Leddit

Progetto Basi di Dati

Luca Polese Matricola 1225425
Alessandro Poloni Matricola 1224444

1. Abstract

Leddit è un popolare sito americano fondato nel 2010 che consente ai suoi utenti di condividere e commentare notizie di varia natura, organizzate per argomento all'interno di sezioni conosciute come “subleddit". Ad ogni utente, previa registrazione, viene offerta la possibilità di creare subleddit ed iscriversi a quelli che più lo interessano, creare post al loro interno e commentarli; è inoltre offerta la possibilità di creare uno o più feed personalizzati, che consentono di raggruppare sub diversi selezionati dall’utente.
Per valutare la qualità dei contenuti, Leddit utilizza un sistema di "upvotes" e "downvotes", ovvero voti positivi e negativi, che determinano la posizione dei post all’interno del sito.
Il sito offre anche un sistema di "premi", conosciuti come “awards”, che un utente può acquistare e assegnare ai commenti generati da altri utenti, supportando direttamente il sito. Un utente che abbia acquistato un award viene considerato anche un utente “premium”, al quale viene garantito accesso a subleddit riservati a questa categoria di utenti.
Tra gli utenti del sito si distinguono i moderatori, ossia i responsabili della gestione di uno specifico subleddit, e gli amministratori, cioè i dipendenti del sito.
Per tutelare gli utenti, subleddit e post possono essere contrassegnati come "nsfw" ("not safe for work").
Infine, il sito offre anche basilari funzionalità "social”, quali la messaggistica tra utenti e la possibilità di avere una lista di amici.

2. Analisi dei requisiti

2.1 Descrizione

Si vuole realizzare una base di dati per l'organizzazione e la gestione degli utenti, dei post, dei commenti e delle sezioni del sito web Leddit.
Nell'atto dell'iscrizione, l'utente che voglia registrarsi deve inserire:

- uno username, univoco all’interno della piattaforma;
- una password personale, per poter accedere al sito;
- un indirizzo e-mail, per ricevere eventuali informazioni o notifiche.

Viene inoltre memorizzata la data di iscrizione alla piattaforma, per indicare all’utente da quanti giorni è registrato.
Una volta iscritto alla piattaforma, l’utente potrà:

- richiedere l’amicizia di altri utenti della piattaforma;
- inviare/ricevere messaggi da altri utenti (non necessariamente appartenenti alla lista degli amici);
- creare dei feed personalizzati con cui visualizzare i post dei subleddit desiderati;
- creare dei post in vari subleddit che al più contengono un media;
- commentare i post o rispondere ad altri commenti;
- dare un voto positivo (“upvote”) o negativo (“downvote”) ad ogni post o commento;
- decidere di cancellare la propria iscrizione. I post e commenti effettuati da un utente non verranno rimossi dalla base di dati.

Ogni utente registrato può creare un subleddit (conosciuto semplicemente anche come “sub”), contraddistinto da:

- un nome univoco, di massimo 20 caratteri;
- una descrizione dell’argomento trattato;
- la data in cui è stato creato;
- un flag che segnali se il subleddit è riservato esclusivamente agli utenti premium.

L’utente può inoltre creare diversi feed personalizzati all’interno dei quali raccogliere subleddit che trattano argomenti simili; per ogni feed, che dovrà avere un nome specificato dal relativo proprietario, si dovranno quindi memorizzare i subleddit associati.
All'interno di un subleddit, l'utente può creare un post, che dovrà avere:

- un titolo, di massimo 100 caratteri;
- un contenuto testuale;
- la data di creazione.

