descrizione e del glossario: per comodità, si riportano qui le frasi della descrizione relative alle entità di cui si parla.

È allegato alla relazione il file `3-4-schema-finale.drawio`: esso contiene lo schema scheletro finale di cui si vedono le immagini in questo capitolo.

Legenda

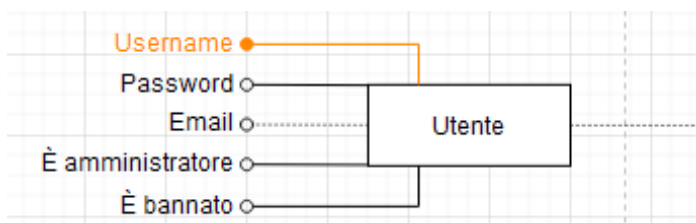


Utenti

Chiunque può registrarsi al sito web scegliendo un username univoco e inserendo una password segreta (sarà hashata con l'algoritmo bcrypt prima che venga inserita nel database), creando così un utente.

Esisterà una tipologia particolare di utente: l'utente **amministratore**.

Inoltre, potranno decidere di **bannare** utenti dal sito, impedendo loro di effettuare l'accesso e di conseguenza di interagire con la loro raccolta.



Per gli **Utenti**, si sono aggiunti gli attributi all'entità seguendo strettamente le specifiche, assieme a un attributo opzionale "email" per permettere un eventuale recupero dell'account nel caso che ci si sia dimenticati la password.

Elementi

Gli utenti potranno aggiungere *elementi* alla loro raccolta multimediale.

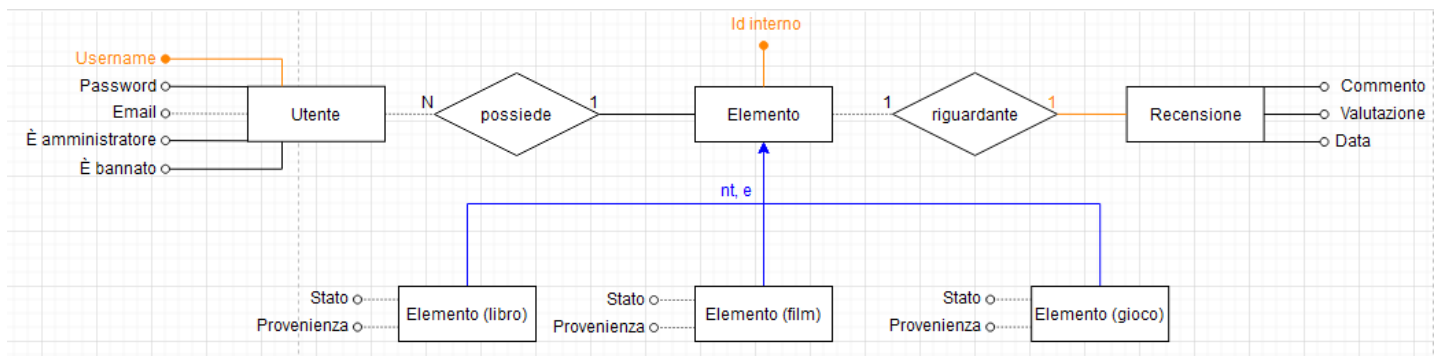
Un elemento rappresenta una copia di un libro, di un film o di un videogioco posseduta da un utente.

Ogni elemento avrà associato uno **stato** da una lista di opzioni diversa per ogni tipologia: [...]

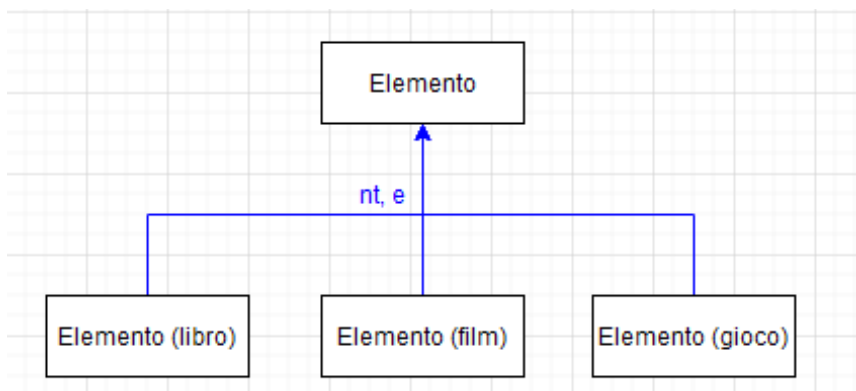
Inoltre, ogni elemento avrà associata una **provenienza** da un'altra lista: [...]

Un utente potrà lasciare una recensione ad ogni elemento presente nella sua raccolta.

La recensione sarà composta da una valutazione (tra 0 e 100, dove 100 è la valutazione migliore), un commento e la data di pubblicazione.



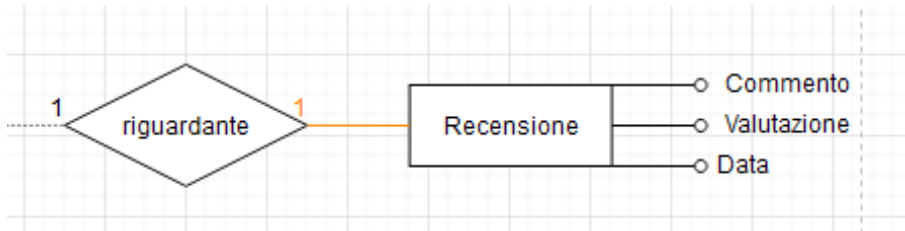
Sottoentità



Gli **Elementi** sono stati suddivisi in tre sottoentità **Elemento (libro)**, **Elemento (film)** ed **Elemento (gioco)** per permettere loro di possedere attributi e relazioni di tipo diverso gli uni dagli altri; infatti, ognuna delle tre sottoentità è dotata di una associazione **istanza di** che le collega rispettivamente alle entità **Edizione (libro)**, **Film** e **Edizione (gioco)**.

Attributi e relazioni di queste sottoentità saranno descritti nelle sezioni successive.

Recensioni



Le **Recensioni** sono collegate agli **Elementi** attraverso la relazione **riguardante**.

Sono un caso di **identificatore esterno**: le **Recensioni** infatti usano come identificatore l'*id* dell'**Elemento** a cui si riferiscono.

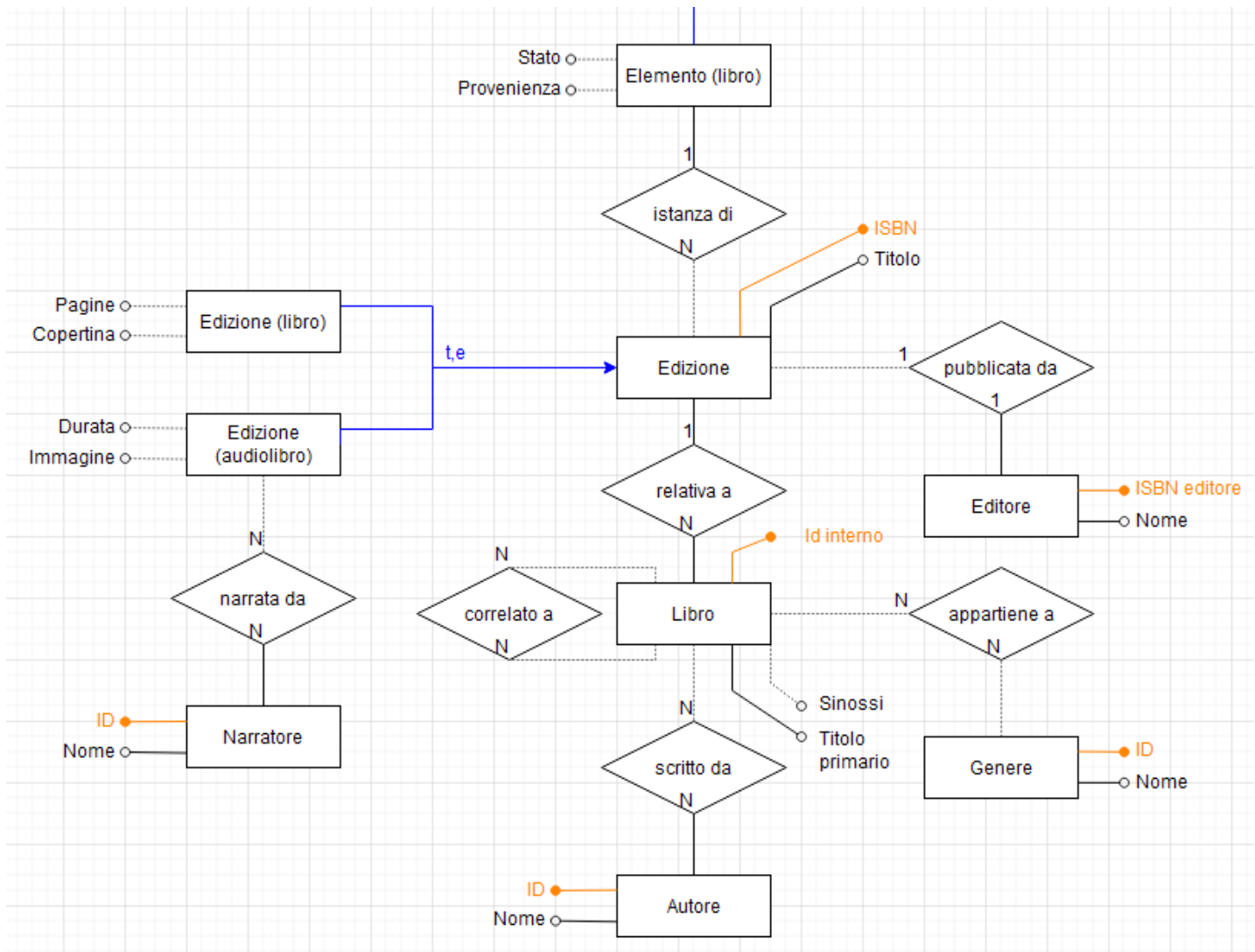
Si può inoltre dire che le **Recensioni** siano una **entità debole** rispetto agli **Elementi**, in quanto senza il suo relativo **Elemento**, una **Recensione** perderebbe importanti informazioni di contesto, e non avrebbe quindi più senso di esistere (infatti, senza **Elemento**, essa non sarebbe più dotata di un identificatore).

