



UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA



Dipartimento di Ingegneria e Scienze
dell'Informazione e Matematica

Università degli Studi dell'Aquila



Documentazione di Progetto

Laboratorio di Basi di Dati

DOCENTE : **Pelliccione Patrizio**

PROGETTO REALIZZATO DAL GRUPPO 1

Federico Di Menna *Matr. 253962*

Fulvio Lapenna *Matr. 252334*



Contratto di gruppo

<https://docs.google.com/document/d/1oP6d0y9gpyMqvICHjMbOHA0C3f8jk7ev5TP2UwZsIQw/edit?usp=sharing>

Introduzione

Si vuole realizzare una base di dati volta alla gestione di un catalogo bibliografico online.

Tale base di dati sarà a disposizione, principalmente, di tutti quegli applicativi che necessitano di gestire l'inserimento, la cancellazione e le modifiche sui dati delle pubblicazioni da parte dei propri utenti. La base di dati **non** gestisce per ragioni di efficienza il rollback di modifiche effettuate sulle pubblicazioni ma soltanto la possibilità di consultare queste ultime grazie a uno storico, quindi si consiglia di prestare particolare attenzione ogni qualvolta venga effettuata una transazione sul database, e di gestire opportunamente tale situazione a livello applicativo. Nel seguito della documentazione si parlerà spesso di “modifiche” di un pubblicazione riferendosi al singolo elemento dello storico, questo termine generalizza tutte le azioni fatte da un utente su una pubblicazione, quindi aggiunta, modifica degli attributi, eliminazione, aggiunta o eliminazione di like e recensioni ecc...

Le pubblicazioni non vengono inserite interamente in formato elettronico all'interno della base di dati ma sono schede bibliografiche rappresentate dall'entità *Pubblicazione* che ne possiede gli attributi caratterizzanti. Gli utenti invece sono rappresentati dall'entità *Utente* che ovviamente deve contenere tutte le proprietà degli utilizzatori (per un buon grado di tracciabilità è quindi opportuno che ogni utente abbia un account proprio legato alle sue *capabilities* mediante i ruoli gestiti dal database).

Oltre ad essere usufruibile da parte di biblioteche o associazioni che si occupano di gestire la suddetta attività, la base di dati amministra anche l'interazione con utenti che non si occupano del management del catalogo ma di fornire feedback sulle pubblicazioni (garantendo così la possibilità di utilizzare il database su applicazioni di tipo forum o social). Infatti gli utenti base possono ricevere una promozione da un utente di livello superiore e entrare nello staff di management del catalogo.

Gli applicativi che vorranno utilizzare tale funzionalità dovranno implementare operazioni di registrazione e accesso anche degli “utenti utilizzatori” visto che il sistema esige di tali dati. Infatti un utente *anonimo* (ovvero un qualsiasi visitatore che non ha effettuato l'accesso sul sistema) è in grado esclusivamente di consultare il catalogo senza essere in grado di interagire con esso.

Da un'accurata analisi della specifica fornita, abbiamo individuato i principali attori e servizi richiesti da essa, e abbiamo inserito e/o vincolato alcune funzionalità che fruivano di ambiguità.

La base di dati è, infatti, composta da 3 macro-sezioni:

- dati relativi alle **pubblicazioni**
- dati relativi agli **utenti**
- dati relativi alle **interazioni** fra utenti e pubblicazioni (*likes, recensioni e storico*)

Successivamente sono descritte tutte le decisioni prese allo scopo di rendere efficiente, funzionale e consistente il nostro sistema.

INDICE

ANALISI DELLA SPECIFICA

Glossario dei Termini	5
Descrizione dettagliata dei Termini	6
Operazioni Richieste	7
Altre operazioni	8

PROGETTAZIONE CONCETTUALE

Dettagli relativi alla scelta delle Entità	8
Utente	8
Pubblicazione	9
Autore	10
Sorgente	10
Storia	11
Recensione	11
Dettagli relativi alla scelta delle Relationship	12
Scritto (Pubblicazione - Autore)	12
Associata (Pubblicazione - Recensione)	13
Composizione (Pubblicazione - Sorgente)	13
Conservazione (Pubblicazione - Storia)	13
Like (Pubblicazione - Utente)	14
Registrazione (Storia - Utente)	14
Inserimento (Recensione - Utente)	14
Approvazione (Recensione - Moderatore)	15
Scelte progettuali relative allo schema ER	15
Schema E-R	17
Descrizione dello Schema	18
Dizionario delle Entità	18
Dizionario delle Relationship	19
Descrizione degli attributi	20
Tavola dei Volumi	22



Tavola degli accessi	23
PROGETTAZIONE LOGICA	24
Analisi delle ridondanze	24
Analisi attributo NumLike	25
Analisi attributo NumRec	26
Analisi attributo NumPubb	27
Analisi attributo DataUltimaRecensione	28
Analisi attributo DataUltimaModifica	29
Analisi attributo DataUltimaRistampa	30
Scelte relative alla ristrutturazione dello schema ER	31
Schema E-R Ristrutturato	32
Dizionario delle Entità (dopo la ristrutturazione)	33
Dizionario delle Relationships (dopo la ristrutturazione)	34
Descrizione degli Attributi	35
Schema Logico	38
Vincoli	39
Vincoli di integrità referenziale	39
Vincoli non esprimibili	40
Vincoli di dominio	40
PROGETTAZIONE FISICA	
Tecnologie Utilizzate	42
Schema Fisico	42
Scelte relative all'implementazione	43
Creazione del Database	44
Implementazioni delle operazioni richieste	49
Implementazione delle operazioni aggiuntive	63
Divisione del lavoro	74



ANALISI DELLA SPECIFICA

Glossario dei Termini

TERMINI	DESCRIZIONE	SINONIMI	COLLEGAMENTI
Pubblicazione	Riproduzione attraverso la stampa di scritti e/o immagini destinati alla diffusione e in genere alla vendita	Libro Scheda Bibliografica Opera	<i>Utente, Sorgente, Recensione, Storia, Like, Metadati</i>
Metadati	Attributi specifici di una pubblicazione	-	<i>Pubblicazione</i>
Utente	Utilizzatore del sistema	Moderatore Amministratore Utilizzatore	<i>Recensione, Like, Pubblicazione, Storia</i>
Recensione	Testo valutativo e interpretativo di una pubblicazione da parte di un utente	-	<i>Pubblicazione, Utente</i>
Like	Valutazione positiva relativa ad una pubblicazione da parte di un utente	-	<i>Pubblicazione, Utente</i>
Sorgente	Risorse esterne relative ad una pubblicazione	-	<i>Pubblicazione</i>
Storia	Elenco di tutte le <i>modifiche</i> effettuate relative alle schede bibliografiche	Storico	<i>Pubblicazione, Utente</i>
Ruolo	Tipologia di utenza presente nel sistema	-	<i>Utente</i>

Descrizione dettagliata dei Termini

Pubblicazione

Una pubblicazione è una scheda bibliografica di un libro, opera letteraria, e tutto quello che può essere identificato mediante codice ISBN.

Le pubblicazioni vengono inserite nel catalogo dagli utenti (di grado moderatore o superiore).

Utente

Con il termine utente indichiamo qualsiasi tipo di utilizzatore del sistema. Gli utenti base possono consultare il catalogo e inserire recensioni e like. I gestori possono in aggiunta inserire o modificare le pubblicazioni. Gli utenti base si suddividono in attivi e passivi in base alle azioni che effettuano nel corso del tempo (e non hanno privilegi differenti, la distinzione è a scopo statistico). Invece gli utenti gestori, che ovviamente sono considerati attivi dal sistema, sono di tipo moderatore, amministratore e super amministratore. Ogni ruolo ha i propri privilegi, in aggiunta a quelli delle categorie sottostanti.

Recensione

Una recensione è un breve testo di carattere qualitativo che un utente associa ad una pubblicazione nel catalogo. Le recensioni dovranno essere approvate da parte di un utente moderatore prima di essere pubblicamente visibili. Ogni recensione è associata alla data e ora di inserimento.

Like

I like vengono assegnati dagli utenti alle pubblicazioni nel catalogo. Per cui ogni like è rappresentato come una relazione tra utente e pubblicazione. Ogni pubblicazione può ricevere un solo like per ogni utente.

Sorgente

Le sorgenti legate ad una pubblicazione rappresentano qualsiasi tipo di risorsa digitale (immagini, link per download, ecc...) e sono associate al proprio URL.

Metadati

I metadati di una pubblicazione rappresentano l'insieme dei dati aggiuntivi associati ad una pubblicazione (ISBN, numero di pagine, lingua, data di pubblicazione, lista delle ristampe, parole chiave identificative).

Storia

Tiene traccia di tutte le modifiche fatte dagli utenti al catalogo bibliografico.

Ogni entry della storia è associata ad una breve descrizione della modifica e la data e l'ora di quando essa è stata effettuata (scritta sul database).

Ruolo

Il ruolo indica la tipologia di utenza riferita ad un particolare utente con i vari privilegi permessi. I ruoli da noi proposti sono : attivo, passivo, moderatore, amministratore, superamministratore.

Operazioni Richieste

1. Modifica del livello di un utente (da *attivo* a *passivo* e viceversa).
2. Estrazione elenco delle ultime dieci pubblicazioni inserite.
3. Estrazione elenco delle pubblicazioni aggiornate negli ultimi 30 giorni.
4. Estrazione elenco dei 30 utenti più “collaborativi” (cioè quelli che hanno inserito più pubblicazioni).
5. Estrazione elenco delle pubblicazioni inserite da un utente.
6. Estrazione catalogo cioè elenco di tutte le pubblicazioni con Titolo, Editore, Autore e anno di pubblicazione ordinato per titolo.
7. Estrazione dati completi di una pubblicazione specifica, dato il suo ID.
8. Ricerca di pubblicazioni per ISBN, titolo, autore e parole chiave.
9. Inserimento di una recensione relativa a una pubblicazione.
10. Approvazione di una recensione (da parte del moderatore).
11. Inserimento di un like relativo a una pubblicazione.
12. Calcolo numero dei like per una pubblicazione dato l'ID della pubblicazione.
13. Estrazione elenco delle recensioni approvate per una pubblicazione dato l'ID della pubblicazione.
14. Estrazione elenco delle recensioni in attesa di approvazione.
15. Estrazione log delle modifiche effettuare su una pubblicazione dato l'ID della pubblicazione.
16. Estrazione elenco delle pubblicazioni per le quali è disponibile un download.
17. Estrazione della lista delle pubblicazioni in catalogo, ognuna con la data dell'ultima ristampa.
18. Data una pubblicazione, restituire tutte le pubblicazioni del catalogo aventi gli stessi autori.

Altre operazioni

Necessarie per il corretto funzionamento della base di dati.

- Le operazioni CRUD per le entità pubblicazione e utente
- Promozione e degradazione di un utente
- Operazioni necessarie per l'accesso:
 - Data una email, controllare se è presente nel database.
 - Data una email esistente e una password, controllare che la password sia quella associata all'utente con quella email.

PROGETTAZIONE CONCETTUALE

Per quanto riguarda la strategia di progettazione è stata adottata una strategia ***ibrida*** tra la TOP-DOWN e la BOTTOM-UP in quanto siamo partiti da un scheletro dello schema in cui le macro-componenti del sistema erano definite separatamente per poi sviluppare le stesse e le loro relazioni.

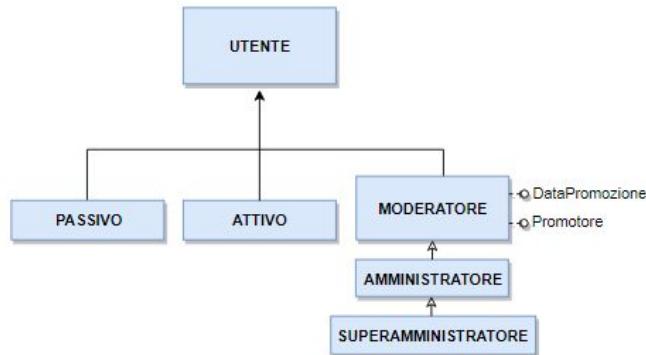
Dettagli relativi alla scelta delle Entità

Utente

Dalla specifica abbiamo subito identificato l'entità Utente necessaria per rappresentare tutti i dati relativi ad essi. In questa prima analisi abbiamo rappresentato **tutti** i dettagli riguardanti gli utilizzatori all'interno dell'entità Utente.



Abbiamo poi rappresentato i vari ruoli mediante una struttura gerarchica :



Notiamo che gli attributi *DataPromozione* e *Promotore* sono propri degli utenti di grado moderatore o superiore. Infatti gli utenti attivi e passivi non derivano da una promozione ma soltanto dalle attività dell'utente. Infatti abbiamo stabilito che un utente passa da passivo ad attivo nel momento in cui scrive una recensione (ed era passivo ovviamente), e passa da attivo a passivo dopo due mesi di inattività sul sistema (ovvero quando l'ultima recensione risale a più di due mesi fa). Gli utenti di grado moderatore o superiore sono definiti attivi a prescindere, essendo i gestori del sistema.

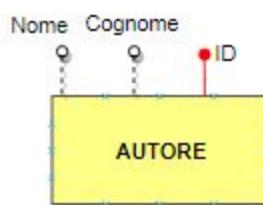
Pubblicazione

L'entità pubblicazione è necessaria per modellare la scheda bibliografica di una pubblicazione. Infatti come possiamo vedere in questo primo modello, contiene tutti i dati relativi ad essa. Notiamo che l'attributo *ListaRistampe* ha molteplicità (0,N), infatti una pubblicazione può essere ristampata da 0 a N volte. Tale attributo dovrà infatti essere ritoccato poiché questa tipologia di attributi non può essere riportata direttamente nel modello logico. Lo stesso discorso va fatto per l'attributo *Parole Chiave* di identica molteplicità. Abbiamo utilizzato un identificatore artificiale per semplicità invece di utilizzare attributi di altro tipo come ad esempio l'ISBN che è unico ed è una sequenza di 13 cifre.



Autore

Per la modellazione del concetto di Autore abbiamo deciso di utilizzare un'entità a parte, invece di includerlo come attributo molteplice nell'entità pubblicazione. Questo perché in futuro il database potrebbe estendere le proprie funzionalità gestendo in maniera più complessa anche la realtà degli autori, ma soprattutto perché possono esistere pubblicazioni che hanno gli stessi autori. Quindi visto che la specifica ci richiede espressamente di implementare l'operazione di ricerca per autore e l'operazione di selezione delle pubblicazioni che hanno gli stessi autori, ci è sembrato opportuno modellare questo concetto con un'entità. Come possiamo vedere l'entità autore è identificata da una chiave artificiale **ID** inserita da noi.



Sorgente

Un altro concetto che abbiamo modellato come entità dall'inizio è quello di sorgente. Questo principalmente perché è un concetto che possiede molti attributi quindi rappresentarlo come attributo multivalore ci è sembrato poco opportuno. Abbiamo deciso di identificare la sorgente con una chiave artificiale poiché l'alternativa era quella di identifierla tramite URI. Questa seconda soluzione ci è sembrata poco ragionevole in quanto una sorgente (ad esempio un'immagine) può appartenere anche a più pubblicazioni, cosa ingestibile o ridondante avendo come chiave primaria l'attributo URI.



Storia

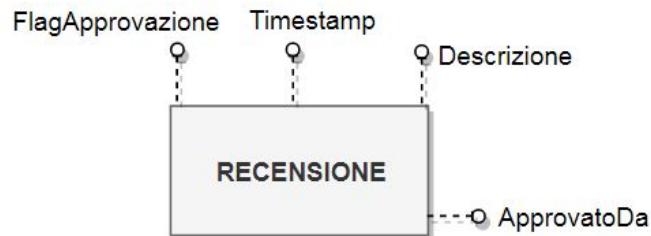
Come espressamente richiesto, lo storico è una funzionalità obbligatoria da realizzare. Abbiamo concepito la storia delle schede bibliografiche come un'entità invece che una relazione perché anche qui concettualmente è complessa e necessita di molti attributi. Come per le entità precedenti abbiamo scelto l'identificatore artificiale **ID** invece che più campi.



Recensione

Abbiamo modellato anche recensione come entità invece di relazione perché è un elemento che nella modellazione da noi concepita necessita di relazionarsi con 3 entità quali: *Utente*, *Moderatore* e *Pubblicazione*, infatti la recensione deve memorizzare l'utente che l'ha scritta, il moderatore che ha approvato e la pubblicazione alla quale si riferisce. Quindi invece di utilizzare una relazione ternaria abbiamo utilizzato un'entità collegata alle altre tre tramite opportune relazioni binarie.