Trascorsi 180 giorni dalla sua creazione, un post viene automaticamente archiviato, con annessa

disabilitazione delle funzioni di voto e commento.
Fondamentale è la possibilità di contrassegnare materiale (subleddit e post) come “nsfw”, nel caso in cui il contenuto sia offensivo, volgare od esplicito, ad ogni subleddit e post dovrà quindi essere associato un flag che consenta appunto di segnalare la natura dei suoi contenuti.
Ogni post può essere oggetto di commenti da parte della community, pertanto occorre memorizzare anche tali informazioni, sapendo che i commenti hanno una dimensione massima consentita di 500 caratteri.
È possibile, inoltre, effettuare commenti di risposta ad altri commenti.
Ad un post può anche essere associato un contenuto multimediale scelto dall’utente, come una foto, un video od un link ad un sito esterno, che può avere un nome ed una descrizione, entrambi opzionali. A foto e video l’utente può anche assegnare uno o più tag.
I post ed i commenti possono essere valutati dagli utenti tramite “upvotes” e “downvotes”, nel caso in cui l’utente rispettivamente apprezzi il contenuto e lo ritenga invece errato o fuori tema. Occorre quindi memorizzare queste informazioni per consentire il calcolo dei voti ricevuti in totale, elemento molto importante poiché tale numero determinerà la posizione e la visibilità del post all’interno del sito e del relativo sub.
Per quanto concerne la gestione degli award, essi si suddividono in tre categorie: argento, oro e platino. Tali categorie si distinguono per il loro costo in denaro. Ogni award garantisce all’utente che lo ha acquistato di entrare nella categoria degli utenti “premium”. All’interno della base di dati per ogni award saranno quindi memorizzati:

- il commento al quale è stato assegnato;
- l’utente che lo ha acquistato;
- il tipo;
- il prezzo;
- la data di acquisto.

Gli utenti del sito si distinguono in tre categorie: gli utenti base (eventualmente premium), i moderatori e gli amministratori. Per ogni utente bisogna pertanto memorizzarne il ruolo. Inoltre, nel caso di un moderatore, bisogna anche memorizzare i subleddit all’interno del quale copre l’incarico. Ogni moderatore può inoltre nominare a sua volta altri moderatori. Un utente che crea un sub diviene in automatico moderatore di quest’ultimo. Di estrema importanza per i moderatori è la possibilità di bannare gli utenti: il ban può essere temporaneo, e in tal caso avrà una data di scadenza, oppure permanente. Il moderatore è inoltre tenuto a specificare le ragioni del ban. I moderatori possono anche cancellare post e commenti all’interno del relativo sub.
In conclusione, il sito offre anche funzionalità social: ogni utente può inviare una richiesta di amicizia ad un altro utente, perciò è necessaria la registrazione di tale vincolo tra coppie di utenti; per di più, gli utenti possono scambiarsi messaggi, ognuno dei quali avrà un mittente ed un destinatario, un corpo ed una data di invio.

2.2 Glossario dei termini

Termine	Descrizione	Collegamenti
Utente	Utente normale del sito	Sub, Post, Commento, Moderatore, Feed, Award, Messaggio, Ban
Moderatore	Utente che può gestire post, commenti e ban di utenti all’interno di un Sub	Sub, Utente, Ban
Sub	Sezione del sito dedicata ad uno specifico argomento	Utente, Post, Ban, Moderatore, Feed
Post	Contenuto testuale, eventualmente con media allegati, inserito da un utente all’interno di un Sub	Sub, Utente, Commento, Media
Commento	Commento di un utente ad un post	Utente, Post, Award
Award	Premio donato da un utente ad un commento	Utente, Commento
Media	Contenuto multimediale come foto, video o link allegato ad un Post da parte di un Utente	Post
Feed	Insieme di Sub che trattano argomenti simili selezionati dall’Utente	Sub, Utente
Ban	Informazione relativa al ban effettuato da un moderatore nei confronti di un utente	Utente, Moderatore, Sub
Messaggio	Messaggio di testo inviato privatamente da un utente ad un altro	Utente