Libri

Ogni libro avrà una sua pagina in cui sarà presente il titolo originale, gli autori, i generi, un breve riassunto della trama, l'elenco di tutte le sue edizioni (sia in formato libro sia in formato audiolibro) e [...].

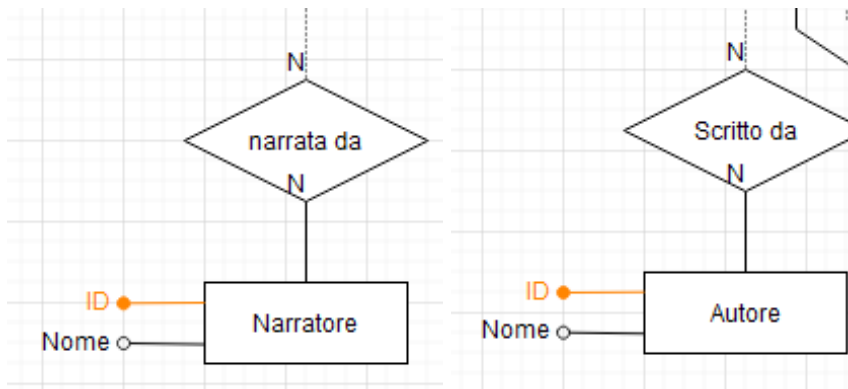
Ciascuna edizione del libro avrà una seconda pagina con ulteriori informazioni, quali il suo titolo, la copertina, la casa editrice e il numero di pagine; ciascuna edizione sarà identificata da il relativo codice ISBN.

Le edizioni in formato audiolibro avranno attributi diversi: invece che avere il numero di pagine e la copertina, essi avranno la durata in minuti e secondi della registrazione e opzionalmente un'immagine che rappresenti l'audiolibro.



Lo schema dei **Libri** è stato realizzato seguendo in buona parte le specifiche; si sono però effettuate alcune aggiunte: - Le **Edizioni** di un **Libro** sono state dotate di un *titolo*, che rappresenta il titolo dell'**Edizione** specifica (titolo in una lingua diversa, edizione speciale che cambia il titolo, etc...); - Le **Edizioni** di un **Audiolibro** possono avere associato uno o più **Narratori**.

Una ricorrenza nelle relazioni: il pattern "1NN0"



Osservando lo schema, si nota che le relazioni "narrata da" e "scritto da" sono molto simili tra loro: tutte e

due sono **0 a N** nel lato che si collega all'opera (il **Libro** o una sua **Edizione**), sono **1 a N** dall'altro lato e si ricollegano a una entità con due soli attributi, *Nome* e *ID*.

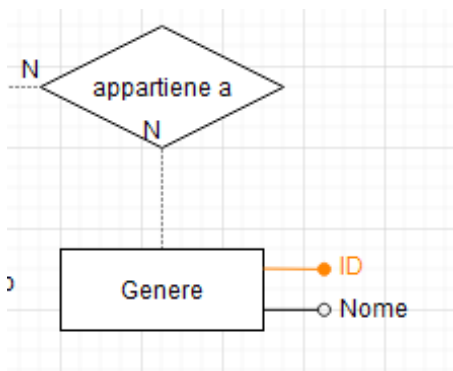
Questa particolare struttura compare in molte parti dello schema relazionale: per evitare di ridescriverla ogni volta e per permettere invece di referenziarla, si è deciso di darle il nome **1NN0** (leggi: "inno").

La lato dell'opera si ha una cardinalità **0 a N**, in modo da evitare agli utenti la compilazione obbligatoria di tutti i campi di un'opera al momento della sua aggiunta, ma permettendo una loro compilazione più dettagliata in futuro, prevedendo anche casi in cui ad esempio un libro sia stato scritto da più autori.

Dato che si è voluto rendere possibili query come "quali **Libri** ha scritto questo autore" o "quali **Libri** ha narrato questo narratore" e che inserire nel database autori o narratori a cui non appartiene nessun libro non avrebbe alcun senso, si è scelto invece di usare una cardinalità **1 a N** dall'altro lato della relazione.

Infine, per l'entità connessa al lato **1 a N** della relazione, si è deciso di usare un **identificatore surrogato**, in modo da permettere la modifica del *Nome* associato senza dover andare a modificare tutte le opere.

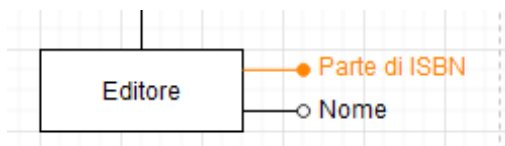
Generi



Notiamo che la relazione che associa un **Libro** a un **Genere** è molto simile alla struttura **1NN0**, ma essa ha una cardinalità **0 a N** anche dal lato dell'entità **Genere**.

Questo perchè si intende aggiungere alcuni **Generi** al database tra cui gli utenti potranno scegliere prima ancora che esistano dei libri.

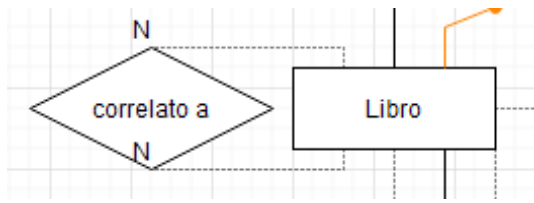
Editori



Tutti i codici ISBN contengono al loro interno un codice univoco che identifica l'editore di un libro; si è quindi deciso di usare questo codice per identificare l'entità **Editore**.

Correlazioni

[...] e opzionalmente una lista di opere correlate (sequel, prequel, libri ambientati nello stesso universo, etc)

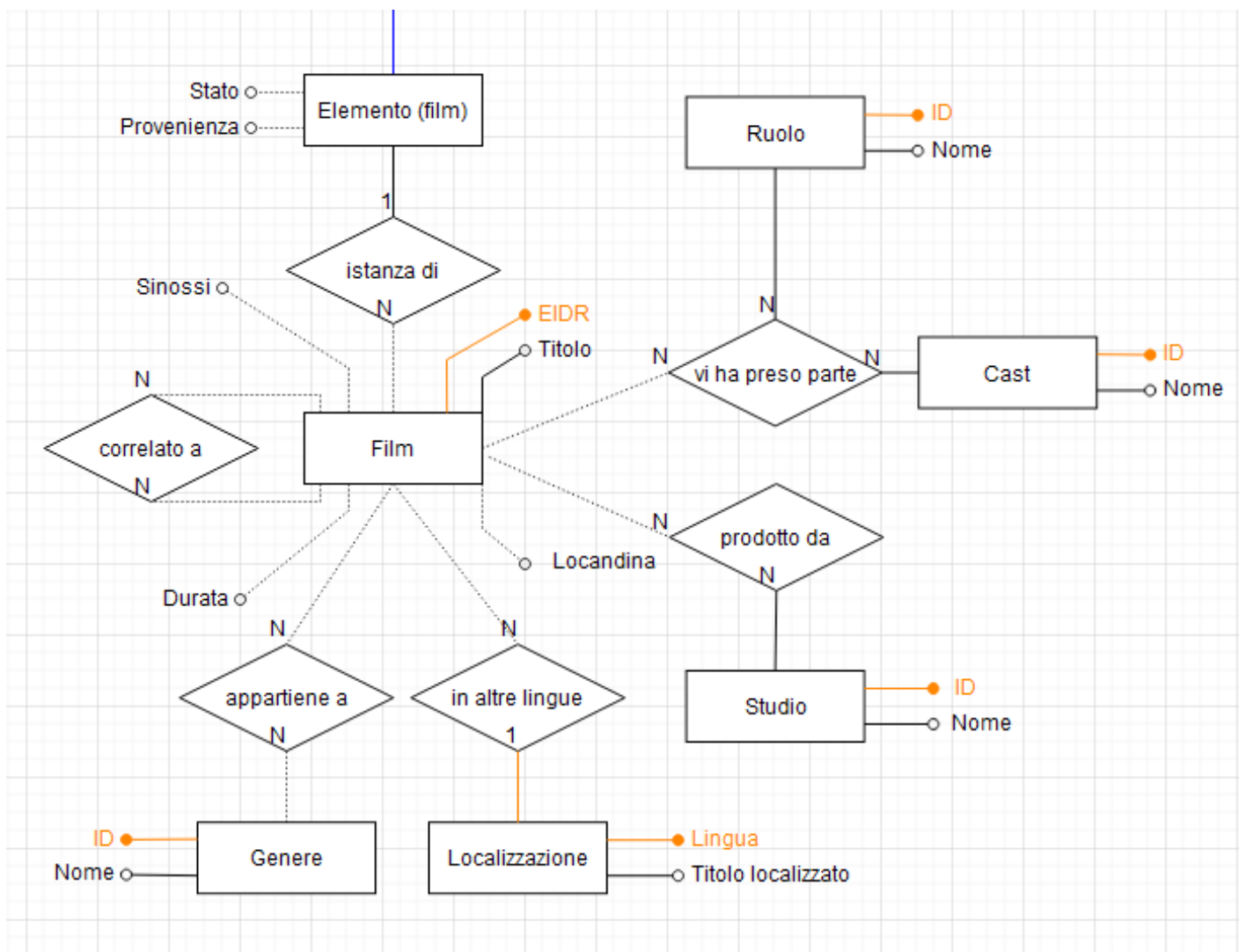


Come menzionato in precedenza, l'entità **Libro** è dotata di una autoassociazione che permette di identificare gli altri **Libri** ad essa correlati.