Una *Recensione* è identificata tramite ID dell'utente che scrive e ID della pubblicazione. Infatti un utente non può scrivere più di una recensione per pubblicazione.



Dettagli relativi alla scelta delle Relationship

Scritto (*Pubblicazione - Autore*)

'Scritto' è la relazione che lega l'entità autore all'entità pubblicazione, ed ha i seguenti vincoli: un autore può scrivere da 0 ad N pubblicazioni, invece la pubblicazione deve essere scritta da almeno un autore. Formalmente:



Associata (*Pubblicazione - Recensione*)

'Associata' lega Pubblicazione a Recensione, ed ha i seguenti vincoli: Una pubblicazione può essere associata a 0 o più recensioni, viceversa una recensione è associata ad 1 sola pubblicazione. Formalmente:



Composizione (*Pubblicazione - Sorgente*)

'Composizione' associa l'entità Pubblicazione all'entità Sorgente, ed ha i seguenti vincoli: Una sorgente corrisponde ad una sola pubblicazione, invece una pubblicazione può avere da 0 ad N sorgenti. Formalmente:



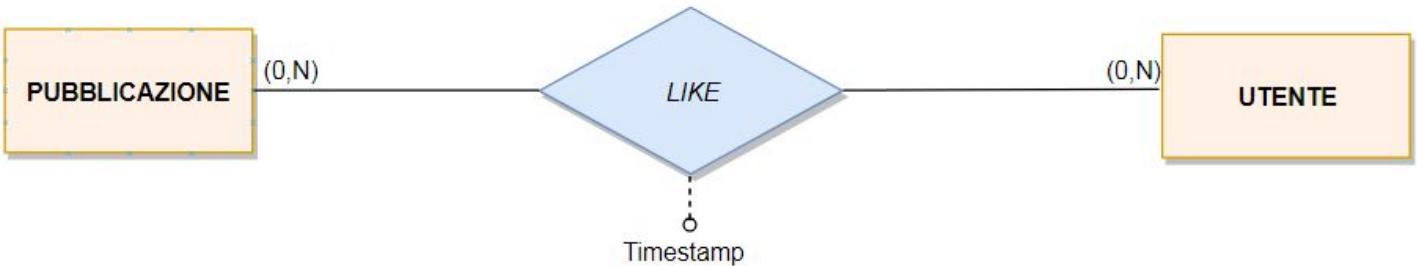
Conservazione (Pubblicazione - Storia)

‘Conservazione’ lega Pubblicazione all’entità Storia, con i seguenti vincoli: Una pubblicazione avrà da 1 a N modifiche ad essa associate (almeno una poiché l’inserimento è considerato come tale), viceversa ogni record della storia farà riferimento ad una sola pubblicazione (a meno che non sia stata eliminata). Formalmente:



Like (Pubblicazione - Utente)

‘Like’ lega l’entità Pubblicazione e l’entità Utente. I vincoli sono: una pubblicazione può avere 0 o N like da N utenti, un utente può mettere like a più libri. Formalmente :



Registrazione (Storia - Utente)

‘Registrazione’ lega l’entità Storia all’entità Utente. I vincoli sono: una modifica di una pubblicazione può essere fatta da un solo utente (quindi (1,1)), viceversa un utente può effettuare molteplici modifiche. Formalmente :



Inserimento (*Recensione - Utente*)

‘Inserimento’ lega l’entità Utente e l’entità Recensione, e rappresenta l’utente che ha inserito la recensione. Vincolata nel seguente modo: una recensione è legata ad 1 solo utente, un utente può scrivere da 0 a N recensioni. Formalmente :



Approvazione (*Recensione - Moderatore*)

‘Approvazione’ lega l’entità Recensione e l’entità Moderatore. I vincoli sono: una recensione è approvata da 1 solo moderatore, un moderatore può approvare da 0 a N recensioni. Formalmente :



Scelte progettuali relative allo schema ER

- I ruoli che abbiamo definito per l'utente sono, in ordine di rilievo, i seguenti:
(Ogni utente di grado superiore eredita i privilegi dei sottostanti)

Passivo : Utenti registrati che non hanno mai inserito una recensione, o non ne hanno inserita una negli ultimi due mesi (l'inserimento di un like non influisce sul ruolo dell'utente).

Attivo : Utenti registrati che hanno inserito una recensione negli ultimi due mesi. Gli utenti di grado moderatore o superiore vengono considerati sempre attivi indipendentemente da tale condizione.

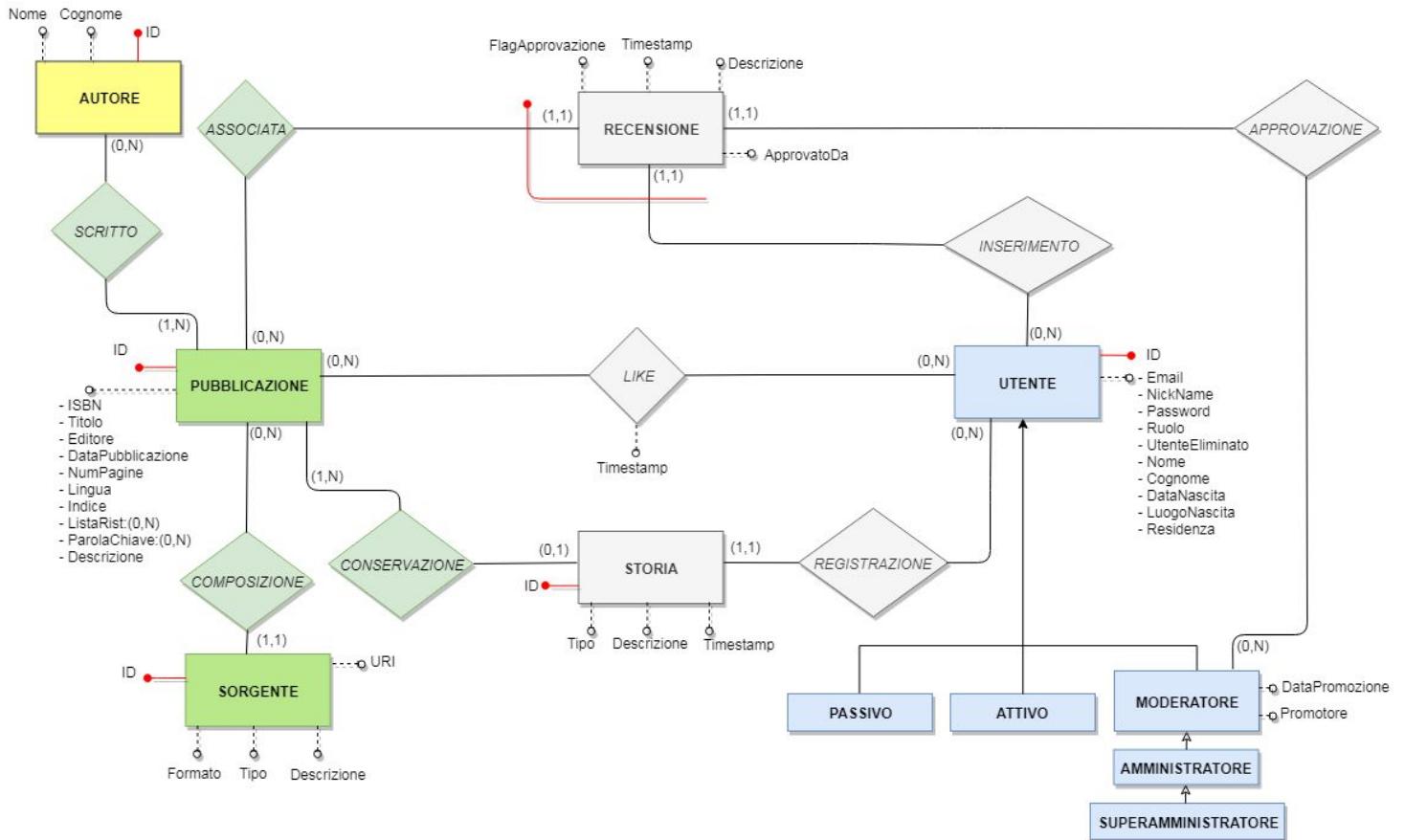
Moderatore : Utenti abilitati all'inserimento e modifica delle pubblicazioni e all'approvazione delle recensioni di altri utenti.

Amministratore : Privilegi di cancellazione di una pubblicazione o di un utente, e di promozione di utenti base a moderatori e degradazione.

Super Amministratore : Unico, gestisce tutte le tipologie di utenti, promuovendo e degradando, gli amministratori.

- Abbiamo deciso di aggiungere all'entità utente il flag **UtenteEliminato**. Quindi, quando l'account di un utente viene cancellato per qualsiasi motivo, l'unica cosa che succede è che **UtenteEliminato** viene posto a 'vero'.
- Abbiamo deciso di aggiungere un **NickName** all'entità utente. Unico per ognuno e visibile nel sistema agli altri utenti.
- Abbiamo inserito all'entità moderatore gli attributi **Promotore** e **DataPromozione**. Il primo identifica l'amministratore che gli ha conferito l'ultima promozione, il secondo rappresenta la data dell'ultima promozione assegnata.
- Abbiamo deciso di aggiungere all'entità recensione gli attributi **FlagApprovazione**, per determinare se la recensione è stata approvata o meno, e **ApprovatoDa** per sapere chi è il moderatore che ha approvato la recensione.
- Abbiamo inserito l'entità **Storia** in relazione sia con l'entità pubblicazione, sia con l'entità utente, in questo modo registrerà tutte le modifiche alle pubblicazioni, e inoltre sarà utile per avere una cronologia delle interazioni pubblicazione-utente. Nello storico non andremo a registrare attività riguardanti i soli utenti (promozione, cancellazione ecc...).
- In questa prima modellazione abbiamo deciso di non creare un'entità metadati e di assegnare i suoi attributi all'entità pubblicazione.

Schema E-R





Descrizione dello Schema

Dizionario delle Entità

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Pubblicazione	-	ID, ISBN, Titolo, Editore, DataPubblicazione, NumPagine, Lingua, Indice, ListaRist:(0,N), ParolaChiave:(1,N), Descrizione	<i>ID (Pubblicazione)</i>
Sorgente	-	ID, URI, Formato, Tipo, Descrizione	<i>ID (Sorgente)</i>
Autore	Autore (o coautore) della pubblicazione	ID, Nome, Cognome	<i>ID (Autore)</i>
Utente	-	Email, NickName, Password, UtenteEliminato, Nome, Cognome, DataNascita, LuogoNascita, Residenza	<i>ID (Utente)</i>
Attivo	utente attivo	-	<i>ID (Utente)</i>
Passivo	utente passivo	-	<i>ID (Utente)</i>
Moderatore	utente moderatore	DataPromozione, Promotore	<i>ID (Utente)</i>
Amministratore	utente di grado amministratore	-	<i>ID (Utente)</i>
SuperAdmin	utente super amministratore	-	<i>ID (Utente)</i>
Recensione	-	Descrizione, FlagApprovazione, Timestamp, ApprovatoDa	<i>ID (Pubblicazione)</i> , <i>ID (Utente)</i>
Storia	-	ID, Descrizione, Tipo, Timestamp	<i>ID (Pubblicazione)</i> , <i>ID (Utente)</i> , <i>Timestamp</i>

Dizionario delle Relationship

RELAZIONE	DESCRIZIONE	COMPONENTI	ATTRIBUTI
Like	I like assegnati alle pubblicazioni dagli utenti del catalogo	<i>Utente, Pubblicazione</i>	<i>Timestamp</i>
Inserimento	Un utente inserisce una propria recensione	<i>Utente, Recensione</i>	-
Associata	La recensione inserita è associata ad un'unica pubblicazione	<i>Recensione, Pubblicazione</i>	-
Approvazione	Un moderatore approva la recensione scritta da un utente	<i>Moderatore, Recensione</i>	-
Conservazione	La storia conserva delle entry con le modifiche fatte ad una pubblicazione	<i>Pubblicazione, Storia</i>	-
Registrazione	In maniera analoga la storia registra anche le modifiche svolte dai vari utenti	<i>Storia, Utente</i>	-
Composizione	Una pubblicazione è associata a più sorgenti, ogni sorgente è associata ad una sola pubblicazione	<i>Pubblicazione, Sorgente</i>	-
Scritto	Una pubblicazione è associata ad uno o più autori, coloro che l'hanno scritta	<i>Pubblicazione, Autore</i>	-

Descrizione degli attributi

Pubblicazione

ID 	Chiave artificiale legata alla pubblicazione.
ISBN	L'ISBN univoco associato alla pubblicazione.
Titolo	Il titolo della pubblicazione, o il nome con cui viene riconosciuta.
Editore	L'editore che ha pubblicato l'opera.
DataPubblicazione	La data in cui l'opera viene pubblicata.
NumPagine	Il numero di pagine della pubblicazione.
Lingua	La lingua in cui la pubblicazione viene scritta.
Indice	L'indice della pubblicazione.
ListaRist (0,N)	Contiene la lista delle ristampe di una pubblicazione, possono essere zero o più.
ParolaChiave (0,N)	Le parole chiave associate alla pubblicazione, possono essere zero o più.
Descrizione	Breve testo per descrivere la pubblicazione.

Utente

ID 	Chiave artificiale associata all'utente.
Email	L'email dell'utente, da inserire durante il login.
Nickname	Il nickname dell'utente, il nome che apparirà sul sito.
Password	La password dell'utente, da inserire durante il login.
UtenteEliminato	Flag usata per determinare se l'utente ha eliminato il suo account.
Nome	Il nome anagrafico dell'utente.
Cognome	Il cognome anagrafico dell'utente.
DataNascita	La data di nascita dell'utente.
LuogoNascita	Il luogo di nascita dell'utente, formato da nazione e città.
Residenza	La residenza dell'utente, formato da nazione e città.

Moderatore

Promotore	Associa l'utente all'amministratore che gli ha conferito l'ultima promozione.
DataPromozione	La data dell'ultima promozione assegnata.

Recensione

Descrizione	Il testo scritto dall'utente che ha inviato la recensione.
FlagApprovazione	Flag usata per determinare se la recensione è stata approvata o è in attesa di approvazione.
Timestamp	Attributo che registra la data e ora di quando la recensione è stata inviata.
ApprovatoDa	Associa alla recensione il moderatore che ha approvato la recensione

20

Like

Timestamp Attributo che registra data e ora di quando l'utente ha messo like alla pubblicazione.

Sorgente

ID 

Chiave artificiale creata per Sorgente.

URI

Il collegamento ipertestuale della sorgente.

Formato

Il formato del file collegato.

Tipo

Il tipo del file (download,immagine, ecc...).

Descrizione

Breve descrizione del contenuto dell'URI.

Storia

ID 

Chiave artificiale creata per Storia.

Descrizione

Breve descrizione della modifica effettuata.

Timestamp

Attributo che registra data e ora di quando la modifica è stata effettuata.

Tipo

Flag per determinare il tipo di modifica effettuata:

1 - Inserimento pubblicazione
2 - Cancellazione pubblicazione
3 - Modifica pubblicazione
4 - Approvazione recensione
5 - Inserimento recensione
6 - Eliminazione recensione
7 - Inserimento like
8 - Eliminazione like

Autore

ID 

Chiave artificiale creata per Autore.

Nome

Il nome anagrafico dell'autore, o il nome d'arte con cui è conosciuto.

Cognome

Il cognome anagrafico dell'autore, il campo è vuoto se l'autore è conosciuto per il suo nome d'arte.