2.3 Strutturazione dei requisiti

Frasi relative ad **Utente**
Nell'atto dell'iscrizione, l'utente che voglia registrarsi deve inserire: - uno username, univoco all'interno della piattaforma; - una password personale, per poter accedere al sito; - un indirizzo e-mail, per ricevere eventuali informazioni o notifiche. Viene inoltre memorizzata la data di iscrizione alla piattaforma, per indicare all'utente da quanti giorni è registrato. Una volta iscritto alla piattaforma, l'utente potrà: - richiedere l'amicizia di altri utenti della piattaforma - inviare/ricevere messaggi da altri utenti (non necessariamente appartenenti alla lista degli amici) - creare dei feed personalizzati con cui visualizzare i post dei subleddit desiderati - creare dei post in vari subleddit. Ogni post può contenere dei media - commentare sotto ad ogni post, o sotto ad altri commenti - dare un voto positivo ("upvote") o negativo ("downvote") ad ogni post o commento - decidere di cancellare la propria iscrizione, tuttavia post e commenti da lui effettuati non verranno rimossi dalla base di dati. Gli utenti del sito si distinguono in tre categorie: gli utenti base (eventualmente premium), i moderatori e gli amministratori. Per ogni utente bisogna pertanto memorizzarne il ruolo.
Frasi relative a **Moderatore**
Gli utenti del sito si distinguono in tre categorie: gli utenti base (eventualmente premium), i moderatori […] I moderatori possono anche cancellare post e commenti all'interno del relativo sub. Per ogni utente bisogna pertanto memorizzarne il ruolo […] diviene in automatico moderatore di quest'ultimo.
Frasi relative a **Sub**
Ogni utente registrato può creare un subleddit, contraddistinto da: - un nome univoco, di massimo 20 caratteri; - una descrizione dell'argomento trattato; - la data in cui è stato creato; - un flag che segnali se il subleddit è riservato esclusivamente agli utenti premium. Fondamentale è la possibilità di contrassegnare materiale […] come "nsfw", nel caso in cui il contenuto sia offensivo, volgare od esplicito, ad ogni subleddit […] dovrà quindi essere associato un flag che consenta appunto di segnalare la natura dei suoi contenuti.
Frasi relative a **Post**
All'interno di un subleddit, l'utente può creare un post, che dovrà avere: - un titolo, di massimo 100 caratteri; - un contenuto testuale; - la data di creazione. Trascorsi 180 giorni dalla sua creazione, un post viene automaticamente archiviato, con annessa disabilitazione delle funzioni di voto e commento. I post […] possono essere valutati […] del post all'interno del sito e del relativo sub.
Frasi relative a **Commento**
Ogni post può essere oggetto di commenti da parte della community […] i commenti hanno una dimensione massima consentita di 500 caratteri. È possibile, inoltre, effettuare commenti di risposta ad altri commenti. […] ed i commenti possono essere valutati dagli utenti tramite "upvotes" e "downvotes" […] per consentire il calcolo dei voti ricevuti in totale
Frasi relative a **Media**
Ad un post può anche essere associato un contenuto multimediale scelto dall'utente, come una foto, un

video od un link ad un sito esterno, che può avere un nome ed una descrizione, entrambi opzionali. A foto e video l'utente può anche assegnare uno o più tag.
Frasi relative ad **Award**
[…] si suddividono in tre categorie: argento, oro e platino. Tali categorie si distinguono per il costo in denaro. […] Ogni award garantisce all'utente che lo ha acquistato di entrare nella categoria degli utenti "premium". All'interno della base di dati per ogni award saranno quindi memorizzati: - il commento al quale è stato assegnato; - l'utente che lo ha acquistato; - il tipo; - il prezzo; - la data di acquisto.
Frasi relative a **Feed**
L'utente può inoltre creare diversi feed personalizzati all'interno dei quali raccogliere subleddit che trattano argomenti simili; per ogni feed, che dovrà avere un nome specificato dal relativo proprietario, si dovranno quindi memorizzare i subleddit associati.
Frasi relative a **Ban**
[…] possibilità di bannare gli utenti: il ban può essere temporaneo, e in tal caso avrà una data di scadenza, oppure permanente. Il moderatore è inoltre tenuto a specificare le ragioni del ban.
Frasi relative a **Messaggio**
[…] gli utenti possono scambiarsi messaggi, ognuno dei quali avrà un mittente ed un destinatario, un corpo ed una data di invio.