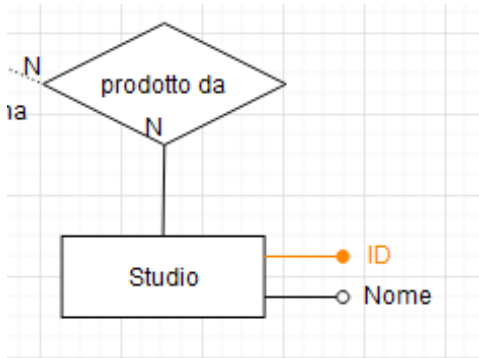
Film

Ogni film avrà una sua pagina in cui sarà presente il titolo originale, i titoli nelle varie lingue (identificati dal codice ISO 639 della lingua), una sinossi della trama, la durata, la casa produttrice, il cast, e, come per i libri, una lista opzionale di pellicole correlate.

I film saranno identificati dal loro codice EIDR, e per ciascuno di essi verrà calcolata la valutazione media dalle recensioni, che sarà visualizzata sulla pagina assieme a un campione di recensioni.

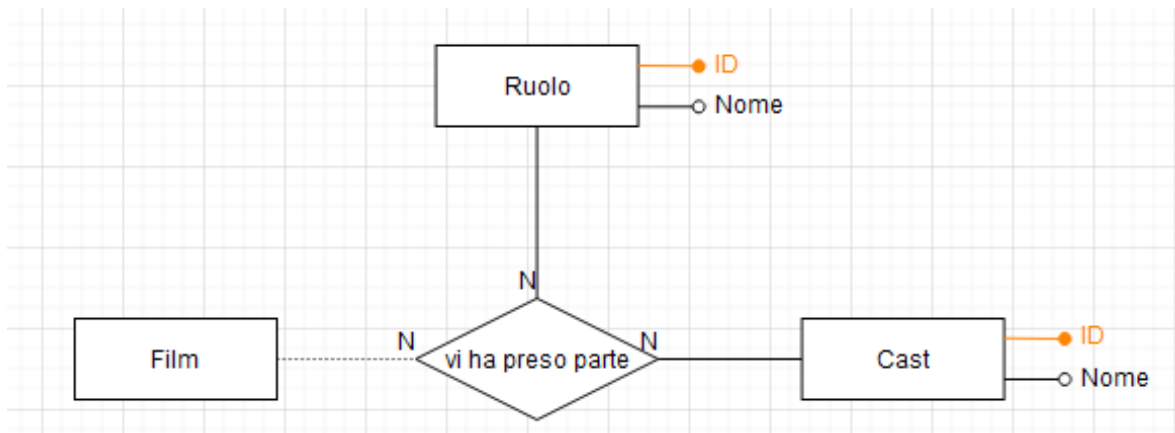


Il pattern **1NN0** nei **Film**



Nello schema dei **Film**, si può notare il pattern **1NN0**, in corrispondenza alla relazione **prodotto da**.

vi ha preso parte: una relazione **1NN0** ternaria

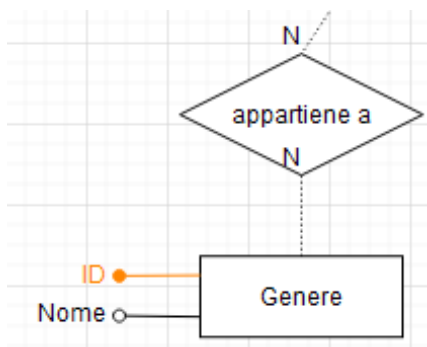


Nello schema dei film è presente la **relazione ternaria** **vi ha preso parte**.

Essa associa una persona (**Cast**) a un **Film**, specificando il **Ruolo** ("attore", "regista", "sceneggiatore"...) per cui vi ha preso parte.

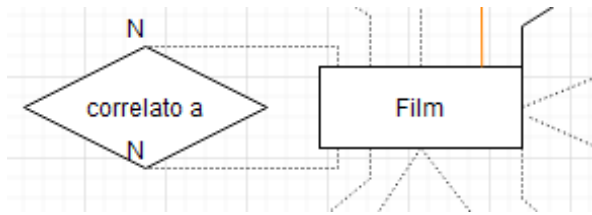
È stata modellata in questo modo per consentire query avanzate sul cast di un **Film**: ad esempio, "in quali film Quentin Tarantino ha avuto il ruolo di regista", oppure "che ruoli ha ricoperto Johnny Depp nei film usciti dopo il 2010".

Generi



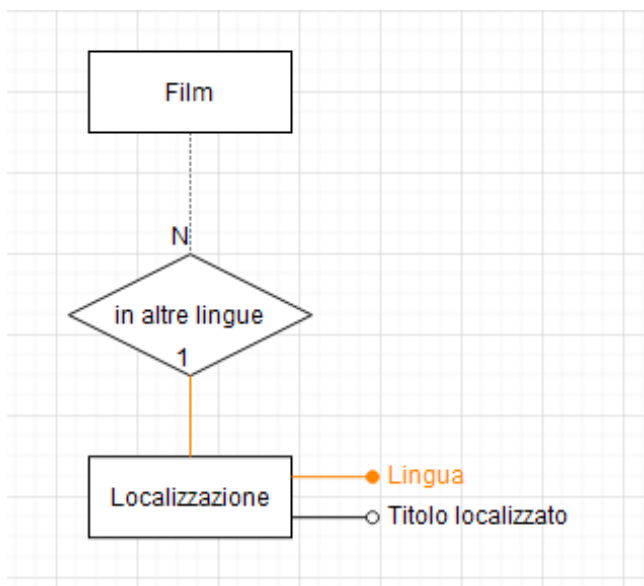
Come per i **Libri**, l'entità **Genere** è coinvolta in una **1NN0** con cardinalità **0 a N** dal lato del **Genere**.

Correlazioni



Come i **Libri**, anche i **Film** hanno un'autoassociazione per determinare le pellicole correlate.

Localizzazione



Simile al pattern **1NN0**, ma fondamentale è diversa è la relazione **in altre lingue**: essa ha cardinalità **0 a N** dal lato dei **Film**, e **1 a 1** dal lato dell'entità **Localizzazione**; in più, l'associazione è identificatore esterno di **Localizzazione**.

L'entità **Localizzazione** rappresenta il **titolo** di un **Film** tradotto in una lingua: ad esempio, *Il Padrino* è una localizzazione in "it" (italiano) del **Film** *The Godfather*.

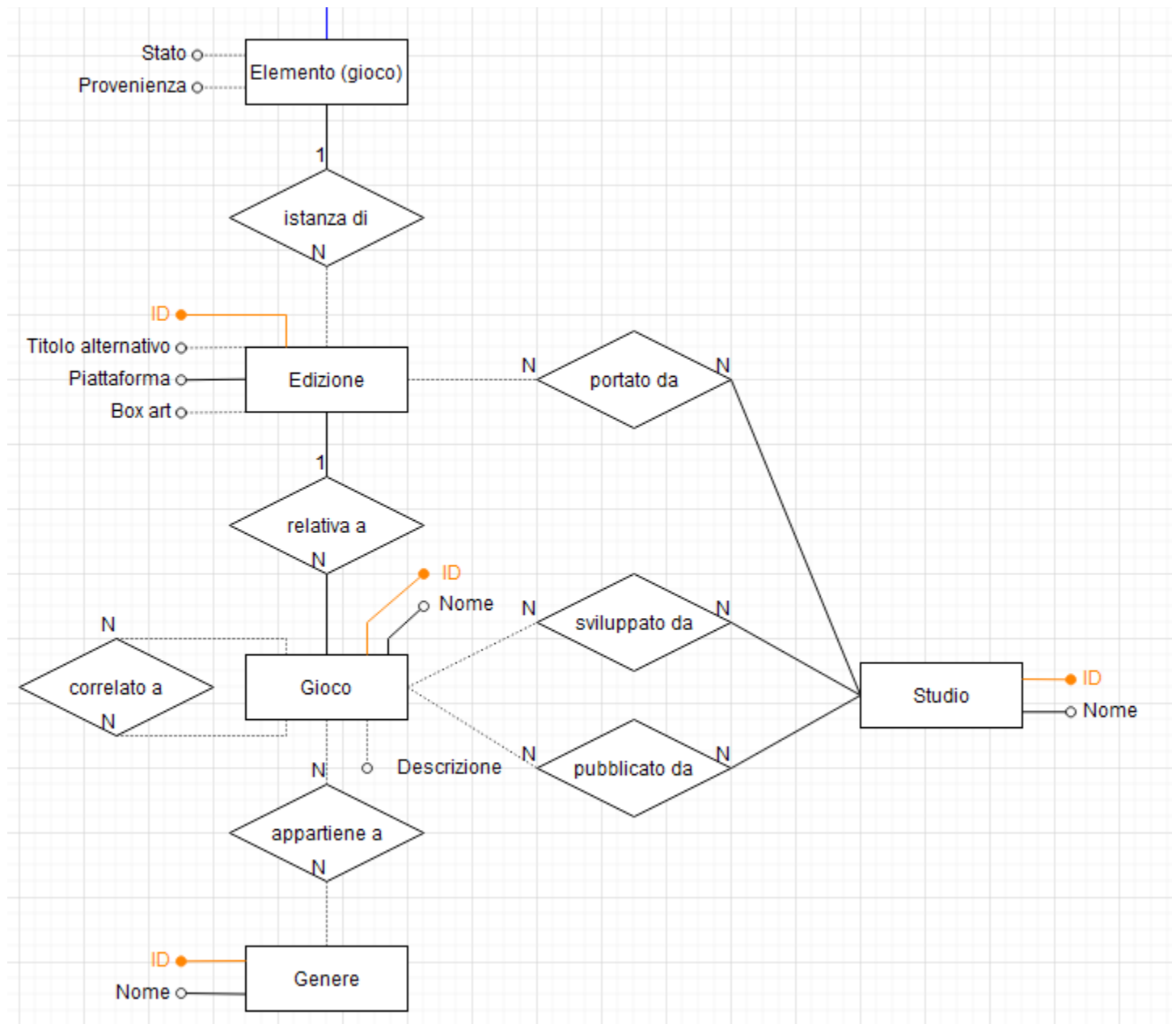
Il suo identificatore è composto dal codice ISO della lingua a cui si riferisce e dall'identificatore del **Film** a cui essa si riferisce: è dunque una **entità debole**.