Tavola dei Volumi

CONCETTO	TIPO	VOLUME (~)
Pubblicazione - Ristampe	E	500.000 - 1.500.000 (3 x N.Pub.)
Utente - Attivi - Moderatori - Amministratori	E	1.000.000 - Attivi (50%) - Moderatori (0.1%) - Amministratori (~ 4) Utenti giornalieri (20.000)
Sorgente	E	2.500.000 (N.Pub. * 5)
Autore	E	200.000
Recensione	E	5.000.000
Like	R	50.000.000
Storia - Modifiche	E	500.000.000 - 25.000.000 (5%)

Tavola degli accessi

Op	Descrizione	Freq / gg
1	Modifica del livello di un utente <ul style="list-style-type: none"> - attivo -> passivo - passivo -> attivo - promozione 	- 4 - 5 - 1
2	Estrazione elenco delle ultime dieci pubblicazioni inserite	1.000
3	Estrazione elenco delle pubblicazioni aggiornate di recente (ultimi 30 giorni).	1.000
4	Estrazione elenco dei 30 utenti più “collaborativi”	50
5	Estrazione elenco delle pubblicazioni inserite da un utente.	200
6	Estrazione catalogo, cioè elenco di tutte le pubblicazioni con titolo, autori, editore e anno di pubblicazione, ordinato per titolo.	300
7	Estrazione dati complete di una pubblicazione specifica, dato il suo ID.	2.000
8	Ricerca di pubblicazioni per ISBN, titolo, autore, e parole chiave.	3.000
9	Inserimento di una recensione relativa a una pubblicazione.	100
10	Approvazione di una recensione (da parte del moderatore).	90
11	Inserimento di un like relativo a una pubblicazione.	3.000
12	Calcolo numero dei like per una pubblicazione.	9.000
13	Estrazione elenco delle recensioni approvate per una pubblicazione.	1.500
14	Estrazione elenco delle recensioni in attesa di approvazione.	1.000
15	Estrazione log delle modifiche effettuate su una pubblicazione.	10
16	Estrazione elenco delle pubblicazioni per le quali è disponibile un download.	2.000
17	Estrazione della lista delle pubblicazioni in catalogo, ognuna con la data dell'ultima ristampa	500
18	Data una pubblicazione, restituire tutte le pubblicazioni del catalogo aventi gli stessi autori	500
19	Inserimento di una pubblicazione da parte di un utente moderatore o di grado sup.	2
20	Modifica di una pubblicazione da parte di un utente moderatore o di grado sup. <ul style="list-style-type: none"> - riguardante aggiornamento o inserimento di una ristampa 	20 ~ 20% (4)

PROGETTAZIONE LOGICA

Analisi delle ridondanze

1. **NumLike** (*Numero Like*), per l'entità *Pubblicazione*.
Contatore che viene incrementato per ogni like assegnato alla pubblicazione.
2. **NumRec** (*Numero Recensioni*), per l'entità *Pubblicazione*.
Contatore che viene incrementato per ogni recensione **approvata** della pubblicazione.
3. **NumPubb** (*Numero Pubblicazioni*), per l'entità *DatiModeratore*.
Il numero di pubblicazioni inserite nel catalogo dall'utente.
4. **DataUltimaRecensione**, per l'entità *Utente*.
La data dell'ultima recensione inserita dall'utente.
5. **DataUltimaModifica**, per l'entità *Pubblicazione*.
Data dell'ultima modifica effettuata sulla pubblicazione.
6. **DataUltimaRistampa**, per l'entità *Pubblicazione*.
Data dell'ultima ristampa, in ordine cronologico, associata alla pubblicazione.

N.B.

Per le frequenze delle operazioni sono state utilizzate quelle giornaliere, mentre per il numero di accessi alle varie tabelle abbiamo considerato il numero di accessi medi in base alla tavola dei volumi:

Es. se ci sono 100 libri e 10.000 like nel database il numero di accessi per contare il numero di like ad un libro è $10.000/100 = 100$

Analisi attributo NumLike

Tabella degli accessi in assenza di ridondanze

Operazione 11: Inserimento Like

Concetto	Costrutto	Accessi	Tipo
Like	R	1	W
Storia	E	1	W

$$(2 \times 2) \times 3.000 = 12.000 A$$

$$\text{Totale: } 12.000 + 900.000 = 912.000 A$$

Tabella degli accessi in presenza di ridondanze

Operazione 11: Inserimento Like

Concetto	Costrutto	Accessi	Tipo
Like	R	1	W
Pubblicazione	E	1	W
Storia	E	1	W

$$(3 \times 2) \times 3.000 = 18.000 A$$

$$\text{Totale: } 18.000 + 9.000 = 27.000 A$$

Abbiamo scelto di aggiungere la ridondanza *NumLike*

Operazione 12: Lettura Like / Pubb

Concetto	Costrutto	Accessi	Tipo
Like	R	100	R

$$(1 \times 100) \times 9.000 = 900.000 A$$

Operazione 12: Lettura Like / Pubb

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	1	R

$$1 \times 9.000 = 9.000 A$$

Analisi attributo NumRec

Tabella degli accessi in assenza di ridondanze

Operazione 9: Inserimento Recensione

Concetto	Costrutto	Accessi	Tipo
Recensione	E	1	W
Storia	E	1	W

$$(2 \times 2) \times 100 = 400 \text{ Accessi}$$

$$\text{Totale : } 400 + 37.500 = \textcolor{red}{37.900 A}$$

Tabella degli accessi in presenza di ridondanze

Operazione 9: Inserimento Recensione

Concetto	Costrutto	Accessi	Tipo
Recensione	E	1	W
Pubblicazione	E	1	W
Storia	E	1	W

$$(3 \times 2) \times 100 = 600 \text{ Accessi}$$

$$\text{Totale : } 600 + 1.500 = \textcolor{green}{2.100 A}$$

Abbiamo scelto di aggiungere la ridondanza *NumRec*

Operazione 13: Lettura Recensione / Pubb

Concetto	Costrutto	Accessi	Tipo
Recensione	R	25	R

$$(1 \times 25) \times 1.500 = 37.500 \text{ Accessi}$$

Operazione 13: Lettura Recensione / Pubb

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	1	R

$$1 \times 1.500 = 1.500 \text{ Accessi}$$

Analisi attributo NumPubb

Tabella degli accessi in assenza di ridondanze

Operazione 19: Inserimento Pubblicazione

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	1	W
Metadati	E	1	W
Storia	E	1	W

$$(3 \times 2) \times 2 = 12 \text{ Accessi}$$

Totale : 12 + 25.001.500 = 25.001.515 A

Operazione 4: utenti piu' collaborativi

Concetto	Costrutto	Accessi	Tipo
Storia	E	500.000	R
Utente	E	30	R

$$(500.000 + 30) \times 50 = 25.001.500$$

Tabella degli accessi in presenza di ridondanze

Operazione 19: Inserimento Pubblicazione

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	1	W
Metadati	E	1	W
Storia	E	1	W
DatiModeratore	E	1	W

$$(4 \times 2) \times 2 = 16 \text{ Accessi}$$

Totale : 16 + 501.500 = 501.516 A

Abbiamo scelto di aggiungere la ridondanza *NumPubb*

Operazione 4: Lettura DatiModeratore

Concetto	Costrutto	Accessi	Tipo
DatiModeratore	E	10.000	R
Utente	E	30	R

$$(10.000 + 30) \times 50 = 501.500 \text{ Accessi}$$

Analisi attributo DataUltimaRecensione

Tabella degli accessi in assenza di ridondanze

Operazione 9: Inserimento Recensione

Concetto	Costrutto	Accessi	Tipo
Recensione	E	1	W
Storia	E	1	W

$$(2 \times 2) \times 100 = 400 \text{ Accessi}$$

$$\text{Totale : } 400 + 5.500.008 = \textcolor{red}{5.500.408 A}$$

Operazione 1: Utenti attivo -> passivo (evento)

Concetto	Costrutto	Accessi	Tipo
Storia	E	5.000.000	R
Utente	E	500.000	R
Utente	E	4	W

$$(5.500.000 + 4 \times 2) \times 1 = 5.500.008 \text{ Accessi}$$

Tabella degli accessi in presenza di ridondanze

Operazione 9: Inserimento Recensione

Concetto	Costrutto	Accessi	Tipo
Recensione	E	1	W
Storia	E	1	W
Utente	E	1	W

$$(3 \times 2) \times 100 = 600 \text{ Accessi}$$

$$\text{Totale : } 600 + 500.008 = \textcolor{green}{500.608 A}$$

Operazione 1: Utenti attivo -> passivo (evento)

Concetto	Costrutto	Accessi	Tipo
Utente	E	500.000	R
Utente	E	4	W

$$(500.000 + 4 \times 2) \times 1 = 500.008 \text{ Accessi}$$

Abbiamo scelto di aggiungere la ridondanza *DataUltimaRecensione*

Analisi attributo DataUltimaModifica

Tabella degli accessi in assenza di ridondanze

Operazione 20: Modifica Pubblicazione

Concetto	Costrutto	Accessi	Tipo
Storia	E	1	W

$$(1 \times 2) \times 20 = 40 A$$

Totale: $40 + 2.550.000.000 = 20.550.000.040 A$

Tabella degli accessi in presenza di ridondanze

Operazione 20: Modifica Pubblicazione

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	1	W
Storia	E	1	W

$$(2 \times 2) \times 20 = 80 A$$

Totale: $80 + 500.000.000 = 500.000.080 A$

Abbiamo scelto di aggiungere la ridondanza *DataUltimaModifica*

Operazione 3: Estrazione pubb. aggiornate

Concetto	Costrutto	Accessi	Tipo
Storia	E	25.000.000	R
Pubb.	E	500.000	R

$$(1 \times 25.500.000) \times 1000 = 20.550.000.000 A$$

Operazione 3: Estrazione pubb. aggiornate

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	500.000	R

$$(500.000) \times 1000 = 500.000.000 A$$

Analisi attributo DataUltimaRistampa

Tabella degli accessi in assenza di ridondanze

Operazione 20: Modifica Pubblicazione (ristampa)

Concetto	Costrutto	Accessi	Tipo
Storia	E	1	W
Ristampa	E	1	W

$$(2 \times 2) \times 4 = 16 A$$

Totale: **16 + 1.000.000.000 = 1.000.000.016 A**

Operazione 17: Estrazione pubb. con dataUlRis

Concetto	Costrutto	Accessi	Tipo
Ristampa	E	1.500.000	R
Pubblicaz.	E	500.000	R

$$(2.000.000) \times 500 \\ = 1.000.000.000 A$$

Tabella degli accessi in presenza di ridondanze

Operazione 20: Modifica Pubblicazione (ristampa)

Concetto	Costrutto	Accessi	Tipo
Ristampa	E	1	W
Storia	E	1	W
Pubblicazione	E	1	W

$$(3 \times 2) \times 4 = 24 A$$

Totale: **24 + 250.000.000 = 250.000.024 A**

Operazione 17: Estrazione pubb. con dataUlRis

Concetto	Costrutto	Accessi	Tipo
Pubblicazione	E	500.000	R

$$(1 \times 500.000) \times 500 = 250.000.000 A$$

Campo di tipo Data : 3 bytes x 500.000 circa 2 Gb aggiuntivi per $\sqrt{2}$ degli accessi

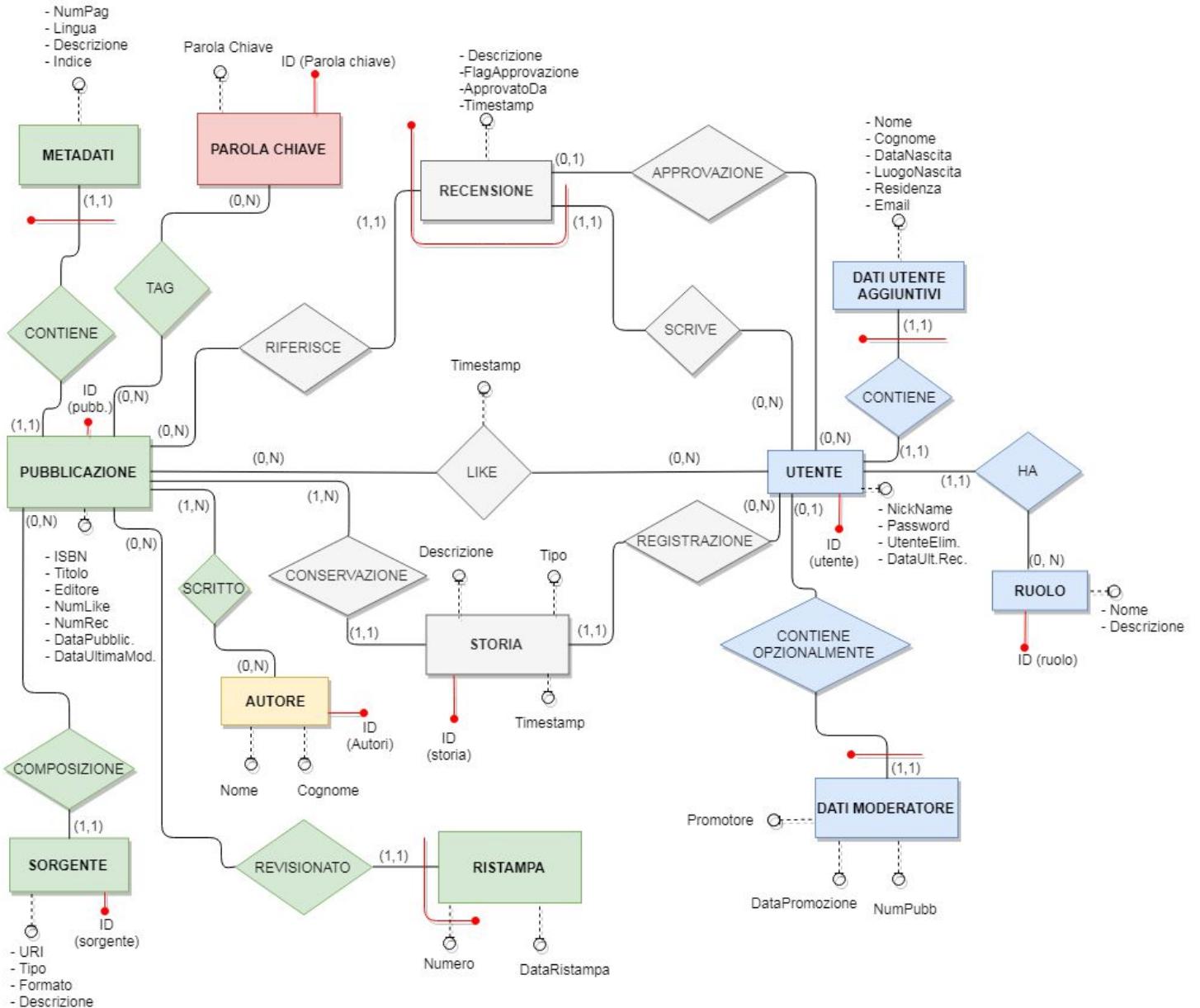
Abbiamo scelto di NON aggiungere la ridondanza DataUltimaRistampa

Scelte relative alla ristrutturazione dello schema ER

- ❑ Abbiamo deciso di inserire le ridondanze in base all'analisi precedente.
- ❑ Abbiamo deciso di eliminare la **generalizzazione sull'entità utente** tramite un collasso verso l'alto, lasciando solo l'entità *Utente* e identificandone il tipo tramite l'attributo Ruolo. Esso infatti farà riferimento alla tabella **Ruolo** che abbiamo deciso di inserire per un maggiore grado di trasparenza della base di dati.
- ❑ I dati dell'entità utente così ottenuta sono poi stati divisi su 3 entità diverse, in modo da alleggerire il carico operazionale sulla tabella utente riducendo il numero di attributi, inoltre: L'entità **Dati Moderatore** contiene informazioni riguardo i soli utenti di grado moderatore o superiore, questo ci aiuterà a non avere un numero elevato di campi lasciati nulli sugli utenti base al momento dell'implementazione.
L'entità **Dati Utente Aggiuntivi** conserva tutte le informazioni più sensibili dell'utente, in questo modo al momento della cancellazione dell'utente basterà cancellare il contenuto di questo schema senza modificare l'entità utente e compromettere le varie relazioni con le altre entità.
In questo modo sarà anche possibile permettere il recupero di un utente eliminato, tramite l'inserimento del nickname e della password di quell'utente, recuperando così tutte le attività passate.
- ❑ Abbiamo deciso di suddividere l'entità Pubblicazione in varie tabelle, per ridurre il numero di attributi dell'entità principale e rendere più efficienti le query sulle pubblicazioni:
L'entità **Metadati** contiene le informazioni delle pubblicazioni che non sono utili per le query di ricerca, visiteremo gli attributi di questa entità solo quando ci interesserà avere l'elenco completo di tutte le informazioni riguardanti una pubblicazione.
L'entità **Parola chiave** contiene tutte le parole chiave conservative nel database legate alle pubblicazioni tramite la relazione tag questo renderà più veloci le query basate sulla ricerca per parole chiave.
L'entità **Autore** conserva tutti gli autori del sistema, legate alle pubblicazione tramite la relazione scritto.
L'entità **Ristampa** conserva le informazioni sulle varie ristampe di tutte le pubblicazioni, questo renderà più semplice l'inserimento di nuove ristampe per una pubblicazione, e semplificherà l'operazione #17 che richiede di associare ad ogni pubblicazione la data di ultima ristampa.