2.4 Operazioni sul Database

Tipo di operazione	L/S	Quantità stimata (al giorno)
Login al sito	L	4000
Iscrizione al sito da parte di nuovi utenti	S	190
Eliminazione di un account	S	50
Assegnazione di un award da parte di un utente	S	180
Aggiunta di un amico da parte di un utente	S	200
Creazione di nuovi subleddit	S	20
Creazione di nuovi post	S	1500
Creazione di nuovi commenti	S	5000
Creazione di nuovi media	S	300
Accesso ad un post	L	20000
Accesso ad un commento	L	100000
Accesso a media	L	6000
Eliminazione di un post	S	50
Eliminazione di un commento da parte di un utente	S	100
Votazione ad un post	S	16000
Votazione ad un commento	S	8000
Accesso ai messaggi personali	L	2500
Invio di messaggio privato ad un altro account	S	35000
Modifica del contenuto di un post	S	100
Modifica del contenuto di un commento	S	1500

3. Progettazione concettuale

3.1 Lista delle entità

Gli attributi sottolineati costituiscono la chiave primaria, i campi sono NOT NULL salvo diversamente specificato.

Utente		
<u>username</u>	varchar(20)	Username dell'utente
password	varchar(30)	Password dell'utente
email	varchar(320)	Indirizzo mail dell'utente
dataIscrizione	date	Data di iscrizione al sito
isAdmin	bit	Bit che segnala se l'utente in questione è amministratore del sito

Sub		
nome	varchar(20)	Nome del subleddit
dataCreazione	date	Data di creazione del subleddit
descrizione	varchar(255)	Descrizione del subleddit
isPremium	bit	Bit che segnala se il subleddit è riservato ad utenti premium
nsfw	bit	Bit che segnala se il contenuto del subleddit è volgare od esplicito

Post		
ID	int	Identificativo del post
titolo	varchar(100)	Titolo del post
dataCreazione	date	Data di creazione del post
contenuto	varchar(500)	Contenuto testuale del post
nsfw	bit	Bit che segnala se il contenuto del post è volgare od esplicito
numeroVoti	int	Numero dei voti ricevuti

Commento		
ID	int	Identificativo del commento
dataCommento	date	Data del commento
contenuto	varchar(500)	Contenuto testuale del commento
votiCommento	int	Numero dei voti ricevuti

Moderatore (identificatore esterno: attributo “username” dell’entità Utente e attributo “nome” dell’entità Sub)		
dataInizio	date	Data di inizio dell’attività di moderatore all’interno di un sub

Ban (identificatore esterno: attributo “username” dell’entità Utente, attributo “nome” dell’entità Sub chiave esterna: attributi “nome” e “username” dell’entità Moderatore)		
dataBan	date	Data in cui è stato effettuato il ban
scadenza	date	Data di termine del ban, può avere valore NULL (ban permanente)
motivo	varchar(255)	Motivo per cui è stato effettuato il ban

Feed (identificatore esterno: attributo “username” dell’entità Utente)		
nome	varchar(30)	Nome del feed

Award (identificatore esterno: attributo “username” dell’entità Utente e attributo “ID” dell’entità Commento)		
tipo	char	Categoria dell’award
dataAcquisto	date	Data in cui l’award è stato acquistato
costo	decimal(3, 1)	Costo dell’award

Messaggio		
ID	int	Identificativo del messaggio
dataInvio	date	Data di invio del messaggio
contenuto	varchar(255)	Contenuto testuale del messaggio

Media (identificatore esterno: attributo “ID” dell’entità Post)		
nome	varchar(30)	Nome del media, può avere valore NULL
descrizione	varchar(30)	Descrizione del media, può avere valore NULL
percorso	varchar(60)	Percorso del media

Foto (identificatore esterno: attributo “ID” dell’entità Post)		
tag	varchar(100)	Lista di tag associati alla foto

Video (identificatore esterno: attributo “ID” dell’entità Post)		
tag	varchar(100)	Lista di tag associati al video

N.B: le entità “Utente attivo”, “Utente cancellato”, “Commento attivo”, “Commento cancellato”, “Link” non hanno informazioni in più rispetto ai rispettivi genitori, pertanto sono state omesse dalla lista.