Nei **Libri** e nei **Giochi** non è stato necessario creare questa entità perchè nei **Libri** lingue diverse hanno codici ISBN diversi (e quindi è possibile inserire i titoli localizzati all'interno dell'entità **Edizione**), e per i **Giochi** è molto raro avere titoli tradotti.

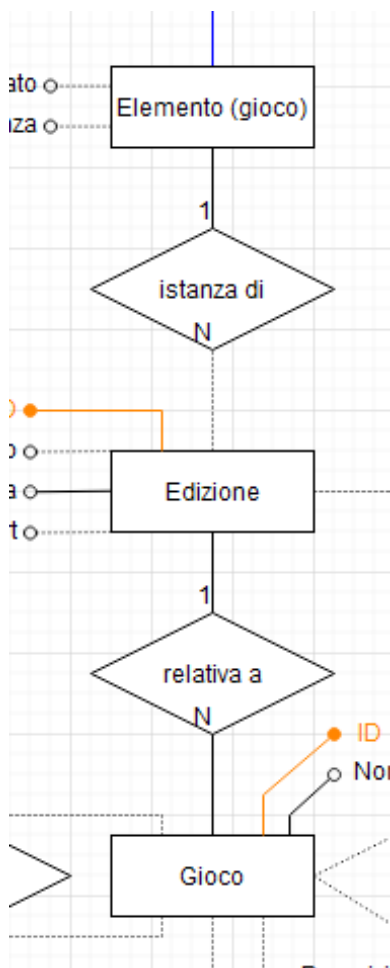
Giochi

Ogni videogioco avrà una sua pagina in cui sarà presente il titolo, lo sviluppatore, il publisher, una breve descrizione del gioco, l'elenco di tutte le piattaforme in cui esso è disponibile e, come per libri e film, un elenco di altri giochi correlati.

Per ogni piattaforma sarà disponibile una sottopagina, che conterrà la box art di quella versione, il nome dello studio che ha effettuato il porting ed eventualmente il titolo se diverso da quello principale.



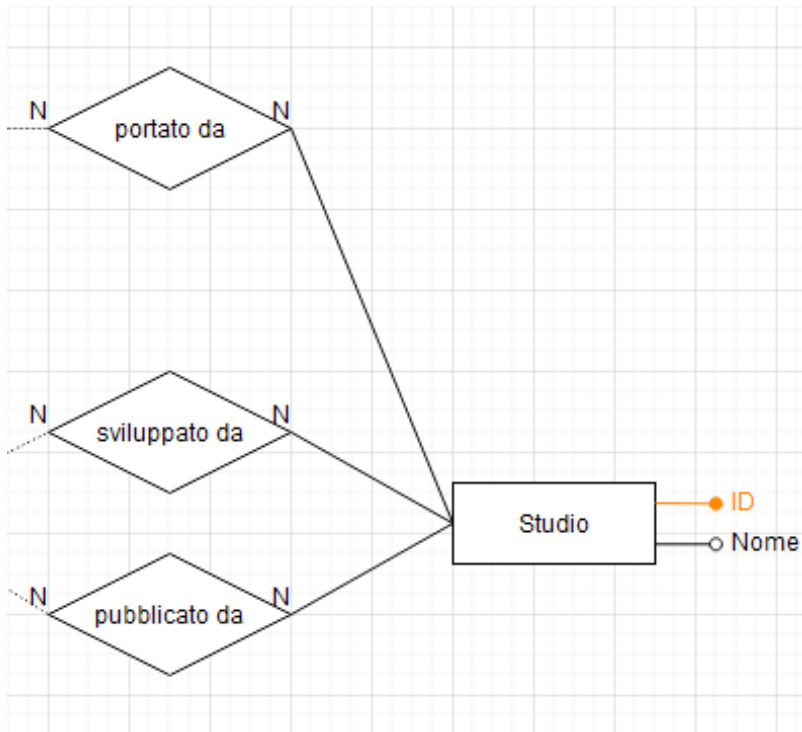
Edizioni



Quelle che nella descrizione vengono chiamate *sottopagine* sono state realizzate nello schema attraverso l'entità *Edizione*: una *Edizione* rappresenta una versione di un gioco pubblicata su una specifica piattaforma (PC, PS4, Xbox One, etc...).

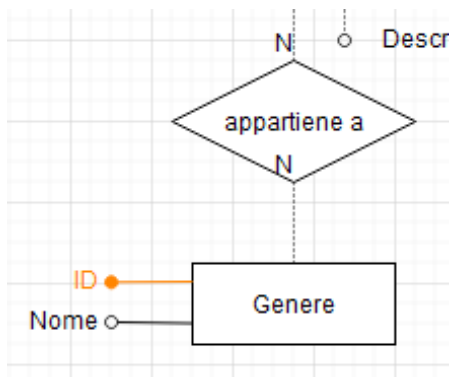
Come per gli *Elementi (libro)*, gli *Elementi (gioco)* saranno istanziati relativamente a una specifica *Edizione* di un gioco.

Il pattern **1NN0** nei Giochi



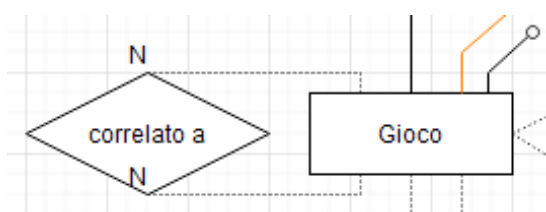
Il pattern **1NN0** è presente anche all'interno dello schema dei **Giochi**: si può notare nelle relazioni **sviluppato da**, **pubblicato da** e **portato da**.

Generi



Come i **Libri** e i **Film**, anche i **Giochi** hanno una relazione **appartiene a** che li collega a uno o più **Generi**.

Giochi correlati



Unimore | Basi di Dati | Chiara Calzolari e Stefano Pigozzi

Alexandria | Piattaforma per organizzare e condividere la propria libreria multimediale

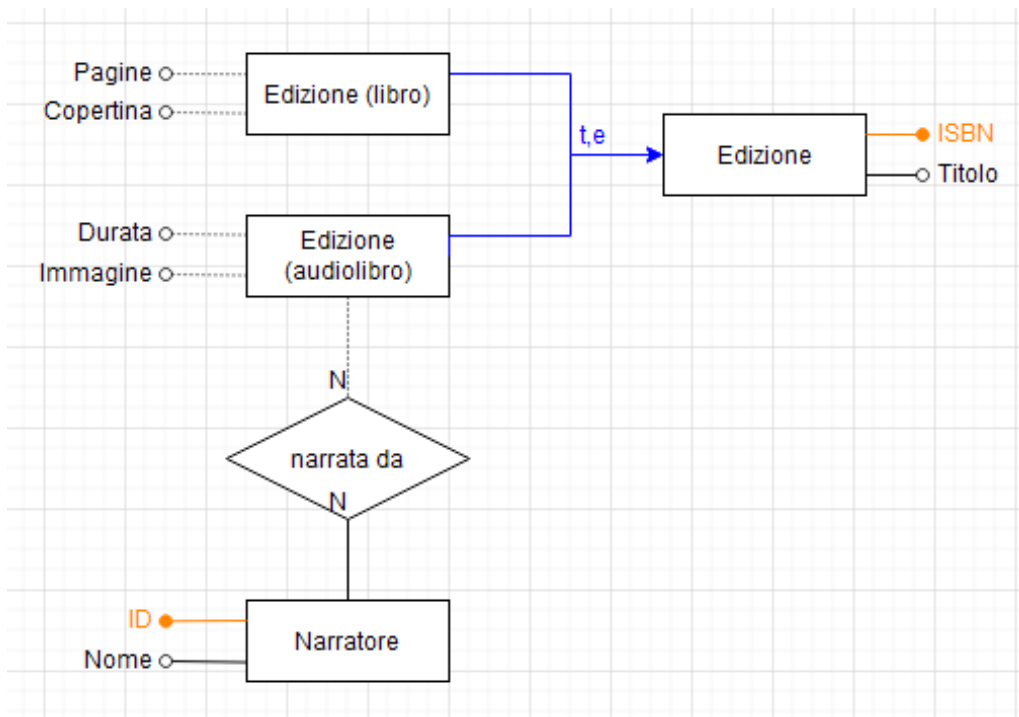
Ancora, anche i **Giochi** hanno un'autoassociazione per determinare le correlazioni tra di essi.