Schema E-R Ristrutturato



Dizionario delle Entità (dopo la ristrutturazione)

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Pubblicazione	-	ID, ISBN, Titolo, Editore, NumLike, NumRec, DataPubblicazione, DataUltimaModifica	ID (Pubblicazione)
Metadati	Parte dei dati della pubblicazione	NumPag, Lingua, Descrizione, Indice	ID (Pubblicazione)
Parola Chiave	una Parola chiave che può essere collegata ad una pubblicazione	Parola chiave	ID (Parola Chiave)
Ristampa	La ristampa di una pubblicazione	ID, DataRistampa, Numero	ID (Pubblicazione), numero
Sorgente	-	ID, URI, Formato, Tipo, Descrizione	ID (Sorgente)
Autore	Autore della pubblicazione	ID, Nome, Cognome	ID (Autore)
Utente	-	ID, NickName, Password, Ruolo, UtenteEliminato, DataUltimaRecensione	ID (Utente)
Dati utente aggiuntivi	Parte dei dati di un utente	Email, Nome, Cognome, DataNascita, LuogoNascita, Residenza	ID (Utente)
Dati moderatore	Dati aggiuntivi di utenti di grado di moderatore o superiore	Promotore, DataPromozione, NumPubb	ID (Utente)
Recensione	-	Descrizione, FlagApprovazione, ApprovatoDa, Timestamp	ID (Pubblicazione), ID (Utente)
Storia	-	ID, Descrizione, Tipo, Timestamp	ID (Pubblicazione), ID (Utente), Timestamp
Ruolo	Tipologia di utenza	ID, Nome, Descrizione	ID (Ruolo)

Dizionario delle Relationships (dopo la ristrutturazione)

RELAZIONE	DESCRIZIONE	COMPONENTI	ATTRIBUTI
Like	I like assegnati alle pubblicazioni dagli utenti del catalogo	<i>Utente, Pubblicazione</i>	<i>Timestamp</i>
Inserimento	Un utente inserisce una propria recensione	<i>Utente, Recensione</i>	-
Associata	La recensione inserita è associata ad un'unica pubblicazione	<i>Recensione, Pubblicazione</i>	-
Approvazione	Un moderatore approva la recensione scritta da un utente	<i>Moderatore, Recensione</i>	-
Conservazione	La storia conserva delle entry con le modifiche fatte ad una pubblicazione	<i>Pubblicazione, Storia</i>	-
Registrazione	In maniera analoga la storia registra anche le modifiche svolte dai vari utenti	<i>Storia, Utente</i>	-
Composizione	Una pubblicazione è associata a più sorgenti, ogni sorgente è associata ad una sola pubblicazione	<i>Pubblicazione, Sorgente</i>	-
Scritto	Una pubblicazione è associata ad uno o più autori, coloro che l'hanno scritta	<i>Pubblicazione, Autore</i>	-
Contiene	connette l'insieme completo dei dati riguardanti un utente	<i>Utente, Dati utente aggiuntivi</i>	-
Contiene opzionalmente	associa ad un utente di grado moderatore o superiore le sue informazioni "speciali"	<i>Utente, Dati moderatore</i>	-
Tag	associa la pubblicazione alle parole chiave che gli vengono assegnate, ogni pubblicazione può avere assegnata una o più parole chiave	<i>Pubblicazione, Parola chiave</i>	-
Ha	associa un utente al suo ruolo, che non può essere nullo	<i>Utente, Ruolo</i>	-
Revisionato	una pubblicazione è associata alle sue varie ristampe, se ce ne sono	<i>Pubblicazione, Ristampa</i>	-

Descrizione degli Attributi

Pubblicazione

ID 	Chiave artificiale legata alla pubblicazione.
ISBN	L'ISBN univoco associato alla pubblicazione.
Titolo	Il titolo della pubblicazione, o il nome con cui viene riconosciuta.
Editore	L'editore che ha pubblicato l'opera.
NumLike	Contatore che viene incrementato per ogni like assegnato alla pubblicazione.
NumRec	Contatore che viene incrementato per ogni recensione approvata della pubblicazione.
DataPubblicazione	La data in cui l'opera viene pubblicata.
DataUltimaModifica	Data dell'ultima modifica effettuata sulla pubblicazione.

Metadati

NumPag	Il numero di pagine della pubblicazione.
Lingua	La lingua in cui la pubblicazione viene scritta.
Indice	L'indice della pubblicazione.
Descrizione	Breve testo per descrivere la pubblicazione.

Parola chiave

ID 	Chiave artificiale di identificazione della Keyword.
Parola chiave	la parola chiave.

Ristampa

Numero	Per determinare l'ordine di uscita delle ristampe (utile per esempio se si vuole inserire la data della quinta ristampa di un libro senza volere prima inserire le date delle prime quattro).
DataRistampa	La data in cui è stata pubblicata la ristampa.



Utente

ID	Chiave artificiale associata all'utente.
NickName	Il nickname dell'utente, il nome che apparirà sul sito.
Password	La password dell'utente, da inserire durante il login.
UtenteEliminato	Flag usata per determinare se l'utente ha eliminato il suo account.
DataUltimaRecensione	La data dell'ultima recensione inserita dall'utente.
Ruolo	Attributo usato per determinare la tipologia dell'utente (ogni valore è associato ad un ruolo) : 1 - Passivo 2 - Attivo 3 - Moderatore 4 - Amministratore 5 - SuperAmministratore

Dati Utente Aggiuntivi

Email	L'email dell'utente, da inserire durante il login.
Nome	Il nome anagrafico dell'utente.
Cognome	Il cognome anagrafico dell'utente.
DataNascita	La data di nascita dell'utente.
LuogoNascita	Il luogo di nascita dell'utente, formato da nazione e città.
Residenza	La residenza dell'utente, formato da nazione e città.

Dati Moderatore

Promotore	Associa l'utente all'amministratore che gli ha conferito l'ultima promozione.
DataPromozione	La data dell'ultima promozione assegnata.
NumPubb	Il numero di pubblicazioni inserite nel catalogo dall'utente.

Recensione

Descrizione	Il testo scritto dall'utente che ha inviato la recensione.
FlagApprovazione	Flag usata per determinare se la recensione è stata approvata o è in attesa di approvazione.
Timestamp	Attributo che registra la data e ora di quando la recensione è stata inviata.
ApprovatoDa	Associa alla recensione il moderatore che ha approvato la recensione

36

Like

Timestamp

Attributo che registra data e ora di quando l'utente ha messo like alla pubblicazione.

Sorgente

ID

Chiave artificiale creata per Sorgente.

URI

Il collegamento ipertestuale della sorgente.

Formato

Il formato del file collegato.

Tipo

Il tipo del file (download,immagine, ecc...).

Descrizione

Breve descrizione del contenuto dell'URI.

Storia

ID

Chiave artificiale creata per Storia.

Descrizione

Breve descrizione della modifica effettuata.

Timestamp

Attributo che registra data e ora di quando la modifica è stata effettuata.

Tipo

Flag per determinare il tipo di modifica effettuata:

1 - Inserimento pubblicazione
2 - Cancellazione pubblicazione
3 - Modifica pubblicazione
4 - Approvazione recensione
5 - Inserimento recensione
6 - Eliminazione recensione
7 - Inserimento like
8 - Eliminazione like

Autore

ID

Chiave artificiale creata per Autore.

Nome

Il nome anagrafico dell'autore, o il nome d'arte con cui è conosciuto.

Cognome

Il cognome anagrafico dell'autore, il campo è vuoto se l'autore è conosciuto per il suo nome d'arte.

Ruolo

ID

Chiave artificiale creata per Ruolo.

Nome

Il nome del ruolo.

Descrizione

Descrizione del ruolo con le capabilities che possiede.

Schema Logico

Pubblicazione	( <u>ID</u> , Titolo, Editore, NumLike, NumRec, DataPubblicazione, DataUltimaModifica, ISBN)
Metadati	( <u>IDPubblicazione</u> , NPag, Lingua, Descrizione, Indice)
Sorgente	( <u>ID</u> , <u>IDPubblicazione</u> , URI, Tipo, Formato, Descrizione)
Scritto	( <u>IDPubblicazione</u> ,  <u>IDAutore</u>)
Autore	( <u>ID</u> , Cognome, Nome)
Tag	( <u>IDPubblicazione</u> ,  <u>IDParolaChiave</u>)
ParolaChiave	( <u>ID</u> , ParolaChiave)
Ristampa	( <u>ID</u> , <u>IDPubblicazione</u> , DataRistampa, numero)
Utente	( <u>ID</u> , NickName, Password, Ruolo, UtenteEliminato, DataUltimaRecensione)
DatiUtente	( <u>IDUtente</u> , Nome, Cognome, DataNascita, LuogoNascita, Residenza, Email)
DatiModeratori	( <u>IDUtente</u> , NumPubb, DataPromozione, Promotore)
Recensione	( <u>IDPubblicazione</u> ,  <u>IDUtente</u> , Descrizione, FlagApprovazione, ApprovatoDa, Timestamp)
Storia	( <u>ID</u> , <u>IDPubblicazione</u> , <u>IDUtente</u> , Timestamp, Descrizione, Tipo)
Like	( <u>IDUtente</u> ,  <u>IDPubblicazione</u> , Timestamp)
Ruolo	( <u>ID</u> , Nome, Descrizione)

* FOREIGN_KEYS



Vincoli

Vincoli di integrità referenziale

TABELLA	ATTRIBUTO REFERENTE	TAB. E ATTRIBUTI RIFERITI
DatiUtente	<i>IDUtente</i>	Utente : ID
DatiModeratore	<i>IDUtente</i>	Utente : ID
DatiModeratore	<i>Promotore</i>	Utente : ID
Utente	<i>Ruolo</i>	Ruolo : ID
Recensione	<i>IDPubblicazione</i>	Pubblicazione : ID
Recensione	<i>IDUtente</i>	Utente : ID
Recensione	<i>ApprovatoDa</i>	Utente : ID
Like	<i>IDPubblicazione</i>	Pubblicazione : ID
Like	<i>IDUtente</i>	Utente : ID
Storia	<i>IDPubblicazione</i>	Pubblicazione : ID
Storia	<i>IDUtente</i>	Utente : ID
Metadati	<i>IDPubblicazione</i>	Pubblicazione : ID
Ristampa	<i>IDPubblicazione</i>	Pubblicazione : ID
Sorgente	<i>IDPubblicazione</i>	Pubblicazione : ID
Scritto	<i>IDPubblicazione</i>	Pubblicazione : ID
Scritto	<i>IDAutore</i>	Autore : ID
Tag	<i>IDPubblicazione</i>	Pubblicazione : ID
Tag	<i>IDParolaChiave</i>	ParolaChiave : ID

Vincoli non esprimibili

- L'utente passa da attivo a passivo se non scrive recensioni per 2 mesi.
- L'utente passa da passivo a attivo quando scrive una recensione.

Pubblicazione:

NumLike: deve sempre corrispondere al numero di like associati alla pubblicazione.

NumRec: deve sempre corrispondere al numero di recensioni **approvate** della pubblicazione.

DataUltimaModifica: deve corrispondere alla data dell'ultima entry di modifica in storia di quella pubblicazione

Ristampa:

DataRistampa e numero : le ristampe di una pubblicazione in ordine per ‘Numero’ dovranno anche essere in ordine cronologico.

DatiUtente:

LuogoNascita e Residenza : entrambi i capi sono una stringa formata da Nazione (di nascita/residenza) e città. *Es. Italia Roma*

Recensione:

Una recensione non approvata non è visibile e non aumenta il numero di recensioni della pubblicazione, sono utenti moderatori o di grado superiore possono visualizzarla.

Una recensione non può essere approvata dalla stessa persona che l'ha scritta.

DatiModeratore:

NumPubb: deve sempre corrispondere al numero di pubblicazioni inserite dall'utente.

Utente:

DataUltimaRecensione: deve corrispondere alla data in cui l'utente ha scritto la sua ultima recensione

Vincoli di dominio

Valore NULL ammesso

Utente	(DataUltimaRecensione)
Autore	(Cognome)
Sorgente	(Descrizione)
Storia	(IDPubblicazione : quando la pubblicazione è stata eliminata)
Recensione	(ApprovatoDa)
DatiModeratore	(Promotore)

Variabili numeriche unsigned

Pubblicazione	(ID, NumLike, NumRec)
Metadati	(IDPubblicazione, NPag)
Sorgente	(ID, IDPubblicazione)
Scritto	(IDPubblicazione, IDAutore)

Autore	(ID)
Tag	(IDPubblicazione, IDParolaChiave)
ParolaChiave	(ID)
Ristampa	(ID, IDPubblicazione, numero)
Utente	(ID, Ruolo, UtenteEliminato)
DatiUtente	(IDUtente)
DatiModeratori	(IDUtente, NumPubb, Promotore)
Recensione	(IDPubblicazione, IDUtente, FlagApprovazione, ApprovatoDa)
Storia	(ID, IDPubblicazione, IDUtente, Tipo)
Like	(IDUtente, IDPubblicazione)

Variabili uniche (oltre le chiavi)

Pubblicazione	(ISBN)
ParolaChiave	(ParolaChiave)
Utente	(NickName)
DatiUtente	(Email)

Vincoli specifici

DatiUtente	(Email) : deve essere del formato <i>string@string.string</i>
Sorgente	(URI) : deve essere del formato <i>string://string</i>
Utente	(Password) : la password che viene memorizzata nel database sarà la stringa ottenuta dall'hash dell'algoritmo md5 (implementato a livello applicativo), quindi la stringa inserita dovrà essere di 32 cifre esadecimali che vanno da 0 a f.
Metadati	(ISBN) : un codice di 13 cifre numeriche.
Storia	(Tipo) : i valori ammessi sono da 1 a 8.

L'attributo URI dell'entità sorgente non è stato definito unico, in quanto lo sviluppo di indici su campi di elevata lunghezza (gli URI sono al massimo di circa 2000 caratteri) sono computazionalmente onerosi. In questo modo sarà anche possibile associare due o più pubblicazioni allo stesso URI, magari in caso di risorse condivise.