3.2 Generalizzazioni

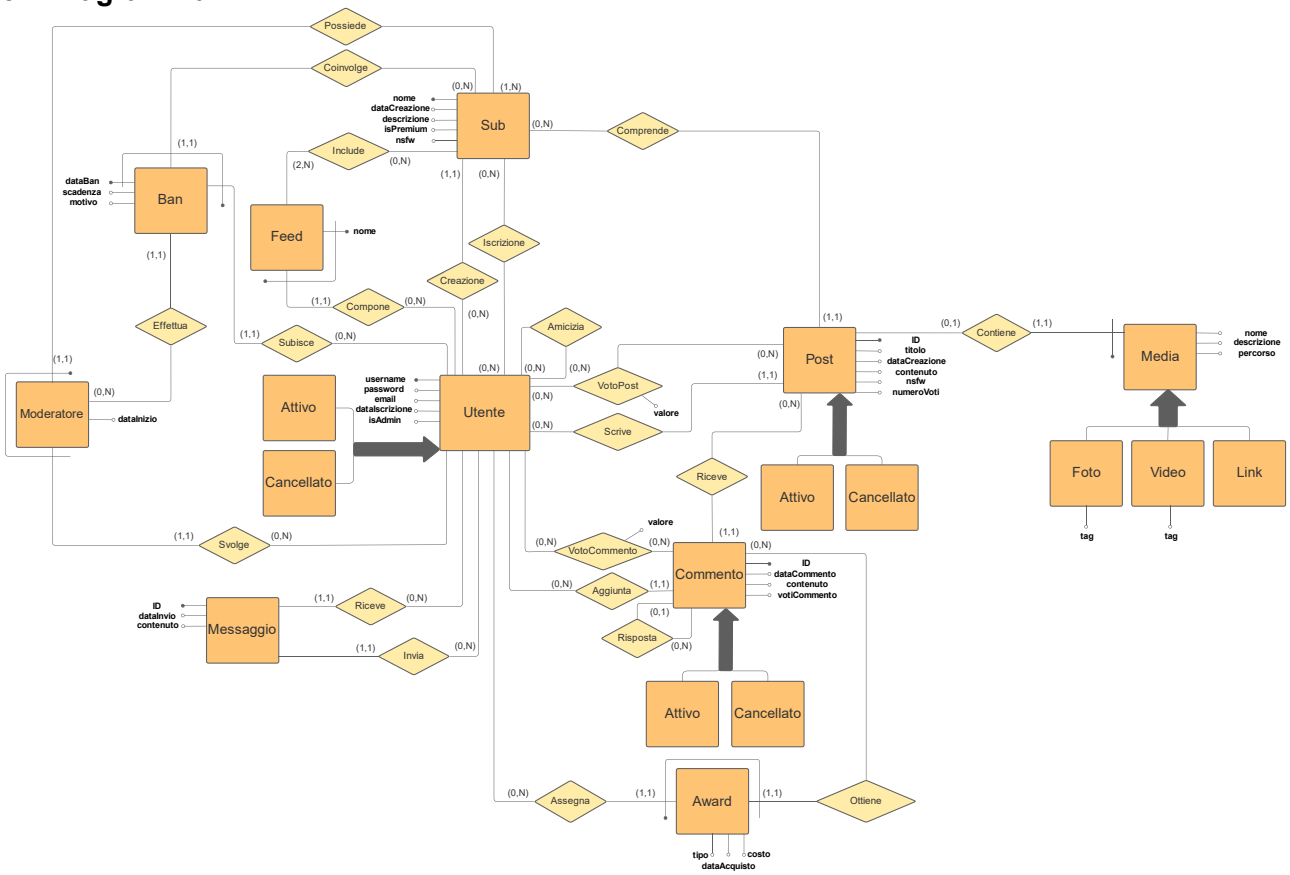
- **Utente** è una generalizzazione totale di **Utente attivo** e **Utente cancellato**
- **Commento** è una generalizzazione totale di **Commento attivo** e **Commento cancellato**
- **Post** è una generalizzazione totale di **Post attivo** e **Post cancellato**
- **Media** è una generalizzazione totale di **Foto**, **Video** e **Link**