Eliminazione delle gerarchie

Per effettuare la progettazione logica è necessaria come prima cosa **eliminare le gerarchie ISA** createsi durante la fase di progettazione concettuale.

Si ricorda che in Alexandria sono presenti due gerarchie: una relativa agli **Elementi** e una relativa alle **Edizioni dei Libri**.

Gerarchia delle Edizioni

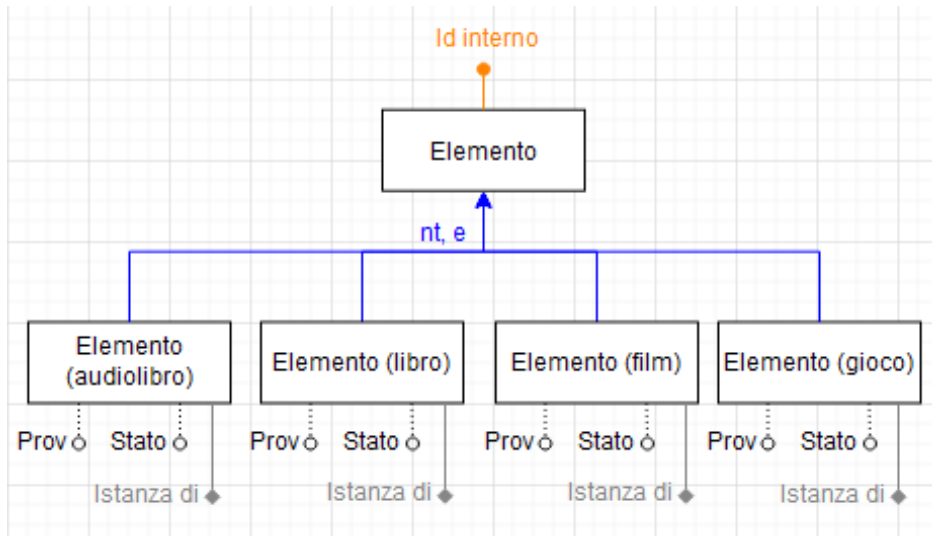


La gerarchia delle **Edizioni** è una gerarchia **totale ed esclusiva**; gli approcci possibili sono dunque 4:

- **Mantenimento delle entità:** Un'opzione sempre possibile, ma che porta a un numero di accessi maggiore e quindi a una velocità di interrogazione minore rispetto alle due alternative.
- **Collasso verso l'alto:** Essendo una relazione esclusiva, collassarla verso l'alto porterebbe ad *almeno* un 50% di valori nulli nella tabella **Edizione** risultante, portando a un significativo spreco di memoria.
- **Collasso verso l'alto con unione:** Sarebbe possibile effettuare il *collasso verso l'alto* unendo però le coppie di attributi {"Copertina", "Immagine"} e {"Pagine", "Durata"} (facendo riferimento rispettivamente alle pagine del libro e ai secondi dell'audiolibro) per rendere 0 i valori nulli introdotti nella tabella, a costo di parte dell'integrità della base di dati.
- **Collasso verso il basso:** Porterebbe alla duplicazione dell'entità **Edizione** e delle associazioni 1 a N con cardinalità N dal lato di **Edizione**; si creerebbe un nuovo tipo di **Elemento**, "Elemento (audiolibro)".

Si è deciso di **collassare la gerarchia verso il basso** perchè si è reputato importante mantenere integra, veloce e compatta la base di dati, anche a costo di maggiore complessità progettuale.

Gerarchia degli Elementi



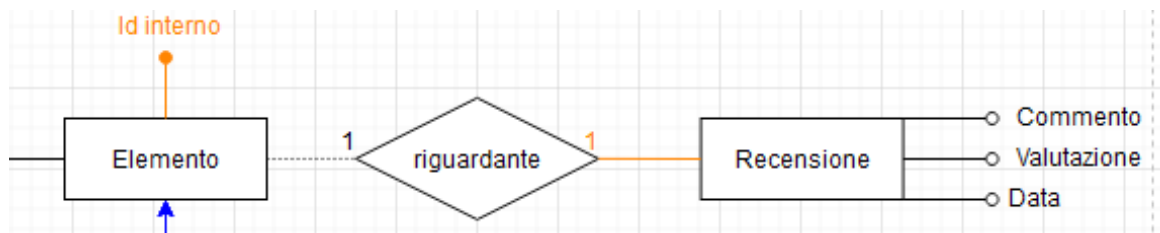
La gerarchia degli `Elementi` è **non totale** ed **esclusiva**. Possiamo dunque applicare i seguenti approcci:

- **Mantenimento delle entità:** Un'opzione sempre possibile, ma che porta a un numero di accessi maggiore e quindi a una velocità di interrogazione minore rispetto alle due alternative.
- **Collasso verso l'alto:** Collassare direttamente verso l'alto sarebbe l'opzione peggiore, in quanto aggiungerebbe 12 nuovi attributi con *almeno* il 75% dei valori nulli (più un selettore), sprecando più memoria del necessario.
- **Collasso verso l'alto con unione:** Per minimizzare i valori nulli, si potrebbero unire gli attributi {"Provenienza", "Stato"} in uno unico, verificando che vengano inseriti valori validi attraverso vincoli di integrità; in questo caso, si introdurrebbero solo 4 nuovi attributi ("istanza di"), con però comunque *almeno* il 75% di valori nulli, costando memoria (ma meno di quanta se ne sarebbe spesa in un normale collasso).
- **Collasso verso il basso:** Porterebbe alla creazione di quattro tabelle `Elemento` e a quattro tabelle `Recensione` relative a ogni tipo di `Elemento`, aumentando la complessità progettuale.

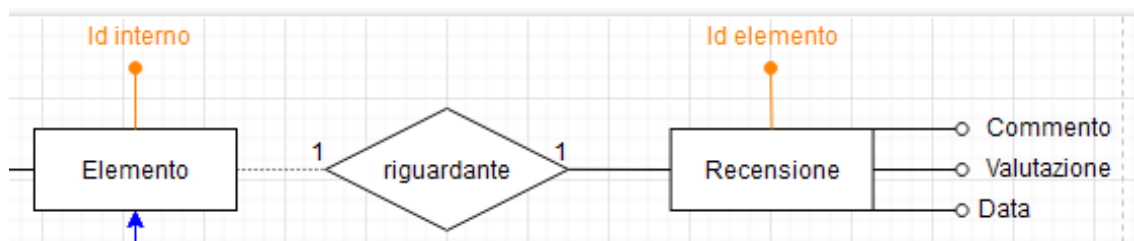
Anche qui si è deciso di **collassare la gerarchia verso il basso**, in quanto il collasso verso il basso favorisce le entità molto specializzate come gli `Elementi`; inoltre, adottare il collasso verso il basso permette di espandere con facilità il database in futuro per aggiungerci altri tipi di `Elementi`, in quanto basterà poi creare due nuove tabelle `Elemento` e `Recensione` per ogni tipo che si vorrà aggiungere. # Eliminazione delle chiavi esterne

In Alexandria sono presenti solo due **chiavi esterne**: una nelle entità `Recensione` e l'altra nell'entità `Localizzazione`.

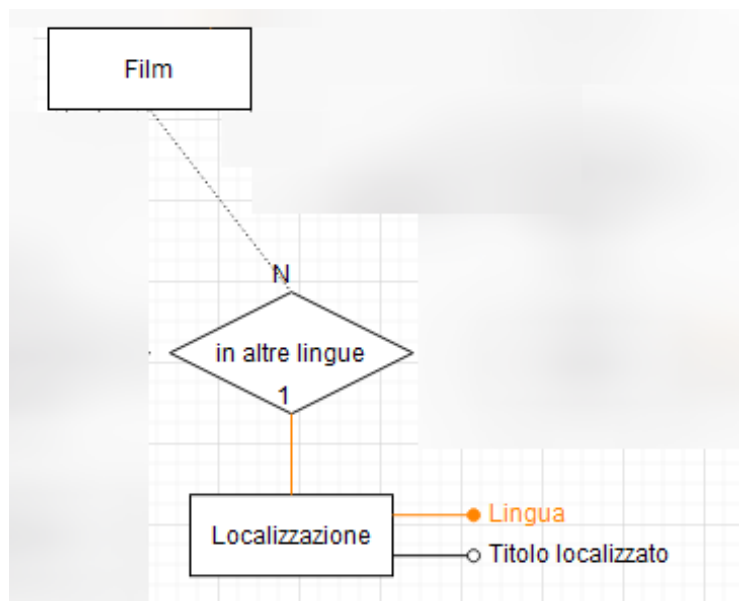
Eliminazione della chiave di Recensione



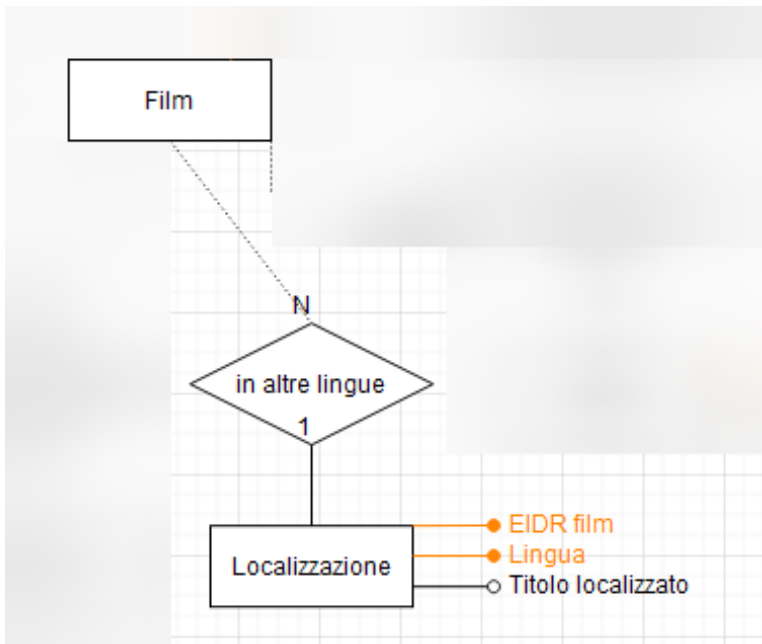
In questo caso, si è aggiunto a `Recensione` l'attributo "ID elemento", che corrisponderà all'ID dell'`Elemento` a cui si riferisce:



Eliminazione della chiave di Localizzazione



Anche in questo caso si è trasformata la chiave esterna in un attributo "EIDR film" che corrisponderà all'EIDR del film il cui titolo è stato tradotto.



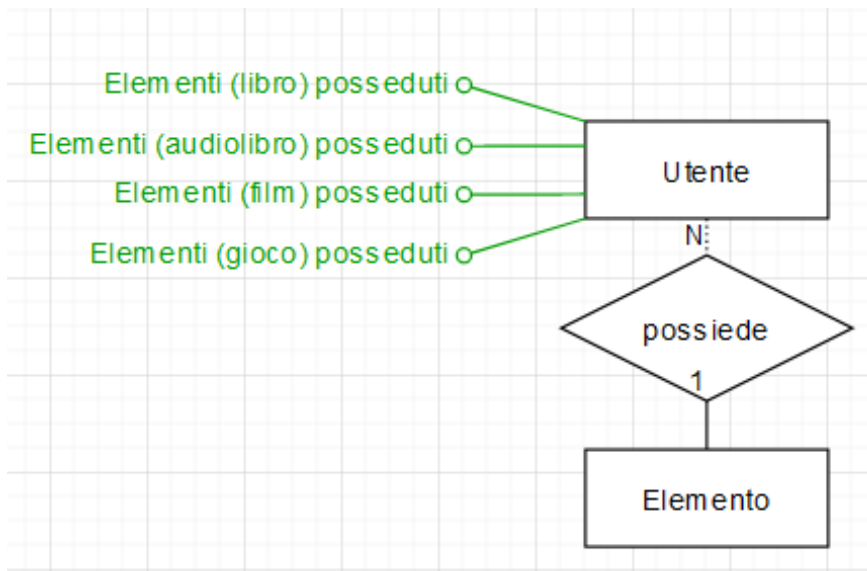
Trasformazione degli attributi composti

Lo schema relazionale di Alexandria non prevede alcun **attributo composto**, pertanto non c'è nulla da trasformare.

Dati derivati

In Alexandria non sono presenti molti dati quantitativi: la maggior parte degli attributi sono infatti qualitativi, e come tali meglio descritti da stringhe.

Si è considerata la possibilità di inserire nella base di dati un attributo contenente il **conteggio** di `Elementi` posseduti da un utente.



Si riportano sotto tutti i calcoli effettuati per decidere se aggiungere o no il dato derivato.

Tabella dei volumi

Secondo una stima effettuata, l'utente medio di Alexandria leggerà all'anno 18 libri, ascolterà 6 audiolibri, guarderà 52 film e giocherà a 24 giochi, per un totale di circa 100 `Elementi` per `Utente` all'anno.

Si ottiene, dunque, la seguente tabella dei volumi:

Concetto	Tipo	Volume
Utente	Entità	1/utente
Elemento (audiolibri)	Entità	6/utente/anno
Elemento (libro)	Entità	18/utente/anno
Elemento (film)	Entità	52/utente/anno
Elemento (gioco)	Entità	24/utente/anno

Si considera solo la categoria degli audiolibri: se per essi è conveniente mantenere il dato derivato, allora lo sarà anche per tutti gli altri tipi di elemento, in quanto hanno volumi maggiori.

Tabella delle operazioni

Si stima che ogni utente riceverà circa 500 visite alla suo profilo all'anno.

Si sono valutate due diverse operazioni:

Operazione	Descrizione	Tipo	Frequenza	Schema
OP1	Creazione di un nuovo Elemento (audiolibro)	Interattiva	6/utente/anno	Elemento (audiolibro)
OP2	Visualizzazione del conteggio di Elementi di qualsiasi tipo creati da un Utente specifico	Interattiva	500/utente/anno	Elemento (audiolibro)

Tabella dei costi

Si sono utilizzati i seguenti costi per realizzare la tabella dei costi: - **read**: costo 1 - **write**: costo 2 - **update**: composta da 1 read e 1 write, costo 3

Senza dato derivato

Operazione	Procedura	Costi	Costo totale
OP1	Si inserisce una nuova tupla nella tabella Elemento (audiolibro).	1 write	2
OP2	Si interroga la tabella Elemento (audiolibro), filtrando le tuple che non sono state create dall'utente desiderato, e si contano le tuple restituite.	6 read	6

Con dato derivato

Operazione	Procedura	Costi	Costo totale
OP1	Si inserisce una nuova tupla nella tabella Elemento (audiolibro) e si aggiorna il dato derivato della tupla dell'Utente creatore.	1 write + 1 update	5
OP2	Si va a vedere il dato derivato nella tabella dell'Utente desiderato.	1 read	1

Risultato

Metodo	Costo OP1	Costo OP2	Costo totale
Senza dato derivato	2 * 100	6 * 500	3200
Con dato derivato	5 * 100	1 * 500	1000

Dato che $1000 < 3200$, conviene **creare il dato derivato** nella tabella `Utente`.

La stessa cosa vale per gli `Elementi` di libri, film e giochi: avendo volumi maggiori, essi possono solo aumentare il costo dell'**OP2 senza dato derivato**. # Creazione dello schema logico

Di seguito si riporta l'intero schema logico dopo aver effettuato tutte le trasformazioni previste dalla fase di progettazione logica.

Legenda

- Entità
 - **Chiave primaria**
 - → Chiave esterna
 - *Opzionale*
 - ♦ Dato derivato

Categoria generale

Utente

- **Username**
- Password
- *Email*
- È amministratore
- È bannato
- ♦ Elementi (libro) posseduti
- ♦ Elementi (audiolibro) posseduti
- ♦ Elementi (film) posseduti
- ♦ Elementi (gioco) posseduti

Categoria condivisa tra libri e audiolibri

Libro

- **ID**
- Titolo primario
- *Sinossi*

Appartenenza a genere (libro)

- **ID Genere** → Genere (libro)
- **ID Libro** → Libro

Autore

- ID
- Nome

Correlazioni (libro)

- ID1 → Libro
- ID2 → Libro

Editore

- Parte ISBN
- Nome

Genere (libro)

- ID
- Nome

Scritto da

- ID Autore → Autore
- ID Libro → Libro

Categoria libri

Edizione (libro)

- ISBN
- Titolo
- *Pagine*
- *Copertina*
- Relativa a → Libro

Elemento (libro)

- ID
- Istanza di → Edizione (libro)
- Appartiene a → Utente
- *Stato*
- *Provenienza*

Recensione (libro)

- ID → Elemento (libro)
- Commento
- Valutazione
- Data

Schema degli audiolibri

Edizione (audiolibro)

- ISBN
- Titolo
- *Durata*
- *Immagine*
- Relativa a → Libro

Elemento (audiolibro)

- ID
- Istanza di → Edizione (audiolibro)
- Appartiene a → Utente
- *Stato*
- *Provenienza*

Narrata da

- ID Edizione → Edizione (audiolibro)
- ID Narratore → Narratore

Narratore

- ID
- Nome

Recensione (audiolibro)

- ID → Elemento (audiolibro)
- Commento
- Valutazione
- Data

Categoria film

Film

- EIDR
- Titolo
- *Sinossi*
- *Durata*
- *Locandina*

Appartenenza a genere (film)

- ID Genere → Genere
- EIDR → Film

Cast

- ID
- Nome

Correlazioni (film)

- EIDR1 → Film
- EIDR2 → Film

Elemento (film)

- ID
- *Stato*
- *Provenienza*
- Istanza di → Film
- Appartiene a → Utente

Genere (film)

- ID
- Nome

Localizzazione

- **Lingua**
- EIDR → Film
- Titolo localizzato

Prodotto da

- ID Studio → Studio
- EIDR → Film

Recensione (film)

- ID → Recensione (film)
- Commento
- Valutazione
- Data

Ruolo

- ID
- Nome

Studio (film)

- ID
- Nome

Vi ha preso parte

- EIDR → Film

- **ID Cast** → Cast
- **ID Ruolo** → Ruolo

Categoria giochi

Gioco

- **ID**
- Nome
- *Descrizione*

Appartenenza a genere (gioco)

- **ID Gioco** → Gioco
- **ID Genere** → Genere (gioco)

Correlazioni (gioco)

- **ID1** → Gioco
- **ID2** → Gioco

Edizione (gioco)

- **ID**
- *Titolo alternativo*
- Piattaforma
- *Box art*
- Relativa a → Gioco

Elemento (gioco)

- **ID**
- *Stato*
- *Provenienza*
- Istanza di → Edizione (gioco)
- Appartiene a → Utente

Genere (gioco)

- **ID**
- Nome

Portato da

- **ID Edizione** → Edizione (gioco)
- **ID Studio** → Studio (gioco)

Prodotto da

- **ID Gioco** → Gioco
- **ID Studio** → Studio (gioco)

Recensione (gioco)

- **ID** → Recensione (gioco)
- Commento
- Valutazione
- Data

Studio (gioco)

- **ID**
- Nome

Sviluppato da

- **ID Gioco** → Gioco
- **ID Studio** → Studio (gioco)

Verifica di normalizzazione

Dopo aver costruito lo schema logico, si è constatato che la base di dati fosse in forma normale, e che non fosse necessaria alcuna ulteriore modifica.