PROGETTAZIONE FISICA

Tecnologie Utilizzate

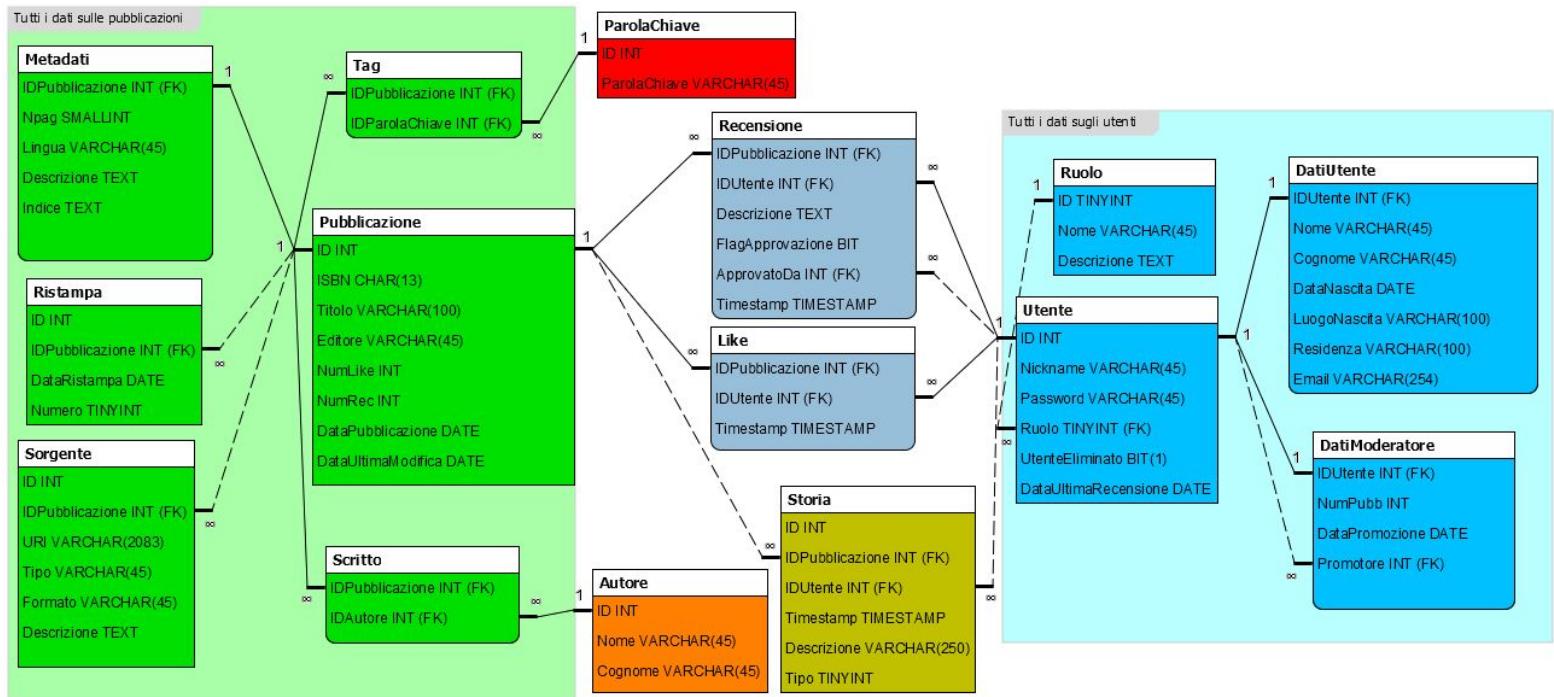


Il database è stato generato e utilizzato su MySQL Server



Per la progettazione logica e fisica del database, stesura di query, procedure e codice SQL abbiamo utilizzato MySQL Workbench

Schema Fisico



Scelte relative all'implementazione

- ❑ Abbiamo deciso di utilizzare delle variabili di tipo TinyInt per l'ID della tabella *Ruolo* (e di conseguenza per l'attributo ruolo della tabella *Utente*) e Tipo della tabella *Storia*, in quanto valori vanno da 1 a 5 per Ruolo e da 1 a 8 per Storia.
- ❑ Per le altre variabili abbiamo scelto il tipo più adatto in base al contenuto.
- ❑ Abbiamo creato degli indici su determinati campi delle varie tabelle in base alle operazioni richieste per aumentare le prestazioni del sistema, abbiamo considerato l'aggiunta di indici quasi sempre come una scelta adeguata in quanto, riguardo le pubblicazioni e gli utenti, il numero di query di ricerca è molto più elevato rispetto al numero di operazioni di aggiornamento, quindi, sebbene la creazione di indici porta ad una riduzione delle prestazioni durante operazioni di modifica, cancellazione e inserimento, il vantaggio di avere query di ricerca più efficienti da parte degli utenti del sito ha un valore maggiore.
Oltre agli indici creati sulle chiavi, i campi '**unique**' e le '**foreign key**' sono stati creati anche indici :

Utente

Sull'attributo 'Ruolo'.

Pubblicazione

Sull'attributo 'Titolo'.

Sorgente

Sull'attributo 'Tipo'.

Autore

Uno sull'attributo 'Nome' e una sull'attributo 'Cognome'.

Ristampa

Sull'attributo 'DataRistampa'.

Storia

Sull'attributo 'Tipo'.

- ❑ Per il passaggio di utenti da attivo a passivo è stato creato un evento che si attiva una volta a settimana di notte (quando si suppone scarico di richieste il database) che pone a passivi tutti gli utenti attivi che negli ultimi due mesi non hanno scritto recensioni
- ❑ Quando una pubblicazione viene eliminata tutti i like, le recensioni e le altre informazioni a lei legate vengono eliminate, l'unica cosa che rimane sono i record in storia con tutti i suoi aggiornamenti, più il record di eliminazione, quando un utente viene eliminato vengono cancellati solo i suoi dati sensibili nella tabella DatiUtente

- ❑ Per quanto riguarda i vincoli di dominio, abbiamo creato alcuni trigger per i check secondo noi più significativi, il resto verrà gestito a livello applicativo.



Creazione del Database

(Generato con MySQL Workbench)

```
-- Schema ProgettoDB
-----
CREATE SCHEMA IF NOT EXISTS `ProgettoDB` DEFAULT CHARACTER SET utf8 ;
USE `ProgettoDB` ;
```

```
-- Table `ProgettoDB`.`Pubblicazione`
-----
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Pubblicazione` (
  `ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `ISBN` CHAR(13) NOT NULL,
  `Titolo` VARCHAR(100) NOT NULL DEFAULT 'Sconosciuto',
  `Editore` VARCHAR(45) NOT NULL DEFAULT 'Sconosciuto',
  `NumLike` INT NOT NULL DEFAULT 0,
  `NumRec` INT NOT NULL DEFAULT 0,
  `DataPubblicazione` DATE NOT NULL,
  `DataUltimaModifica` DATE NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `bytitle` (`Titolo` ASC) INVISIBLE,
  UNIQUE INDEX `ISBN_UNIQUE` (`ISBN` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`Ruolo`
-----
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Ruolo` (
  `ID` TINYINT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Descrizione` TEXT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`Utente`
-----
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Utente` (
  `ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Nickname` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `Ruolo` TINYINT UNSIGNED NOT NULL DEFAULT 1,
  `UtenteEliminato` BIT(1) NOT NULL DEFAULT 0,
  `DataUltimaRecensione` DATE NULL,
```



44

```
PRIMARY KEY (`ID`),  
UNIQUE INDEX `Email_UNIQUE` (`Nickname` ASC) VISIBLE,  
INDEX `indextipo` (`Ruolo` ASC) INVISIBLE,  
CONSTRAINT `fk_Ruolo`  
    FOREIGN KEY (`Ruolo`)  
        REFERENCES `ProgettoDB`.`Ruolo` (`ID`)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`Sorgente`
```

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Sorgente` (  
    `ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
    `IDPubblicazione` INT UNSIGNED NOT NULL,  
    `URI` VARCHAR(2083) NOT NULL,  
    `Tipo` VARCHAR(45) NOT NULL DEFAULT 'Sconosciuto',  
    `Formato` VARCHAR(45) NOT NULL DEFAULT 'Sconosciuto',  
    `Descrizione` TEXT NULL,  
    PRIMARY KEY (`ID`),  
    INDEX `fk_Sorgente_Pubblicazione1_idx` (`IDPubblicazione` ASC) VISIBLE,  
    INDEX `indiceSuTipo` (`Tipo` ASC) VISIBLE,  
    CONSTRAINT `fk_Sorgente_Pubblicazione1`  
        FOREIGN KEY (`IDPubblicazione`)  
            REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)  
            ON DELETE CASCADE  
            ON UPDATE CASCADE)  
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`Metadati`
```

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Metadati` (  
    `IDPubblicazione` INT UNSIGNED NOT NULL,  
    `Npag` SMALLINT UNSIGNED NOT NULL,  
    `Lingua` VARCHAR(45) NOT NULL,  
    `Descrizione` TEXT NOT NULL,  
    `Indice` TEXT NOT NULL,  
    PRIMARY KEY (`IDPubblicazione`),  
    CONSTRAINT `fk_Metadati_Pubblicazione1`  
        FOREIGN KEY (`IDPubblicazione`)  
            REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)  
            ON DELETE CASCADE  
            ON UPDATE CASCADE)  
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`Recensione`
```

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Recensione` (  
    `IDPubblicazione` INT UNSIGNED NOT NULL,  
    `IDUtente` INT UNSIGNED NOT NULL,  
    `Descrizione` TEXT NOT NULL,  
    `FlagApprovazione` BIT NOT NULL DEFAULT 0,  
    `ApprovatoDa` INT UNSIGNED NULL,  
    `Timestamp` TIMESTAMP NOT NULL,  
    PRIMARY KEY (`IDPubblicazione`, `IDUtente`),  
    INDEX `fk_Pubblicazione_has_Utente_Utente1_idx` (`IDUtente` ASC) VISIBLE,
```



```
INDEX `fk_Pubblicazione_has_Utente_Pubblicazione_idx` (`IDPubblicazione` ASC) VISIBLE,
INDEX `fk_approva_idx` (`ApprovatoDa` ASC) VISIBLE,
CONSTRAINT `fk_Pubblicazione_has_Utente_Pubblicazione`
    FOREIGN KEY (`IDPubblicazione`)
    REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
CONSTRAINT `fk_Pubblicazione_has_Utente1`
    FOREIGN KEY (`IDUtente`)
    REFERENCES `ProgettoDB`.`Utente` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
CONSTRAINT `fk_approva`
    FOREIGN KEY (`ApprovatoDa`)
    REFERENCES `ProgettoDB`.`Utente` (`ID`)
    ON DELETE SET NULL
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`Like`
```

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Like` (
`IDPubblicazione` INT UNSIGNED NOT NULL,
`IDUtente` INT UNSIGNED NOT NULL,
`Timestamp` TIMESTAMP NOT NULL,
PRIMARY KEY (`IDPubblicazione`, `IDUtente`),
INDEX `fk_Pubblicazione_has_Utente1_Utente1_idx` (`IDUtente` ASC) VISIBLE,
INDEX `fk_Pubblicazione_has_Utente1_Pubblicazione1_idx` (`IDPubblicazione` ASC) VISIBLE,
CONSTRAINT `fk_Pubblicazione_has_Utente1_Pubblicazione1`
    FOREIGN KEY (`IDPubblicazione`)
    REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
CONSTRAINT `fk_Pubblicazione_has_Utente1_Utente1`
    FOREIGN KEY (`IDUtente`)
    REFERENCES `ProgettoDB`.`Utente` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- Table `ProgettoDB`.`DatiUtente`
```

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`DatiUtente` (
`IDUtente` INT UNSIGNED NOT NULL,
`Nome` VARCHAR(45) NOT NULL,
`Cognome` VARCHAR(45) NOT NULL,
`DataNascita` DATE NOT NULL,
`LuogoNascita` VARCHAR(100) NOT NULL,
`Residenza` VARCHAR(100) NOT NULL,
`Email` VARCHAR(254) NOT NULL,
PRIMARY KEY (`IDUtente`),
UNIQUE INDEX `DatiUtentecol_UNIQUE` (`Email` ASC) VISIBLE,
CONSTRAINT `fk_DatiUtente_Utente1`
    FOREIGN KEY (`IDUtente`)
    REFERENCES `ProgettoDB`.`Utente` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```



-- Table `ProgettoDB`.`Storia`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Storia` (
  `ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `IDPubblicazione` INT UNSIGNED NULL,
  `IDUtente` INT UNSIGNED NOT NULL,
  `Timestamp` TIMESTAMP NOT NULL,
  `Descrizione` VARCHAR(250) NOT NULL,
  `Tipo` TINYINT UNSIGNED NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_Pubblicazione_has_Utente_Utente2_idx` (`IDUtente` ASC) INVISIBLE,
  INDEX `fk_Pubblicazione_has_Utente_Pubblicazione1_idx` (`IDPubblicazione` ASC) VISIBLE,
  INDEX `indextipo` (`Tipo` ASC) VISIBLE,
  CONSTRAINT `fk_Pubblicazione_has_Utente_Pubblicazione1`
    FOREIGN KEY (`IDPubblicazione`)
      REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)
      ON DELETE SET NULL
      ON UPDATE CASCADE,
  CONSTRAINT `fk_Pubblicazione_has_Utente_Utente2`
    FOREIGN KEY (`IDUtente`)
      REFERENCES `ProgettoDB`.`Utente` (`ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
ENGINE = InnoDB;
```

-- Table `ProgettoDB`.`Autore`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Autore` (
  `ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Cognome` VARCHAR(45) NULL,
  PRIMARY KEY (`ID`),
  INDEX `index2` (`Cognome` ASC) VISIBLE,
  INDEX `index3` (`Nome` ASC) VISIBLE)
ENGINE = InnoDB;
```

-- Table `ProgettoDB`.`Scritto`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Scritto` (
  `IDPubblicazione` INT UNSIGNED NOT NULL,
  `IDAutore` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`IDPubblicazione`, `IDAutore`),
  INDEX `fk_Pubblicazione_has_Autore_Autore1_idx` (`IDAutore` ASC) VISIBLE,
  INDEX `fk_Pubblicazione_has_Autore_Pubblicazione1_idx` (`IDPubblicazione` ASC) VISIBLE,
  CONSTRAINT `fk_Pubblicazione_has_Autore_Pubblicazione1`
    FOREIGN KEY (`IDPubblicazione`)
      REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  CONSTRAINT `fk_Pubblicazione_has_Autore_Autore1`
    FOREIGN KEY (`IDAutore`)
      REFERENCES `ProgettoDB`.`Autore` (`ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
ENGINE = InnoDB;
```

-- Table `ProgettoDB`.`ParolaChiave`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`ParolaChiave` (
  `ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `ParolaChiave` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `Parola Chiave_UNIQUE` (`ParolaChiave` ASC) VISIBLE)
ENGINE = InnoDB;
```

-- Table `ProgettoDB`.`Tag`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Tag` (
  `IDPubblicazione` INT UNSIGNED NOT NULL,
  `IDParolaChiave` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`IDPubblicazione`, `IDParolaChiave`),
  INDEX `fk_Pubblicazione_has_Parole Chiave_Parole Chiave1_idx` (`IDParolaChiave` ASC) VISIBLE,
  INDEX `fk_Pubblicazione_has_Parole Chiave_Pubblicazione1_idx` (`IDPubblicazione` ASC) VISIBLE,
  CONSTRAINT `fk_Pubblicazione_has_Parole Chiave_Pubblicazione1`
    FOREIGN KEY (`IDPubblicazione`)
    REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Pubblicazione_has_Parole Chiave_Parole Chiave1`
    FOREIGN KEY (`IDParolaChiave`)
    REFERENCES `ProgettoDB`.`ParolaChiave` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

-- Table `ProgettoDB`.`DatiModeratore`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`DatiModeratore` (
  `IDUtente` INT UNSIGNED NOT NULL,
  `NumPubb` INT UNSIGNED NOT NULL DEFAULT 0,
  `DataPromozione` DATE NOT NULL,
  `Promotore` INT UNSIGNED NULL,
  PRIMARY KEY (`IDUtente`),
  INDEX `fk_Promotore_idx` (`Promotore` ASC) VISIBLE,
  CONSTRAINT `fk_DatiModeratori_Utente1`
    FOREIGN KEY (`IDUtente`)
    REFERENCES `ProgettoDB`.`Utente` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Promotore`
    FOREIGN KEY (`Promotore`)
    REFERENCES `ProgettoDB`.`Utente` (`ID`)
    ON DELETE SET NULL
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

-- Table `ProgettoDB`.`Ristampa`

```
CREATE TABLE IF NOT EXISTS `ProgettoDB`.`Ristampa` (
```

```

`ID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
`IDPubblicazione` INT UNSIGNED NOT NULL,
`DataRistampa` DATE NOT NULL,
`Numero` TINYINT UNSIGNED NOT NULL,
INDEX `fk_RISTAMPA_Pubblicazione1_idx` (`IDPubblicazione` ASC) VISIBLE,
PRIMARY KEY (`ID`),
INDEX `Ordine Tempo` (`DataRistampa` DESC) INVISIBLE,
CONSTRAINT `fk_RISTAMPA_Pubblicazione1`
  FOREIGN KEY (`IDPubblicazione`)
  REFERENCES `ProgettoDB`.`Pubblicazione` (`ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;
  
```

Implementazioni delle operazioni richieste

1. Modifica del livello di un utente (da attivo a passivo e viceversa)

a. Modifica attivo/passivo (EVENTO)

```

CREATE EVENT PassaggioAttivoPassivo ON SCHEDULE EVERY 1 WEEK STARTS CONCAT(CURDATE(), ' 03:00:00')
DO
  UPDATE utente SET Ruolo = 1 WHERE Ruolo = 2 AND datediff(CURDATE(), DataUltimaRecensione ) > 60;
  
```

b. Modifica passivo/attivo (tale operazione è implementata nell'operazione 9 : “Aggiungi Recensione”, qui abbiamo riportato il frammento di procedura interessato)

```

-- aggiornamento del ruolo se l'utente è passivo --

IF ( ( SELECT utente.ruolo FROM utente WHERE utente.ID = idUtente ) = 1 )
THEN
  UPDATE utente SET utente.ruolo = 2 WHERE ID = idUtente;
END IF;
  