3.3 Lista delle relazioni e cardinalità

Entità 1	Entità 2	Nome	Cardinalità
Utente	Sub	Creazione	Un utente può creare uno o più sub (0, N) Un sub è creato da un utente (1, 1)
		Iscrizione	Un utente si può iscrivere ad uno o più sub (0, N) Ad un sub possono iscriversi più utenti (0, N)
	Post	Scrive	Un utente può scrivere uno o più post (0, N) Un post è scritto da un utente (1, 1)
		VotoPost	Un utente può votare uno o più post (0, N) Un post è votato da diversi utenti (0, N)
	Commento	Aggiunta	Un utente può aggiungere 1 o più commenti ai post (0, N) Un commento viene aggiunto da un utente (1, 1)
		VotoCommento	Un utente può votare uno o più post (0, N) Un post è votato da diversi utenti (0, N)
	Award	Assegna	Un utente può assegnare più award (0, N) Un award viene assegnato da un unico utente (1, 1)
	Utente	Amicizia	Un utente può avere più amici (0, N) Un utente può essere amico di più utenti (0, N)
	Messaggio	Invia	Un utente può inviare più messaggi (0, N) Un messaggio viene inviato da un solo utente (1, 1)
		Riceve	Un utente può ricevere più messaggi (0, N) Un messaggio viene ricevuto da un solo utente (1, 1)
	Feed	Compone	Un utente può comporre uno o più feed (0, N) Un feed è composto da un unico utente (1, 1)
	Moderatore	Svolge	Un utente può essere moderatore in più sub (0, N) Il ruolo di moderatore viene svolto da un unico utente (1, 1)
	Ban	Subisce	Un utente può subire uno o più ban (0, N) Un ban viene subito da un unico utente (1, 1)
Sub	Ban	Coinvolge	Un sub può essere coinvolto in più ban (0, N) Un ban coinvolge un singolo sub (1, 1)
	Post	Comprende	Un sub può comprendere uno o più post (0, N) Un post è compreso in un solo sub (1, 1)
Post	Media	Contiene	Un post può contenere un unico media (0, 1) Un media appartiene esclusivamente ad un post (1, 1)
	Commento	Riceve	Un post può ricevere uno o più commenti (0, N) Un commento può appartenere ad un unico post (1, 1)
Commento	Commento	Risposta	Un commento può rispondere ad un altro commento (0, 1) Un commento può ricevere più commenti di risposta (0, N)
	Award	Ottiene	Un commento può ottenere uno o più award (0, N) Un award viene assegnato ad un unico commento (1, 1)
Feed	Sub	Include	Un feed include due o più sub (2, N)

			Un sub può essere incluso in più feed (0, N)
Moderatore	Sub	Possiede	Un moderatore appartiene ad un unico sub (1, 1) Un sub può possedere uno o più moderatori (1, N)
	Ban	Effettua	Un moderatore può effettuare uno o più ban (0, N) Un ban viene effettuato da un unico moderatore (1, 1)

3.4 Diagramma ER



4. Progettazione logica

4.1.1 Analisi delle ridondanze

Nel database sono state riscontrate 2 ridondanze:

1. **attributo numeroVoti in Post**
   Si prende in considerazione l'attributo numeroVoti che definisce il numero totale di voti che un determinato Post ha ricevuto nel suo periodo di attività (180 giorni come definito inizialmente).
   La ridondanza rappresenta il valore dato dalla somma dei valori positivi e negativi che vengono assegnati dagli utenti.
   L'attributo risulta essere ridondante, poiché esso è dato semplicemente dalla somma di tutti i voti assegnati allo specifico post. *Potrebbe* non essere necessario utilizzare un apposito attributo che necessiterebbe di un costante aggiornamento.
   Seguirà quindi l'analisi della ridondanza.
2. **attributo votiCommento in Commento**

Il ragionamento che si può applicare per questa ridondanza è il medesimo applicabile all'attributo numeroVoti, riferito però all'attributo votiCommento dell'entità Commento.

Tabella delle ridondanze

Concetto	Tipo	Volume Totale approssimativo*
Utente	E	15000
Sub	E	39500
Feed	E	20000 (~2 feed / utente)
Ban	E	150
Post	E	165000 (~11 post / utente)
VotoPost	R	4950000 (~30 voti / post)
Commento	E	2475000 (~15 commenti / post)
VotoCommento	R	12375000 (~5 voti / commento)
Media	E	55000(~1 media ogni 3 post)
Messaggio	E	130000 (~10 messaggi / utente)
Moderatore	E	18250 (~2 moderatori / sub)
Award	E	7500 (~1 award ogni 2 utenti)

** Il volume calcolato si basa sulle operazioni previste giornaliere e calcolate in un anno (365 giorni)*

1. Attributo numeroVoti in Post
L'attributo **numeroVoti** all'interno della relazione **Post** è di interesse per due delle operazioni più comuni sulla base di dati:

- l'accesso ai post da parte degli utenti, con una frequenza stimata di circa 20000 accessi giornalieri
- l'aggiunta di un voto ad un post da parte degli utenti, operazione effettuata circa 16000 volte al giorno

Accesso - CON RIDONDANZA				Accesso - SENZA RIDONDANZA			
Concetto	**Costrutto**	**Accesso**	**Tipo**	**Concetto**	**Costrutto**	**Accesso**	**Tipo**
Post	E	1	L	Post	E	1	L
				VotoPost	E	x=30	L

In presenza di ridondanza la valutazione è molto semplice, poiché il numero di accessi da effettuare alla tabella Post corrisponde esattamente al volume definito nella stima riportata nella tabella delle operazioni sul database. In presenza dell'attributo **numeroVoti** all'interno dei post sono perciò sufficienti 20000 accessi alla base di dati.

In mancanza di ridondanza la valutazione è più complessa poiché la lettura dei valori contenuti nei post non sarebbe sufficiente: per svolgere l'intera operazione è necessario, infatti, un accesso alla relazione VotoPost contenente tutti i voti assegnati al Post.
Imposta una media di 30 voti per Post (come da tabella dei volumi), è necessario eseguire 30 volte l'accesso a quest'ultima tabella per calcolare il voto finale del singolo Post.
In mancanza dell'attributo **numeroVoti** all'interno dei Post saranno necessari 20000 accessi (Post) + 30*20000 accessi (VotoPost).

Totale accessi con ridondanza = 20000
Totale accessi senza ridondanza = 20000 + 30*20000 = 620000

Votazione - CON RIDONDANZA				Votazione - SENZA RIDONDANZA			
Concetto	**Costrutto**	**Accesso**	**Tipo**	**Concetto**	**Costrutto**	**Accesso**	**Tipo**
VotoPost	R	1	S	VotoPost	R	1	S
Post	E	1	S				

Per quanto riguarda invece l'aggiunta di un voto ad un post si evince che:

In presenza di ridondanza, per effettuare un voto è necessario:

- registrare il voto nella tabella VotoPost;
- aggiornare il valore dell'attributo numeroVoti dell'entità Post.