Tecnologia database

Database Management System

Si è scelto di utilizzare **PostgreSQL** (*Postgres*) come DBMS in quanto è quello con il quale i membri del gruppo erano più famigliari.

Strumenti aggiuntivi

Per assistere nella progettazione fisica, sono stati usati due strumenti per la manipolazione di database Postgres:

- pgAdmin 4
- Il plugin Database Tools and SQL per IDE basati su IntelliJ

Hosting

Per facilitare la collaborazione, si è scelto di condividere un'**istanza remota** di PostgreSQL installata su un server Ubuntu 18.04.4 LTS, alla quale ci si è connessi con username e password individuali.

```
lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.4 LTS
Release:       18.04
Codename:      bionic
```

```
SELECT version();
```

version

```
PostgreSQL 10.12 (Ubuntu 10.12-0ubuntu0.18.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0, 64-bit
```

Riproduzione del database

È allegato alla relazione il file `5-database.sql`, contenente tutte le istruzioni necessarie per ricreare il database descritto in questo capitolo.

Esso è stato ottenuto tramite `pg_dump`, un'utilità per l'archiviazione di database Postgres, eseguito con il seguente comando Bash:

```
pg_dump --dbname=alexandria --schema='public' --file="5-database.sql" --no-owner --no-acl
```

È possibile ricreare il database eseguendo manualmente tutte le istruzioni contenute nel file `.sql`, oppure eseguendo `pg_restore`, la controparte di `pg_dump` per il ripristino, con il seguente comando Bash:

```
pg_restore --dbname="alexandria" --schema="public" --file="5-database.sql"
```

Creazione database

Come primo passo della progettazione fisica si è creato un nuovo database su PostgreSQL attraverso il seguente comando Bash:

```
createdb alexandria
```

Esso è equivalente alla seguente istruzione SQL:

```
CREATE DATABASE alexandria;
```

Creazione tabelle

Dopo aver creato il database, il secondo passo della progettazione fisica è stato quello di convertire lo schema logico in un database Postgres.

In generale:

- Le **entità** sono diventate *TABLES* (tabelle);
- Gli **attributi opzionali** sono diventati *COLUMNS* (colonne);
- Gli **attributi obbligatori** sono diventati *COLUMNS* con il vincolo *NOT NULL*;
- Le **chiavi primarie** sono state implementate come *PRIMARY KEYS* (chiavi primarie);
- Le **chiavi esterne** sono state implementate come *FOREIGN KEYS* (chiavi esterne);
- Le **chiavi surrogate** sono state implementate come *PRIMARY KEYS* autoincrementate tramite *SEQUENCES* (sequenze);
- I **dati derivati** sono stati implementati come *COLUMNS* aventi dei *TRIGGER* che le aggiornino.

Schema dei nomi delle tabelle

Tutte le tabelle sono state istanziate con il nome che le corrispondenti entità avevano nello schema logico, sostituendo tutte le lettere maiuscole con **lettere minuscole** a-z, spazi con **underscore** `_` e rimuovendo le parentesi con il loro contenuto.

Inoltre, a tutte le tabelle tranne `utente` è stato dato un nome prefissato da `libro_`, `audiolibro_`, `film_` e `gioco_` per indicare la categoria a cui le entità appartenevano nello schema logico.

Esempi

Entità

Utente
Libro
Edizione (libro)
Cast

Tabella

utente
libro
libro_edizione
film_cast

Creazione tabelle

Si riportano solo le tabelle con qualche particolarità; le tabelle per la quale la conversione è banale sono omesse da questo file (ma non dal file `5-database.sql`).

`audiolibro_edizione`

```
CREATE TABLE public.audiolibro_edizione (
  isbn character(13) NOT NULL,
  titolo character varying NOT NULL,
  durata interval,
  immagine bytea,
  relativa_a integer NOT NULL,
  CONSTRAINT durata_check CHECK ((date_part('epoch'::text, durata) >= (0)::double
precision))
);
```

```
ALTER TABLE ONLY public.audiolibro_edizione
```

```
ADD CONSTRAINT audiolibro_edizione_pkey PRIMARY KEY (isbn);
```

La durata delle edizioni degli audiolibri è di tipo *interval*, che rappresenta un intervallo di tempo con la precisione di centesimi di secondo; l'immagine dell'audiolibro ha invece tipo *bytea*, e sarà salvata nel database come un blob binario di dati.

Inoltre, la durata è dotata di un *CHECK* che impedisce che essa sia minore di 0, convertendo l'*interval* in secondi e controllando che essi siano maggiori di 0.

audiolibro_recensione

```
CREATE TABLE public.audiolibro_recensione (
  id bigint NOT NULL,
  commento text NOT NULL,
  valutazione smallint NOT NULL,
  data timestamp without time zone DEFAULT now() NOT NULL,
  CONSTRAINT audiolibro_recensione_valutazione_check CHECK (((valutazione >= 0) AND
(valutazione <= 100)))
);
```

```
ALTER TABLE ONLY public.audiolibro_recensione
  ADD CONSTRAINT audiolibro_recensione_pkey PRIMARY KEY (id);
```

```
ALTER TABLE ONLY public.audiolibro_recensione
  ADD CONSTRAINT id FOREIGN KEY (id) REFERENCES public.audiolibro_elemento(id);
```

Per minimizzare valori nulli, si è tradotta l'associazione binaria 1 a 1 *relativa* a in due tabelle separate, usando in una delle due una chiave esterna come chiave primaria.

Si notino dunque i due *CONSTRAINT* *audiolibro_recensione_pkey* e *id*.

La valutazione delle recensioni deve essere obbligatoriamente tra 0 e 100: a tale scopo, è stato introdotto un *CHECK* sulla tabella che verifichi questa condizione.

La data di pubblicazione è rappresentata da un *timestamp*, tipo che rappresenta un istante specifico di tempo con precisione fino ai microsecondi.

film

```
CREATE TABLE public.film (
  eidr character(34) NOT NULL,
  titolo character varying NOT NULL,
  sinossi text,
  locandina bytea,
  durata interval,
  CONSTRAINT durata_check CHECK ((date_part('epoch'::text, durata) > (0)::double
precision))
);
```

```
ALTER TABLE ONLY public.film
  ADD CONSTRAINT film_pkey PRIMARY KEY (eidr);
```

Nella tabella *film* compare nuovamente il *CHECK* utilizzato nelle *audiolibro_edizioni* per la durata, e lo stesso tipo *bytea* per la locandina.

Si può notare nella tabella compare il tipo *char(34)*: essendo gli EIDR sempre lunghi 34 caratteri, si è scelto di minimizzare lo spazio di archiviazione utilizzando rendendo il campo a lunghezza non variabile.

film_correlazioni

```
CREATE TABLE public.film_correlazioni (  
    eidr_1 character(34) NOT NULL,  
    eidr_2 character(34) NOT NULL  
);  
  
ALTER TABLE ONLY public.film_correlazioni  
    ADD CONSTRAINT film_correlazioni_pkey PRIMARY KEY (eidr_1, eidr_2);  
  
ALTER TABLE ONLY public.film_correlazioni  
    ADD CONSTRAINT eidr_1 FOREIGN KEY (eidr_1) REFERENCES public.film(eidr);  
  
ALTER TABLE ONLY public.film_correlazioni  
    ADD CONSTRAINT eidr_2 FOREIGN KEY (eidr_2) REFERENCES public.film(eidr);
```

L'autoassociazione delle correlazioni è stata implementata attraverso una **tabella ponte** che collega due film attraverso i loro `eidr`.

`eidr_1` ed `eidr_2` sono due chiavi esterne separate, e insieme formano la **chiave primaria composta** della tabella.

film_vi_ha_preso_parte

```
CREATE TABLE public.film_vi_ha_preso_parte (  
    eidr character(34) NOT NULL,  
    id_cast integer NOT NULL,  
    id_ruolo integer NOT NULL  
);  
  
ALTER TABLE ONLY public.film_vi_ha_preso_parte  
    ADD CONSTRAINT film_vi_ha_preso_parte_pkey PRIMARY KEY (eidr, id_cast, id_ruolo);  
  
ALTER TABLE ONLY public.film_vi_ha_preso_parte  
    ADD CONSTRAINT eidr FOREIGN KEY (eidr) REFERENCES public.film(eidr);  
  
ALTER TABLE ONLY public.film_vi_ha_preso_parte  
    ADD CONSTRAINT id_cast FOREIGN KEY (id_cast) REFERENCES public.film_cast(id);  
  
ALTER TABLE ONLY public.film_vi_ha_preso_parte  
    ADD CONSTRAINT id_ruolo FOREIGN KEY (id_ruolo) REFERENCES public.film_ruolo(id);
```

L'associazione ternaria è stata realizzata con un'altra **tabella ponte**, avente una **chiave primaria composta** e tre chiavi esterne separate.

elemento_id_seq

```
CREATE SEQUENCE public.elemento_id_seq  
    START WITH 1  
    INCREMENT BY 1  
    NO MINVALUE  
    NO MAXVALUE  
    CACHE 1;
```

Si è deciso di rendere **univoci** gli `id` di **tutti gli elementi**, qualsiasi fosse il loro tipo.