```

2. Estrazione elenco delle ultime dieci pubblicazioni inserite

```

SELECT * FROM Pubblicazione p INNER JOIN Storia s ON p.ID = s.IDPubblicazione
WHERE Tipo = 1 ORDER BY Timestamp DESC LIMIT 10
  
```

ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazion	DataUltimaModif	ID	IDPubblicazione	IDUtente	Timestamp	Descrizione
14	3247969999999	titolo	Rizzoli	0	0	2000-11-11	2019-07-18	104	14	12	2019-07-18 17:31:45	ricky66 ha inserito la pubblic
12	9788804716198	Come una notte a Bali	Mondadori	0	0	2019-07-09	2019-07-18	10	12	12	2019-07-18 16:36:00	ricky66 ha inserito la pubblic
13	9788893255653	La vendetta di Oreste.	Rizzoli	0	0	2019-07-11	2019-07-18	11	13	12	2019-07-18 16:36:00	ricky66 ha inserito la pubblic
6	9788809869097	L'eredità di Agneta. Le signo...	Giunti Editore	2	0	2019-07-10	2019-07-18	4	6	12	2019-07-18 16:35:59	ricky66 ha inserito la pubblic
7	9788865946732	Il visitatore	Nutrimenti	4	0	2019-06-27	2019-07-18	5	7	12	2019-07-18 16:35:59	ricky66 ha inserito la pubblic
8	9788868369989	Mister Napoleone	Piemme	1	2	2019-06-18	2019-07-18	6	8	12	2019-07-18 16:35:59	ricky66 ha inserito la pubblic
9	9788870915686	Norvegia. The passenger	Rizzoli	1	2	2019-06-18	2019-07-18	7	9	12	2019-07-18 16:35:59	ricky66 ha inserito la pubblic
10	9788856671254	Cuore di lupo	Mondadori	0	2	2019-07-09	2019-07-18	8	10	12	2019-07-18 16:35:59	ricky66 ha inserito la pubblic
11	9788817141086	Falso in bilancia	Mondadori	0	0	2019-07-02	2019-07-18	9	11	12	2019-07-18 16:35:59	ricky66 ha inserito la pubblic
4	9788893990172	Quasi è il mio nome	96 Rue De La ...	3	2	2019-06-22	2019-07-18	2	4	12	2019-07-18 16:35:58	ricky66 ha inserito la pubblic

3. Estrazione elenco delle pubblicazioni aggiornate negli ultimi 30 giorni

```
SELECT DISTINCT p.ID, s.Tipo, p.Titolo
FROM Pubblicazione p INNER JOIN Storia s ON p.ID = s.IDPubblicazione
WHERE s.Tipo = 3 AND DATEDIFF (now(), s.Timestamp) < 30
```

	ID	Tipo	Titolo
▶	1	3	Senzanima. Buio.
	3	3	Limbo. Pensieri inversi
	4	3	Quasi è il mio nome
	5	3	Un mondo migliore. Ritratti
	6	3	L'eredità di Agneta. Le signore di Lowenhof
	7	3	Il visitatore
	8	3	Mister Napoleone
	9	3	Norvegia. The passenger
	10	3	Cuore di lupo
	11	3	Falso in bilancia
	12	3	Come una notte a Bali
	13	3	La vendetta di Oreste.

4. Estrazione elenco dei 30 utenti più “collaborativi” (cioè quelli che hanno inserito più pubblicazioni -> ovviamente moderatori)

```
SELECT u.* ,d.* ,m.* FROM utente u INNER JOIN datiutente d ON u.ID = d.IDUtente
INNER JOIN datimoderatore m ON u.ID = m.IDUtente
ORDER BY NumPubb DESC LIMIT 30
```

	ID	Nickname	Password	Ruolo	Utei	DataUltimaRecensione	NumPubb	Nome	Cognome	Residenza	DataNascita	LuogoNascita	Email
▶	12	ricky66	secretpassword	3	0	2019-07-18	11	riicardo	mariani	Chieti, Italia	1993-02-05	Chieti, Italia	ricky@gmail.com
	11	franz94	ssecretpassword	3	0	2019-07-18	1	Francesco	Rossi	Milano, Italia	1993-02-05	Chieti, Italia	francesco.utente@gmail.com
	1	Admin	passwordsegrata	5	0	NULL	0	Admin	Sistema	Chieti, Italia	1993-02-05	Chieti, Italia	admin@gmail.com
	10	Mario92	secretpassword	3	0	NULL	0	Mario	Rossi	L'Aquila, Italia	1992-01-07	Roma, Italia	rossi.mario@student.univaq.it

5. Estrazione elenco delle pubblicazioni inserite da un utente (ID = 12)

```

1 •  SELECT P.Titolo,P.Editore,P.NumLike
2   FROM Pubblicazione P
3   INNER JOIN Storia S
4   On P.ID = S.IDPubblicazione
5   WHERE S.IDUtente = 12 AND S.Tipo = 1
6

```

	Titolo	Editore	NumLike
▶	Quasi è il mio nome	96 Rue De La ...	3
	Un mondo migliore. Ritratti	Bompiani	3
	L'eredità di Agneta. Le signo...	Giunti Editore	2
	Il visitatore	Nutrimenti	4
	Mister Napoleone	Piemme	1
	Norvegia. The passenger	Rizzoli	1
	Cuore di lupo	Mondadori	0
	Falso in bilancia	Mondadori	0
	Come una notte a Bali	Mondadori	0
	La vendetta di Oreste.	Rizzoli	0
	titolo	Rizzoli	0

6. Estrazione catalogo cioè elenco di tutte le pubblicazioni con Titolo, Editore, Autore e anno pubb ordinato per titolo

```

1 •  SELECT p.Titolo,p.Editore, YEAR(P.DataPubblicazione) as 'Anno Pubblicazione',
2   GROUP_CONCAT( concat( Nome, ' ', Cognome) SEPARATOR ', ') as autori
3   FROM pubblicazione p
4   INNER JOIN scritto s
5   ON p.ID = s.IDPubblicazione
6   INNER JOIN autore a
7   ON a.ID = s.IDAutore
8   GROUP BY p.ID
9   ORDER BY p.Titolo

```

	Titolo	Editore	Anno Pubblicazione	autori
▶	Cuore di lupo	Mondadori	2019	Italo Calvino
	Falso in bilancia	Mondadori	2019	Stefano Benni
	Il visitatore	Nutrimenti	2019	Dante Alighieri
	L'eredità di Agneta. Le signore di Lowenhof	Giunti Editore	2019	Agatha Christie
	Limbo. Pensieri inversi	Rizzoli	2019	NULL
	Mister Napoleone	Piemme	2019	Dante Alighieri, Oscar Wilde
	Norvegia. The passenger	Rizzoli	2019	NULL
	Quasi è il mio nome	96 Rue De La Fontaine Edizioni	2019	Italo Calvino, Stefano Benni, It...
	Senzanima. Buio.	Sergio Bonelli	2019	NULL
	Un mondo migliore. Ritratti	Bompiani	2019	Stephen King

7. Estrazione dati completi di una pubblicazione specifica, dato il suo ID (ID=4)

```

1 •  SELECT p.* , m.* , aut.autori , prl.parole , rist.lis Ristampe
2   FROM Pubblicazione p
3   INNER JOIN Metadati m
4   ON p.ID = m.IDPubblicazione
5   INNER JOIN (
6     SELECT s.IDPubblicazione , GROUP_CONCAT( concat( Nome , ' ' , Cognome) SEPARATOR ',' ) as autori
7     FROM scritto s
8     INNER JOIN autore a
9     ON s.IDAutore = a.ID
10    GROUP BY s.IDPubblicazione
11  ) aut
12  ON aut.IDPubblicazione = p.ID
13  INNER JOIN (
14    SELECT t.IDPubblicazione , GROUP_CONCAT( concat(ParolaChiave) SEPARATOR ',' ) as parole
15    FROM tag t
16    INNER JOIN parolachiave pc
17    ON t.IDParolaChiave = pc.ID
18    GROUP BY t.IDPubblicazione
19  ) prl
20  ON prl.IDPubblicazione = p.ID
21  INNER JOIN (
22    SELECT r.IDPubblicazione , GROUP_CONCAT( concat( R.Numero , ' ristampa : ' , R.DataRistampa) SEPARATOR ',' ) as lis
23    FROM ristampa r
24    GROUP BY r.IDPubblicazione
25  ) rist
26  ON rist.IDPubblicazione = p.ID
27  WHERE p.ID = 4
28  GROUP BY p.ID

```

	ID	ISBN	Titolo	Editore	Num	NumF	DataPubblica	DataUltimaMod	Npag	Lingua	Descr	Ini	autori	parole	Ristampe
▶	4	9788893990172	Qua...	Scono...	3	2	2019-06-22	2019-07-18	109	Italiano	De...	I...	Italo Calvino, Stefano Benni, It...	storico, comico, gossip	1 ristampa : 2006-11-11, 2 ristampa : 20



8. Ricerca di pubblicazioni per ISBN, titolo, autore, e parole chiave.

per Parole Chiave

```
1 •  SELECT p.*  
2   FROM pubblicazione p  
3   INNER JOIN tag t  
4   ON t.IDPubblicazione = p.ID  
5   INNER JOIN parolachiave pc  
6   ON pc.ID = t.IDParolaChiave  
7   WHERE pc.ParolaChiave = "giallo"  
8
```

	ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazione	DataUltimaModifica
▶	8	9788868369989	Mister Napoleone	Piemme	1	2	2019-06-18	2019-07-18

per Titolo

```
1 •  SELECT pubblicazione.*  
2   FROM pubblicazione  
3   WHERE pubblicazione.titolo = "Mister Napoleone"  
4
```

	ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazione	DataUltimaModifica
▶	8	9788868369989	Mister Napoleone	Piemme	1	2	2019-06-18	2019-07-18
*		NULL NULL	NULL	NULL	NULL	NULL	NULL	NULL

per ISBN

```
1 •  SELECT pubblicazione.*  
2   FROM pubblicazione  
3   WHERE pubblicazione.ISBN = "9788868369989"  
4
```

	ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazione	DataUltimaModifica
▶	8	9788868369989	Mister Napoleone	Piemme	1	2	2019-06-18	2019-07-18
*		NULL NULL	NULL	NULL	NULL	NULL	NULL	NULL



per Autore

La procedura per la ricerca delle pubblicazioni per autore è stata creata per permettere all'utente che sta cercando le pubblicazioni di cercare inserendo solo il nome dell'autore, solo il cognome, oppure inserendo entrambi

```
1 • CREATE PROCEDURE `ricercaAutori`(`nm` VARCHAR(45), `cgnm` VARCHAR(45) )
2     BEGIN
3         IF ( `nm` IS NULL ) THEN
4             SELECT p.* 
5                 FROM pubblicazione p
6                 INNER JOIN scritto s ON s.IDPubblicazione = p.ID
7                 INNER JOIN autore a ON s.IDAutore = a.ID
8                 WHERE a.cognome = cgnm;
9         ELSEIF( `cgnm` IS NULL ) THEN
10            SELECT p.* 
11                FROM pubblicazione p
12                INNER JOIN scritto s ON s.IDPubblicazione = p.ID
13                INNER JOIN autore a ON s.IDAutore = a.ID
14                WHERE a.nome = nm;
15        ELSE
16            SELECT p.* 
17                FROM pubblicazione p
18                INNER JOIN scritto s ON s.IDPubblicazione = p.ID
19                INNER JOIN autore a ON s.IDAutore = a.ID
20                WHERE a.nome = nm AND a.cognome = cgnm;
21    END IF;
22 END
```

```
1 • call progettodb.ricercaAutori('Oscar', 'Wilde');
```

	ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazione	DataUltimaModifica
▶	8	9788868369989	Mister Napoleone	Piemme	1	2	2019-06-18	2019-07-18

9. Inserimento ed eliminazione di una recensione relativa a una pubblicazione

```
1 • CREATE PROCEDURE `inserimentoRecensione`( idUtente INT, idPubb INT, Des TEXT)
2   BEGIN
3     DECLARE dataoggi DATE;
4     DECLARE tit VARCHAR(100);
5     DECLARE nick VARCHAR(45);
6     DECLARE stamp DATETIME;
7
8     SET stamp = NOW();
9     SELECT NickName INTO nick FROM utente WHERE ID = idUtente;
10    SELECT Titolo INTO tit FROM pubblicazione WHERE ID = idPubb;
11
12    INSERT INTO Recensione ( `IDPubblicazione`, `IDUtente`, `Descrizione`, `Timestamp` )
13    VALUES (IDPubb, IDUtente, Des, stamp);
14
15    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
16    ( idPubb, idUtente, stamp, CONCAT( nick, " ha inserito una recensione alla pubblicazione : ", tit), 5 );
17
18    UPDATE utente SET utente.DataUltimaRecensione = DATE(stamp) WHERE ID = idUtente;
19    UPDATE utente SET Ruolo = 2 WHERE ID = idUtente;
20  END
```

chiamata:

```
1 • call inserimentoRecensione(12, 14, "molto bello")
2 |
```

risultato

recensione:

	IDPubblicazione	IDUtente	Descrizione	FlagApprovazione	ApprovatoDa	Timestamp
▶	14	12	molto bello	0	NULL	2019-07-19 08:44:30

storia:

	ID	IDPubblicazione	IDUtente	Timestamp	Descrizione	Tipo
▶	106	14	12	2019-07-19 08:44:30	ricky66 ha inserito una recensione alla pubblicazione : Bar Sport	3

```
1 • CREATE DEFINER=`root`@`localhost` PROCEDURE `delRecensione`(`idUtnt` INT, `idPubb` INT, `idElimin` INT )
2   BEGIN
3     DECLARE tit VARCHAR(100);
4     DECLARE nick VARCHAR(45);
5     DECLARE nickDel VARCHAR(45);
6     DECLARE approvato BIT;
7     SELECT NickName into nick FROM utente WHERE ID = idUtnt;
8     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
9     SELECT NickName into nickDel FROM utente WHERE ID = idElimin;
10    SELECT FlagApprovazione into approvato FROM recensione WHERE IDPubblicazione = idPubb AND IDUtente = idUtnt;
11
12    DELETE FROM recensione WHERE IDPubblicazione = idPubb AND IDUtente = idUtnt;
13    IF ( approvato = 1 ) THEN
14      UPDATE pubblicazione SET NumRec = NumRec - 1 WHERE ID = idPubb;
15    END IF;
16    IF ( idUtnt = idElimin ) THEN
17      INSERT INTO Storia (IDPubblicazione, IDUtente, `Timestamp`, Descrizione, Tipo) VALUES
18      (idPubb, idElimin, now(), CONCAT( nick, " ha eliminato la propria recensione per la pubblicazione: ",tit ),6 );
19    ELSE
20      INSERT INTO Storia (IDPubblicazione, IDUtente, `Timestamp`, Descrizione, Tipo) VALUES
21      (idPubb, idElimin, now(), CONCAT( nickDel, " ha eliminato la recensione di", nick, "per la pubblicazione: ",tit ),6 );
22    END IF;
--
```

chiamata:

```
1 • call progettodb.delRecensione(12, 14, 10);
2 |
```

risultato

storia:

	ID	IDPubblicazione	IDUtente	Timestamp	▼	Descrizione	Tipo
▶	109	14	10	2019-07-19 08:58:58		Mario92 ha eliminato la recensione diricky66per la pubblicazione: Bar Sport	6

10. Approvazione di una recensione (da parte del moderatore)

```

1 • CREATE PROCEDURE `ApprovazioneRecensione`( IDAppr INT, IDPubb INT, IDUtnt INT)
2   BEGIN
3     DECLARE tit VARCHAR(100);
4     DECLARE nick VARCHAR(45);
5     SELECT NickName into nick FROM utente WHERE ID = IDUtnt;
6     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
7
8     IF ( ( SELECT FlagApprovazione FROM Recensione r WHERE r.IDPubblicazione = IDPubb AND r.IDUtente = IDUtnt ) = 0 ) THEN
9       UPDATE Recensione r SET r.FlagApprovazione = 1 WHERE r.IDPubblicazione = IDPubb AND r.IDUtente = IDUtnt;
10      UPDATE Recensione r SET r.ApprovatoDa = IDAppr WHERE r.IDPubblicazione = IDPubb AND r.IDUtente = IDUtnt;
11
12      INSERT INTO Storia (IDPubblicazione, IDUtente, `Timestamp`, Descrizione, Tipo)
13      VALUES (IDPubb, IDAppr, now(), concat( "è stata approvata la recensione di: ", nick, " del libro: ", tit ), 4 );
14
15      UPDATE Pubblicazione P SET P.NumRec = P.NumRec + 1 WHERE P.ID = IDPubb;
16    END IF;
17  END
  