Per realizzare ciò si è creata una *SEQUENCE* unica che viene poi usata da tutte le tabelle *_elemento per generare i nuovi id.

Stato e provenienza

```
CREATE TYPE public.gioco_provenienza AS ENUM (
    'GRATUITO',
    'ACQUISTATO',
    'IN_ABBONAMENTO',
    'PRESO_IN_PRESTITO',
    'NON_PIU_POSSEDUTO',
    'ALTRO'
);
```

```
CREATE TYPE public.gioco_stato AS ENUM (
    'DA_INIZIARE',
    'INIZIATO',
    'FINITO',
    'COMPLETATO',
    'NON_APPLICABILE'
);
```

Gli stati e le provenienze dei vari elementi sono state realizzate creando *ENUM* contenenti tutte le possibili opzioni selezionabili dall'utente, e utilizzandoli come tipo delle relative colonne; in questo modo, si impedisce l'immissione di valori non validi nelle colonne.

gioco_elemento

```
CREATE TABLE public.gioco_elemento (
    id bigint DEFAULT nextval('public.elemento_id_seq'::regclass) NOT NULL,
    stato public.gioco_stato,
    provenienza public.gioco_provenienza,
    istanza_di integer NOT NULL,
    appartiene_a character varying NOT NULL
);
```

```
ALTER TABLE ONLY public.gioco_elemento
    ADD CONSTRAINT gioco_elemento_pkey PRIMARY KEY (id);
```

```
ALTER TABLE ONLY public.gioco_elemento
    ADD CONSTRAINT appartiene_a FOREIGN KEY (appartiene_a) REFERENCES
public.utente(username);
```

```
ALTER TABLE ONLY public.gioco_elemento
    ADD CONSTRAINT istanza_di FOREIGN KEY (istanza_di) REFERENCES public.gioco_edizione(id);
```

```
CREATE FUNCTION public.update_n_giochi() RETURNS trigger
LANGUAGE plpgsql
AS $$BEGIN
    IF (TG_OP = 'DELETE') THEN
        UPDATE utente
            SET gioco_elementi_posseduti = gioco_elementi_posseduti - 1
            WHERE utente.username = old.appartiene_a;
```

```
        RETURN old;
    ELSIF (TG_OP = 'INSERT') THEN
        UPDATE utente
            SET gioco_elementi_posseduti = gioco_elementi_posseduti + 1
            WHERE utente.username = new.appartiene_a;
        RETURN new;
    END IF;
END;
$$;
```

```
CREATE TRIGGER numero_giochi_trigger BEFORE INSERT OR DELETE
    ON public.gioco_elemento
    FOR EACH ROW EXECUTE PROCEDURE public.update_n_giochi();
```

Le colonne `stato` e `provenienza` utilizzano il relativo *ENUM* creato in precedenza.

La colonna `id` ha un valore di *DEFAULT* particolare: `nextval('public.elemento_id_seq'::regclass)`. Ciò significa che, se non viene specificato un `id` durante un inserimento, alla riga verrà assegnato automaticamente il valore corrente della *SEQUENCE* `public.elemento_id_seq` descritta in precedenza, aumentandone inoltre il valore di 1.

Nella tabella è presente anche un *TRIGGER*, che incrementa o decrementa il conteggio degli elementi dell'utente a cui essi appartengono quando uno o più elementi vengono inseriti o rimossi.

libro_edizione

```
CREATE FUNCTION public.is_numeric(text character varying) RETURNS boolean
    LANGUAGE plpgsql STRICT
    AS $_$DECLARE x NUMERIC;
    BEGIN
        x = $1::NUMERIC;
        RETURN TRUE;
    EXCEPTION WHEN others THEN
        RETURN FALSE;
    END;$_$;
```

```
CREATE TABLE public.libro_edizione (
    isbn character(13) NOT NULL,
    titolo_edizione character varying NOT NULL,
    pagine integer,
    copertina bytea,
    relativa_a integer NOT NULL,
    CONSTRAINT libro_edizione_isbn_check CHECK ((public.is_numeric(("substring"((isbn)::text,
1, 12))::character varying) AND (public.is_numeric(("right"((isbn)::text, 1))::character
varying) OR ("right"((isbn)::text, 1) ~~ '%X'::text)))),
    CONSTRAINT libro_edizione_pagine_check CHECK ((pagine >= 0))
);
```

```
ALTER TABLE ONLY public.libro_edizione
    ADD CONSTRAINT libro_edizione_pkey PRIMARY KEY (isbn);
```

```
ALTER TABLE ONLY public.libro_edizione
```

ADD CONSTRAINT relativa_a FOREIGN KEY (relativa_a) REFERENCES public.libro(id);
La tabella delle edizioni di un libro include due *CHECK*: uno che controlla che le pagine, se specificate, siano un numero positivo, e un'altro che controlla che gli ISBN siano in un formato valido.

In particolare, per quest'ultimo, è stata creata una funzione di utilità `is_numeric`, che verifica che tutti i caratteri di una stringa siano numerici: questa funzione viene poi usata per controllare che tutti i caratteri dell'ISBN siano numeri, permettendo però anche una `X` in ultima posizione (l'ultima cifra degli ISBN più vecchi era in base-11 e utilizzava la lettera X come 10).

utente

```
CREATE TABLE public.utente (  
    username character varying NOT NULL,  
    password bytea NOT NULL,  
    email character varying,  
    is_admin boolean DEFAULT false NOT NULL,  
    is_banned boolean DEFAULT false NOT NULL,  
    libro_elementi_posseduti integer DEFAULT 0 NOT NULL,  
    audiolibro_elementi_posseduti integer DEFAULT 0 NOT NULL,  
    film_elementi_posseduti integer DEFAULT 0 NOT NULL,  
    gioco_elementi_posseduti integer DEFAULT 0 NOT NULL  
);
```

```
ALTER TABLE ONLY public.utente  
    ADD CONSTRAINT username PRIMARY KEY (username);
```

La password, essendo un hash, è rappresentata come un dato binario (*bytea*).

Le colonne `is_admin` e `is_banned` hanno un valore di default di *false*, in quanto alla creazione gli utenti non saranno amministratori o bannati.

Le colonne `*_elementi_posseduti` sono i dati derivati che rappresentano quanti elementi di ogni tipo possiede un dato utente: per i nuovi utenti, questo valore sarà 0.

Queste colonne saranno poi incrementate dai trigger presenti nelle quattro tabelle `*_elemento`. # Query preprogrammate per l'utilizzo del database

Si sono inserite in questo capitolo della relazione alcuni esempi di query che permetteranno al sito web di interagire con la base di dati.

Come nel caso della creazione tabelle, si elencano solo le query più significative.

Creazione di un nuovo utente

```
INSERT INTO utente (username, password, email) VALUES ($username, $hashed_password, $email);
```

Promozione di un utente ad amministratore

```
UPDATE utente SET is_admin = true WHERE username = $username;
```

Creazione di un elemento relativo a un libro non esistente nel database

```
INSERT INTO libro (titolo_primario) VALUES ($titolo);
```

```
INSERT INTO libro_edizione (isbn, titolo_edizione, relativa_a) VALUES ($isbn, $titolo, currval('libro_id_seq'));
```

```
INSERT INTO libro_elemento (istanza_di, appartiene_a) VALUES ($isbn, $username);
```

Creazione di una recensione relativa a un elemento

```
INSERT INTO libro_recensione (id, commento, data, valutazione) VALUES ($isbn, $commento, now(), $valutazione);
```

Conteggio degli elementi posseduti da un utente

```
SELECT
    libro_elementi_posseduti libri,
    audiolibro_elementi_posseduti audiolibri,
    film_elementi_posseduti film,
    gioco_elementi_posseduti giochi,
    (libro_elementi_posseduti + audiolibro_elementi_posseduti + film_elementi_posseduti +
    gioco_elementi_posseduti) totale
FROM utente
WHERE username = $username;
```

Conteggio del numero totale di edizioni di libri presenti nel database

```
SELECT COUNT(*) FROM libro_edizione;
```

Conteggio del numero totale di edizioni di libri di ogni autore

```
SELECT COUNT(*)
FROM libro_edizione
WHERE relativa_a IN (
    SELECT l.id
    FROM libro l
    JOIN libro_scritto_da lsd on l.id = lsd.id_libro
    JOIN libro_autore la on lsd.id_autore = la.id
    WHERE la.id = $id_autore
);
```

Visualizzazione della valutazione media di un gioco

```
SELECT AVG(gr.valutazione)
FROM gioco_recensione gr
JOIN gioco_elemento gel on gr.id = gel.id
JOIN gioco_edizione ge on gel.istanza_di = ge.id
JOIN gioco g on ge.relativa_a = g.id
WHERE g.id = $id_gioco;
```

Visualizzazione di tutte le edizioni di un dato editore

```
SELECT *
  FROM libro_editore ld
  JOIN libro_edizione lz
    ON ld.parte_isbn = substr(lz.isbn, 6, length(ld.parte_isbn))
 WHERE ld.nome = $nome_editore;
```