```

chiamata:

```

1 • call progettodb.ApprovazioneRecensione(10, 14, 12);
  
```

risultato

recensione:

	IDPubblicazione	IDUtente	Descrizione	FlagApprovazione	ApprovatoDa	Timestamp
▶	14	12	molto bello	1	10	2019-07-19 08:44:30

storia:

	ID	IDPubblicazione	IDUtente	Timestamp	Descrizione	Tipo
▶	107	14	10	2019-07-19 08:50:02	è stata approvata la recensione di: ricky66 del libro: Bar Sport	4



11. Inserimento ed eliminazione di un like relativo a una pubblicazione

```
1 • CREATE PROCEDURE inserimentoLike ( idUtente INT , idPubb INT )
2   BEGIN
3
4     DECLARE ora TIMESTAMP;
5     DECLARE tit VARCHAR(100);
6     DECLARE nick VARCHAR(45);
7
8     SET ora = now();
9     SELECT NickName into nick FROM utente WHERE ID = idUtente;
10    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
11
12    INSERT INTO `Like` VALUES (idPubb, idUtente, ora);
13
14    INSERT INTO Storia (IDPubblicazione, IDUtente, Timestamp, Descrizione, Tipo) VALUES
15      (idPubb, idUtente, ora,
16       CONCAT( nick, " ha messo like alla pubblicazione: ",tit ),7 );
17
18    UPDATE Pubblicazione P SET P.NumLike = P.NumLike + 1 WHERE P.ID = idPubb;
19
20  END
```

chiamata:

```
1 • call progettodb.inserimentoLike(12, 14);
```

risultato

like:

	IDPubblicazione	IDUtente	Timestamp
▶	14	12	2019-07-19 08:52:54

storia:

	ID	IDPubblicazione	IDUtente	Timestamp	Descrizione	Tipo
▶	108	14	12	2019-07-19 08:52:54	ricky66 ha messo like alla pubblicazione: Bar Sport	7



```
1 • CREATE DEFINER=`root`@`localhost` PROCEDURE `delLike`( idUtnt INT , idPubb INT )
2 BEGIN
3
4     DECLARE tit VARCHAR(100);
5     DECLARE nick VARCHAR(45);
6
7     SELECT NickName into nick FROM utente WHERE ID = idUtnt;
8     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
9
10    DELETE FROM `Like` WHERE `Like`.IDUtente = idUtnt AND `Like`.IDPubblicazione = idPubb;
11
12    INSERT INTO storia ( `IDPubblicazione` , `IDUtente` , `Timestamp` , `Descrizione` , `Tipo` ) VALUES
13        ( idPubb, idUtnt, now(), CONCAT( nick, " ha rimosso il like alla pubblicazione : ", tit), 8 );
14
15    UPDATE Pubblicazione P SET P.NumLike = P.NumLike - 1 WHERE P.ID = idPubb;
16 END
```

chiamata:

```
1 • call progettodb.delLike(12, 14);
```

risultato

storia:

ID	IDPubblicazione	IDUtente	Timestamp	Descrizione	Tipo
115	12	12	2019-07-19 09:19:45	ricky66 ha rimosso il like alla pubblicazione : Come una notte a Bali	8

12. Calcolo numero dei like per una pubblicazione dato l'ID della pubblicazione (ID = 1)

```
1 • SELECT NumLike
2   FROM Pubblicazione
3   WHERE ID = 1;
```

NumLike
2

13. Estrazione elenco delle recensioni approvate per una pubblicazione dato l'ID della pubblicazione (ID = 3)

```

1 •  SELECT *
2   FROM recensione
3   WHERE IDPubblicazione = 3 && FlagApprovazione = 1;
4

```

	IDPubblicazione	IDUtente	Descrizione	FlagApprovazione	ApprovatoDa	Timestamp
▶	3	11	testo	1	11	2019-07-18 00:00:00
	3	12	testo	1	11	2019-07-18 00:00:00
*	HULL	HULL	NULL	NULL	HULL	HULL

14. Estrazione elenco delle recensioni in attesa di approvazione.

```

1 •  SELECT *
2   FROM recensione
3   WHERE FlagApprovazione = 0;
4

```

	IDPubblicazione	IDUtente	Descrizione	FlagApprovazione	ApprovatoDa	Timestamp
▶	3	19	testo	0	HULL	2019-07-18 00:00:00
	4	19	testo	0	HULL	2019-07-18 00:00:00
	5	20	testo	0	HULL	2019-07-18 00:00:00
	6	12	testo	0	HULL	2019-07-18 00:00:00
	6	14	testo	0	HULL	2019-07-18 00:00:00
	6	20	testo	0	HULL	2019-07-18 00:00:00
	7	12	testo	0	HULL	2019-07-18 00:00:00
	7	15	testo	0	HULL	2019-07-18 00:00:00
*	HULL	HULL	NULL	NULL	HULL	HULL

15. Estrazione log delle modifiche effettuate su una pubblicazione dato l'ID della pubblicazione (ID = 1)

```

1 •  SELECT *
2   FROM storia
3   WHERE Tipo = 3 && IDPubblicazione = 1
4   ORDER BY `Timestamp` DESC
5

```

	ID	IDPubblicazione	IDUtente	Timestamp	Descrizione	Tipo
▶	44	1	1	2019-07-18 16:54:21	Admin ha aggiunto l'autore 1 alla pubblicazione : Senzanim. Buio.	3
	25	1	10	2019-07-18 16:51:09	Mario92 ha aggiunto una sorgente alla pubblicazione : Senzanim. Buio.	3
	14	1	1	2019-07-18 16:49:11	Admin ha aggiunto una ristampa alla pubblicazione : Senzanim. Buio.	3
	15	1	1	2019-07-18 16:49:11	Admin ha aggiunto una ristampa alla pubblicazione : Senzanim. Buio.	3
	13	1	1	2019-07-18 16:47:24	Admin ha aggiunto una ristampa alla pubblicazione : Senzanim. Buio.	3
	12	1	1	2019-07-18 16:47:03	Admin ha aggiunto una ristampa alla pubblicazione : Senzanim. Buio.	3
*	HULL	HULL	HULL	HULL	NULL	NULL



16. Estrazione elenco delle pubblicazioni per le quali è disponibile un download.

```
1 •   SELECT p.Titolo, p.Editore, p.DataPubblicazione
2     FROM sorgente s
3     JOIN pubblicazione p
4       ON s.IDPubblicazione = p.ID
5     WHERE Tipo = "download"
6
```

	Titolo	Editore	DataPubblicazione
▶	Senzanima. Buio.	Sergio Bonelli	2019-06-14
	Limbo. Pensieri inversi	Rizzoli	2019-07-02
	Quasi è il mio nome	96 Rue De ...	2019-06-22
	Un mondo migliore. Ritratti	Bompiani	2019-06-19
	L'eredità di Agneta. Le signo...	Giunti Editore	2019-07-10
	Il visitatore	Nutrimenti	2019-06-27
	Cuore di lupo	Mondadori	2019-07-09
	La vendetta di Oreste.	Rizzoli	2019-07-11

17. Estrazione della lista delle pubblicazioni in catalogo, ognuna con la data dell'ultima ristampa

```
1 •   SELECT P.*, UR.UltimaRistampa
2     FROM Pubblicazione P
3     INNER JOIN
4     (
5       SELECT R.IDPubblicazione as ID, max(R.DataRistampa) as UltimaRistampa
6         FROM Ristampa R
7       GROUP BY IDPubblicazione
8     ) UR
9     ON P.ID = UR.ID
```

	ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazione	DataUltimaModifica	UltimaRistampa
▶	1	1201201860328	Senzanima. Buio.	Sergio Bonelli	2	0	2019-06-14	2019-07-18	2001-11-11
	3	9788817138789	Limbo. Pensieri inversi	Rizzoli	3	2	2019-07-02	2019-07-18	2005-11-11
	4	9788893990172	Quasi è il mio nome	96 Rue De La ...	3	2	2019-06-22	2019-07-18	2008-11-11
	5	9788830101821	Un mondo migliore. Ritratti	Bompiani	3	2	2019-06-19	2019-07-18	2011-11-11

**18. Data una pubblicazione, restituire tutte le pubblicazioni del catalogo aventi gli stessi autori (ID = 1)**

```
1 •  SELECT DISTINCT p.*  
2   FROM pubblicazione p  
3   INNER JOIN scritto s1  
4   ON p.ID = s1.IDPubblicazione  
5   INNER JOIN scritto s2  
6   ON s1.IDAutore = s2.IDAutore  
7   WHERE s2.IDPubblicazione = 1  
8  |
```

	ID	ISBN	Titolo	Editore	NumLike	NumF	DataPubblicazione	DataUltimaModifica
▶	1	1201201860328	Senzanima. Buio.	Sergio Bonelli	2	0	2019-06-14	2019-07-18
	3	9788817138789	Limbo. Pensieri inversi	Rizzoli	3	2	2019-07-02	2019-07-18
	8	9788868369989	Mister Napoleone	Piemme	1	2	2019-06-18	2019-07-18

Implementazione delle operazioni aggiuntive

□ Inserimento Pubblicazione

```
1 • CREATE PROCEDURE `inserimentoPubb` (IN idUtnt INT, IN ISBNPubb CHAR(13), IN titl VARCHAR(45), IN editr VARCHAR(45),
2   IN dataPubb DATE, IN nPg INT, IN ling VARCHAR(45) , IN descrzn TEXT, IN indc TEXT, OUT idPubb INT )
3   BEGIN
4     DECLARE nick VARCHAR(45);
5     DECLARE stamp DATETIME;
6     SET stamp = NOW();
7
8     SELECT Nickname FROM utente WHERE ID = idUtnt INTO nick;
9
10    UPDATE datimoderatore SET NumPubb = NumPubb + 1 WHERE IDUtente = idUtnt ;
11
12    INSERT INTO pubblicazione ( `ISBN`, `Titolo`, `Editore`, `DataUltimaModifica`, `DataPubblicazione` )
13    VALUES (ISBNPubb,titl,editr,DATE(stamp),dataPubb);
14    SET idPubb = LAST_INSERT_ID();
15    INSERT INTO metadati ( `IDPubblicazione`, `Npag`, `Lingua`, `Descrizione`, `Indice` )
16    VALUES (idPubb,nPg,ling,descrzn,indc);
17
18    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` )
19    VALUES ( idPubb,idUtnt,stamp, concat( nick, " ha inserito la pubblicazione titolata: ", titl), 1);
20  END
```

□ Inserimento Utente

```
1   DELIMITER #
2
3 • CREATE PROCEDURE `insUtente`
4   (nick VARCHAR(45), email VARCHAR(254), pass VARCHAR(45), nome VARCHAR(45),
5   cogn VARCHAR(45), dNascita DATE, lNascita VARCHAR(100), residenza VARCHAR(200) )
6
7   BEGIN
8
9     INSERT INTO utente ( `Nickname`, `Password` ) VALUES ( nick, pass );
10    INSERT INTO datiutente( `IDUtente`, `Nome`, `Cognome`, `DataNascita`, `LuogoNascita`, `Residenza`, `Email` )
11    VALUES ( LAST_INSERT_ID(), nome, cogn, dNascita, lNascita, residenza,email );
12
13  END
```

Esempio chiamata

```
CALL insUtente("Mario92" , "mario.rossi@student.univaq.it", "secretpass",
"Mario", "Rossi", '1992-01-07' , "Roma, Italia", "L'Aquila, Italia" )
```



□ Modifica attributi Pubblicazione

□ Modifica Titolo

```
1  DELIMITER #
2
3
4 • CREATE PROCEDURE `modificaTitolo` ( idPubb INT, tit VARCHAR(100), idUtente INT )
5  BEGIN
6      DECLARE stamp DATETIME;
7      DECLARE nick VARCHAR(45);
8      SET stamp = NOW();
9      SELECT NickName into nick FROM utente WHERE ID = idUtente;
10
11     UPDATE pubblicazione p SET p.Titolo = tit WHERE p.ID = idPubb;
12     UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
13
14     INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
15     ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato il titolo della pubblicazione in : ", tit), 3 );
16 END#
```

□ Modifica ISBN

```
1  DELIMITER #
2
3 • CREATE PROCEDURE `modificaISBN` ( idPubb INT, codiceisbn CHAR(13), idUtente INT )
4
5  BEGIN
6      DECLARE nick VARCHAR(45);
7      DECLARE tit VARCHAR(100);
8
9      SELECT NickName into nick FROM utente WHERE ID = idUtente;
10     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
11
12     UPDATE pubblicazione p SET p.ISBN = codiceisbn WHERE p.ID = idPubb;
13     UPDATE pubblicazione p SET p.DataUltimaModifica = CURDATE() WHERE p.ID = idPubb;
14
15     INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
16     ( idPubb, idUtente, NOW(), CONCAT( nick, " ha cambiato il codice ISBN della pubblicazione : ", tit, " in ", codiceisbn), 3 );
17
18 END#
```

□ Modifica Editore

```
1  DELIMITER #
2
3 • CREATE PROCEDURE `modificaEditore` ( idPubb INT, Edit VARCHAR(45), idUtente INT )
4
5  BEGIN
6      DECLARE nick VARCHAR(45);
7      DECLARE tit VARCHAR(100);
8      DECLARE stamp DATETIME;
9      SET stamp = NOW();
10
11     SELECT NickName into nick FROM utente WHERE ID = idUtente;
12     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
13
14     UPDATE pubblicazione p SET p.Editore = Edit WHERE p.ID = idPubb;
15     UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
16
17     INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
18     ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato l'editore della pubblicazione : ", tit, " in ", Edit), 3 );
19 END#
```

□ Modifica Data di Pubblicazione

```

1   DELIMITER #
2
3 •  CREATE PROCEDURE `modificaDataPubb` ( idPubb INT, dt DATE, idUtente INT )
4
5   BEGIN
6     DECLARE nick VARCHAR(45);
7     DECLARE tit VARCHAR(100);
8     DECLARE stamp DATETIME;
9     SET stamp = NOW();
10
11    SELECT NickName into nick FROM utente WHERE ID = idUtente;
12    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
13
14    UPDATE pubblicazione p SET p.DataPubblicazione = dt WHERE p.ID = idPubb;
15    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
16
17    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
18      ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato la data di pubblicazione di : ", tit, " in ", dt ), 3 );
19
END

```

□ Modifica Numero di Pagine

```

1   DELIMITER #
2
3 •  CREATE PROCEDURE `modificaNPag` ( idPubb INT, np SMALLINT, idUtente INT )
4
5   BEGIN
6     DECLARE nick VARCHAR(45);
7     DECLARE tit VARCHAR(100);
8     DECLARE stamp DATETIME;
9     SET stamp = NOW();
10
11    SELECT NickName into nick FROM utente WHERE ID = idUtente;
12    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
13
14    UPDATE metadati m SET m.NPag = np WHERE m.IDPubblicazione = idPubb;
15    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
16
17    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
18      ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato il numero di pagine della pubblicazione : ", tit, " in ", np ), 3 );
19
END#

```

□ Modifica Lingua

```

1   DELIMITER #
2
3 •  CREATE PROCEDURE `modificaLingua` ( idPubb INT, lin VARCHAR(45), idUtente INT )
4   BEGIN
5     DECLARE nick VARCHAR(45);
6     DECLARE tit VARCHAR(100);
7     DECLARE stamp DATETIME;
8     SET stamp = NOW();
9
10    SELECT NickName into nick FROM utente WHERE ID = idUtente;
11    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12
13    UPDATE metadati m SET m.Lingua = lin WHERE m.IDPubblicazione = idPubb;
14    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
15    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
16      ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato la lingua della pubblicazione : ", tit, " in ", lin ), 3 );
17
END#

```



□ Modifica Descrizione

```
1  DELIMITER #
2
3 •  CREATE PROCEDURE `modificaDescrizione` ( idPubb INT, des TEXT, idUtente INT )
4  BEGIN
5      DECLARE nick VARCHAR(45);
6      DECLARE tit VARCHAR(100);
7      DECLARE stamp DATETIME;
8      SET stamp = NOW();
9
10     SELECT NickName into nick FROM utente WHERE ID = idUtente;
11     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12     UPDATE metadati m SET m.Descrizione = des WHERE m.IDPubblicazione = idPubb;
13     UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
14
15     INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
16         ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato la descrizione della pubblicazione : ", tit, " in ", des), 3 );
17
18 END
```

□ Modifica Indice

```
1  DELIMITER #
2
3 •  CREATE PROCEDURE `modificaIndice` ( idPubb INT, ind TEXT, idUtente INT )
4  BEGIN
5      DECLARE nick VARCHAR(45);
6      DECLARE tit VARCHAR(100);
7      DECLARE stamp DATETIME;
8      SET stamp = NOW();
9
10     SELECT NickName into nick FROM utente WHERE ID = idUtente;
11     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12
13     UPDATE metadati m SET m.Indice = ind WHERE m.IDPubblicazione = idPubb;
14     UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
15
16     INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
17         ( idPubb, idUtente, stamp, CONCAT( nick, " ha cambiato l'indice della pubblicazione : ", tit, " in ", ind), 3 );
18
19 END#
```

□ Inserimento nuova Parola Chiave da Pubblicazione

```
1  DELIMITER #
2
3 •  CREATE PROCEDURE `insParolaChiave` ( idPubb INT, idpc INT, idUtente INT )
4
5  BEGIN
6      DECLARE nick VARCHAR(45);
7      DECLARE tit VARCHAR(100);
8      DECLARE stamp DATETIME;
9      SET stamp = NOW();
10     SELECT NickName into nick FROM utente WHERE ID = idUtente;
11     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12     INSERT INTO tag(IDPubblicazione, IDParolaChiave) VALUES ( idPubb, idpc );
13     UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
14     INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
15         ( idPubb, idUtente, stamp, CONCAT( nick, " ha aggiunto una parola chiave alla pubblicazione : ", tit), 3 );
16
17 END#
```

□ Eliminazione Parola Chiave da Pubblicazione

```

1   DELIMITER #
2
3 • CREATE PROCEDURE `delParolaChiave` ( idPubb INT, idpc INT, idUtente INT )
4 BEGIN
5   DECLARE nick VARCHAR(45);
6   DECLARE tit VARCHAR(100);
7   DECLARE stamp DATETIME;
8   SET stamp = NOW();
9
10  SELECT NickName into nick FROM utente WHERE ID = idUtente;
11  SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12  DELETE FROM tag WHERE tag.IDPubblicazione = idPubb AND tag.IDParolaChiave = idpc;
13  UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
14  INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
15  ( idPubb, idUtente, stamp, CONCAT( nick, " ha rimosso una parola chiave alla pubblicazione : ", tit), 3 );
16 END

```

□ Inserimento nuovo Autore da Pubblicazione

```

1   DELIMITER #
2
3 • CREATE PROCEDURE `insAutore` ( idPubb INT, ida INT, idUtente INT )
4 BEGIN
5   DECLARE nick VARCHAR(45);
6   DECLARE tit VARCHAR(100);
7   DECLARE stamp DATETIME;
8   SET stamp = NOW();
9
10  SELECT NickName into nick FROM utente WHERE ID = idUtente;
11  SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12  INSERT INTO scritto(IDPubblicazione,IDAutore) VALUES ( idPubb, ida );
13  UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
14  INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
15  ( idPubb, idUtente, stamp, CONCAT( nick, " ha aggiunto l'autore ", ida, " alla pubblicazione : ", tit), 3 );
16 END

```

□ Eliminazione Autore da Pubblicazione

```

1   DELIMITER #
2
3 • CREATE PROCEDURE `delAutore` ( idPubb INT, ida INT, idUtente INT )
4 BEGIN
5   DECLARE nick VARCHAR(45);
6   DECLARE tit VARCHAR(100);
7   DECLARE stamp DATETIME;
8   SET stamp = NOW();
9   SELECT NickName into nick FROM utente WHERE ID = idUtente;
10  SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
11
12  DELETE FROM scritto WHERE scritto.IDPubblicazione = idPubb AND scritto.IDAutore = ida;
13
14  UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
15
16  INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
17  ( idPubb, idUtente, stamp, CONCAT( nick, " ha rimosso un autore alla pubblicazione : ", tit), 3 );
18 END#

```



▢ Inserimento Ristampa

```
1  DELIMITER #
2
3 • CREATE PROCEDURE `insRistampa` ( idPubb INT, dtRis DATE, num TINYINT, idUtente INT )
4
5 BEGIN
6     DECLARE nick VARCHAR(45);
7     DECLARE tit VARCHAR(100);
8     DECLARE stamp DATETIME;
9     SET stamp = NOW();
10    SELECT NickName into nick FROM utente WHERE ID = idUtente;
11    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12    INSERT INTO ristampa(IDPubblicazione, DataRistampa, Numero) VALUES ( IDPubb, dtRis, num );
13    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
14    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
15    ( idPubb, idUtente, stamp, CONCAT( nick, " ha aggiunto una ristampa alla pubblicazione : ", tit), 3 );
16 END#
```

▢ Eliminazione Ristampa

```
1  DELIMITER #
2
3 • CREATE PROCEDURE `delRistampa` ( idPubb INT, idRistampa INT, idUtente INT )
4 BEGIN
5     DECLARE nick VARCHAR(45);
6     DECLARE tit VARCHAR(100);
7     DECLARE stamp DATETIME;
8     SET stamp = NOW();
9     SELECT NickName into nick FROM utente WHERE ID = idUtente;
10    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
11
12    DELETE FROM ristampa WHERE ID = idRistampa;
13
14    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
15
16    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
17    ( idPubb, idUtente, stamp, CONCAT( nick, " ha eliminato una ristampa della pubblicazione : ", tit), 3 );
18 END
```

▢ Inserimento Sorgente

```
1  DELIMITER #
2
3 • CREATE PROCEDURE `insSorgente`
4 ( idPubb INT, ur VARCHAR(2083), tp VARCHAR(45), frmt VARCHAR(45), des TEXT, idUtente INT )
5 BEGIN
6     DECLARE nick VARCHAR(45);
7     DECLARE tit VARCHAR(100);
8     DECLARE stamp DATETIME;
9     SET stamp = NOW();
10    SELECT NickName into nick FROM utente WHERE ID = idUtente;
11    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12
13    INSERT INTO sorgente( IDPubblicazione, URI, Tipo, Formato, Descrizione ) VALUES ( IDPubb, ur, tp, frmt, des );
14
15    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
16
17    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
18    ( idPubb, idUtente, stamp, CONCAT( nick, " ha aggiunto una sorgente alla pubblicazione : ", tit), 3 );
19 END
```



□ Eliminazione Sorgente

```
1   DELIMITER #
2
3 •  CREATE PROCEDURE `delSorgente` ( idPubb INT, idSorgente INT, idUtente INT )
4
5   BEGIN
6     DECLARE nick VARCHAR(45);
7     DECLARE tit VARCHAR(100);
8     DECLARE stamp TIMESTAMP;
9     SET time = NOW();
10    SELECT NickName into nick FROM utente WHERE ID = idUtente;
11    SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
12
13    DELETE FROM sorgente WHERE ID = idSorgente;
14
15    UPDATE pubblicazione p SET p.DataUltimaModifica = DATE(stamp) WHERE p.ID = idPubb;
16
17    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
18      ( idPubb, idUtente, stamp, CONCAT( nick, " ha rimosso una sorgente della pubblicazione : ", tit), 3 );
19
END
```

□ Promozione a Moderatore

```
1   DELIMITER #
2
3 •  CREATE PROCEDURE `promModer` ( idProm INT, idUtente INT )
4
5   BEGIN
6
7     UPDATE utente u SET u.Ruolo = 3 WHERE u.ID = idUtente;
8     INSERT INTO datimoderatore( IDUtente, NumPubb, DataPromozione, Promotore)
9       VALUES ( IDUtente, 0, CURDATE(), idProm );
10
11 END
```

□ Eliminazione Pubblicazione

```
1   DELIMITER #
2
3 •  CREATE PROCEDURE `delPubb` ( idPubb INT, idUtente INT )
4   BEGIN
5     DECLARE nick VARCHAR(45);
6     DECLARE tit VARCHAR(100);
7     DECLARE idIns INT;
8     SELECT NickName into nick FROM utente WHERE ID = idUtente;
9     SELECT Titolo into tit FROM pubblicazione WHERE ID = idPubb;
10    SELECT IDUtente into idIns FROM storia WHERE Tipo = 1 AND IDPubblicazione = idPubb;
11
12    UPDATE datiModeratore D SET D.NumPubb = D.NumPubb - 1 WHERE D.IDUtente = idIns;
13
14    DELETE FROM pubblicazione WHERE ID = idPubb;
15
16    INSERT INTO storia ( `IDPubblicazione`, `IDUtente`, `Timestamp`, `Descrizione`, `Tipo` ) VALUES
17      ( NULL, idUtente, NOW(), CONCAT( nick, " ha eliminato la pubblicazione titolata : ", tit), 2 );
18
END
```



❑ Eliminazione Utente

```
1   DELIMITER #
2
3 •  CREATE PROCEDURE `delUtente` ( idUtnt INT )
4  BEGIN
5      UPDATE utente SET UtenteEliminato = 1 WHERE D.IDUtente = idUtente;
6      DELETE FROM datiUtente WHERE IDUtente = idUtnt;
7      DELETE FROM datiModeratore WHERE IDUtente = idUtnt;
8  END
```

❑ Cerca Utente per

❑ ID

```
SELECT u.* , d.*
FROM utente u
LEFT JOIN datiutente d
ON u.ID = d.IDUtente
WHERE u.ID = 2
```

❑ Email

```
SELECT u.* , d.*
FROM datiutente d
INNER JOIN utente u
ON u.ID = d.IDUtente
WHERE d.Email = ""
```

❑ Nickname

```
SELECT u.* , d.*
FROM utente u
LEFT JOIN datiutente d
ON u.ID = d.IDUtente
WHERE u.Nickname = ""
```

□ Eliminazione degli autori a cascata:

Questa procedura viene chiamata nel momento in cui un utente vuole eliminare un autore dalla tabella ‘autore’, fa uso di un cursore per eliminare in loop tutto le occorrenze di quell’autore nella tabella ‘Scritto’ tramite le chiamate a ‘delAutore’ in questo modo le modifiche fatte alle pubblicazioni a cui è stato eliminato quell’autore verranno salvate all’interno della tabella storia

```
1 •  CREATE PROCEDURE `delCascadeAutore`(IN idAut INT, IN idUtente INT)
2   BEGIN
3     DECLARE done INT DEFAULT FALSE;
4     DECLARE v_idPubb INT;
5     DECLARE v_idAutore INT;
6     DECLARE cursore CURSOR FOR (SELECT * FROM Scritto s WHERE idAut = s.IDAutore);
7     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
8     OPEN cursore;
9
10    ciclo: LOOP
11      FETCH cursore INTO v_idPubb, v_idAutore;
12      IF(done = TRUE) THEN
13        LEAVE ciclo;
14      END IF;
15
16      -- eliminazione degli autori e scrittura su storia
17      CALL delAutore(v_idPubb, idAut, idUtente);
18    END LOOP;
19    CLOSE cursore;
20    DELETE FROM Autore WHERE ID = idAut;
21  END
```

□ Eliminazione delle parole chiave a cascata:

Questa procedura viene chiamata nel momento in cui un utente vuole eliminare una parola chiave dalla tabella ‘parolachiave’, fa uso di un cursore per eliminare in loop tutto le occorrenze di quella parola chiave nella tabella ‘Tag’ tramite le chiamate a ‘delParolaChiave’ in questo modo le modifiche fatte alle pubblicazioni a cui è stato eliminata quella parola verranno salvate all’interno della tabella storia

```
1 •  CREATE PROCEDURE `delCascadeTag` (IN idPC INT, IN idUtente INT)
2   BEGIN
3     DECLARE done INT DEFAULT FALSE;
4     DECLARE v_idPubb INT;
5     DECLARE v_idPC INT;
6     DECLARE cursore CURSOR FOR (SELECT * FROM Tag t WHERE idPC = t.IDParolaChiave);
7     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
8     OPEN cursore;
9
10    ciclo: LOOP
11      FETCH cursore INTO v_idPubb, v_idPC;
12      IF(done = TRUE) THEN
13        LEAVE ciclo;
14      END IF;
15
16      -- eliminazione degli autori e scrittura su storia
17      CALL delParolaChiave(v_idPubb, idPC, idUtente);
18    END LOOP;
19    CLOSE cursore;
20
21    DELETE FROM ParolaChiave WHERE ID = idPC;
22  END
```

TRIGGER

□ Check password trigger:

Questo Trigger si occupa di controllare il formato del campo password di *Utente*. Infatti utilizza un'espressione regolare per verificare il corretto formato (cifre esadecimali) e la lunghezza.

```

1   DELIMITER $$ 
2 • CREATE TRIGGER checkPasswordTrigger BEFORE INSERT
3   ON `progettodb`.`utente`
4   FOR EACH ROW
5   BEGIN
6       IF NOT(SELECT NEW.`Password` REGEXP '[0-9a-fA-F]{32}') 
7   THEN
8       SIGNAL SQLSTATE '45000' SET message_text = "La stringa inserita in Password non è valida";
9   END IF;
10  END$$

```

I seguenti due trigger sono per il controllo di correttezza al momento di inserimento (1) e aggiornamento (2) della ristampa e controllano che l'ordine cronologico delle ristampe rispetti l'ordine numerico.

□ Check Inserimento ristampa trigger

```

1   DELIMITER $$ 
2 • CREATE TRIGGER validazioneInserimentoRistampeTrigger BEFORE INSERT
3   ON `progettodb`.`ristampa`
4   FOR EACH ROW
5   BEGIN
6       DECLARE idPubb INTEGER;
7       SET idPubb = NEW.`IdPubblicazione`;
8
9       IF NOT EXISTS  (SELECT * FROM Ristampa R
10      WHERE R.IDPubblicazione = idPubb AND R.Numero = NEW.Numero)
11   THEN
12       IF ( SELECT MAX(R.DataRistampa) as RistampaPrecedente FROM Ristampa R
13          WHERE R.IDPubblicazione = idPubb AND R.Numero < NEW.Numero ) > NEW.DataRistampa
14   THEN
15           SIGNAL SQLSTATE '45000' SET message_text = "ERRORE: Esiste una ristampa di numero inferiore con data superiore";
16       ELSE IF ( SELECT MIN(R.DataRistampa) as RistampaPrecedente FROM Ristampa R
17          WHERE R.IDPubblicazione = idPubb AND R.Numero > NEW.Numero ) < NEW.DataRistampa
18   THEN
19           SIGNAL SQLSTATE '45000' SET message_text = "ERRORE: Esiste una ristampa di numero superiore con data inferiore";
20       END IF;
21   END IF;
22   ELSE
23       SIGNAL SQLSTATE '45000' SET message_text = "La ristampa esiste già";
24   END IF;
25   END$$

```

□ check update ristampa trigger

```
1   DELIMITER $$  
2 • CREATE TRIGGER validazioneAggiornamentoRistampeTrigger BEFORE INSERT  
3   ON `progettodb`.`ristampa`  
4   FOR EACH ROW  
5   BEGIN  
6       DECLARE idPubb INTEGER;  
7       SET idPubb = NEW.`IdPubblicazione`;  
8  
9  
10      IF ( SELECT MAX(R.DataRistampa) as RistampaPrecedente FROM Ristampa R  
11          WHERE R.IDPubblicazione = idPubb AND R.Numero < NEW.Numero ) > NEW.DataRistampa  
12      THEN  
13          SIGNAL SQLSTATE '45000' SET message_text =  
14          "ERRORE: Esiste una ristampa di numero inferiore con data superiore";  
15      ELSE IF ( SELECT MIN(R.DataRistampa) as RistampaPrecedente FROM Ristampa R  
16          WHERE R.IDPubblicazione = idPubb AND R.Numero > NEW.Numero ) < NEW.DataRistampa  
17      THEN  
18          SIGNAL SQLSTATE '45000' SET message_text =  
19          "ERRORE: Esiste una ristampa di numero superiore con data inferiore";  
20      END IF;  
21      END IF;  
22  END$$
```

Divisione del lavoro

Per quanto riguarda i primi step dello sviluppo del progetto, ovvero progettazione concettuale, progettazione logica e operazioni richieste, è stata svolta da entrambi i componenti. Per il resto, Fulvio Lapenna si è occupato della maggior parte delle procedure aggiuntive e dell'evento e Federico Di Menna delle restanti procedure, Triggers, e Slides.