Richiamando perciò i volumi stimati, saranno necessari 16000 accessi alla tabella VotoPost e 16000 accessi alla tabella Post.

In mancanza di ridondanza, il numero di operazioni da eseguire risulterebbe essere dato solamente dagli accessi alla tabella Post.

Totale accessi con ridondanza = 2*16000 + 2*16000 = 64000
Totale accessi senza ridondanza = 2*16000 = 32000
Tutti i volumi risultano raddoppiati poiché si assume costo doppio per le operazioni di scrittura.

Valutazioni finali:
Analizzando nel complesso le due operazioni risulta che:
- **in presenza di ridondanza** si effettuano 20000 + 2*16000 + 2*16000 = 84000 operazioni
- **in assenza di ridondanza** si effettuano 20000 + 30*20000 + 2*16000 = 652000 operazioni

Pertanto, si è deciso di conservare l'attributo ridondante.

Va tenuto inoltre in considerazione che, caricando ogni post sulla piattaforma, sarà necessario effettuare la medesima operazione per tutti i commenti ad esso relativi.
Poiché dalla descrizione dell'analisi dei requisiti si evince che ogni commento ha un proprio voto associato, si dovrà trattare allo stesso modo anche la ridondanza dell'attributo votiCommenti, relativo al conteggio dei voti dei commenti.

2. Attributo votiCommenti in Commenti
L'attributo **votiCommento** relativo all'entità **Commento** risulta particolarmente utile in due specifiche operazioni, sulla base di dati, legate alla visualizzazione dei commenti:
- l'accesso ai commenti da parte degli utenti, che secondo le stime ha una frequenza di circa 100000 accessi giornalieri
- l'aggiunta di un voto ad un commento da parte degli utenti, operazione effettuata circa 8000 volte al giorno

Accesso - CON RIDONDANZA				Accesso - SENZA RIDONDANZA			
Concetto	**Costrutto**	**Accesso**	**Tipo**	**Concetto**	**Costrutto**	**Accesso**	**Tipo**
Commento	E	1	L	Commento	E	1	L
				VotoCommento	E	x=5	L

In presenza di ridondanza, si può facilmente sapere quanti sono gli accessi necessari per ottenere i Commenti relativi ai vari Post: il numero di accessi corrisponde, infatti, al volume di esecuzione stimato dell'operazione della base di dati. In presenza dell'attributo **votiCommento** all'interno dei Commenti, sono perciò sufficienti 100000 accessi alla base di dati.

In mancanza di ridondanza, richiede invece una valutazione più complessa, poiché implica sia la lettura dei valori contenuti nella tabella Commento sia quelli contenuti nella tabella VotoCommento; altrimenti le informazioni che si ricaverebbero solamente dalla prima relazione sarebbero incompleti e conseguentemente insufficienti.
Come si può notare dalla tabella dei volumi, mediamente ogni Commento riceve un numero di voti pari a 5. Sarà necessario, perciò, eseguire 5 accessi alla tabella VotoCommento per poter calcolare il voto finale del Commento.
In mancanza dell'attributo **votiCommento** all'interno dei Commenti, saranno necessari 100000 accessi (Commento) + 5*100000 accessi (VotoCommento)

Totale accessi con ridondanza = 100000
Totale accessi senza ridondanza = 100000 + 5*100000 = 600000

Votazione - CON RIDONDANZA				Votazione - SENZA RIDONDANZA			
Concetto	**Costrutto**	**Accesso**	**Tipo**	**Concetto**	**Costrutto**	**Accesso**	**Tipo**
VotoCommento	R	1	S	VotoCommento	R	1	S
Commento	E	1	S				

Per quanto riguarda invece l'aggiunta di un voto ad un Commento si evince che:
In presenza di ridondanza, sarà necessario registrare sia il voto nella tabella VotoCommento sia incrementare o decrementare la valutazione del Commento che ha ricevuto tale voto.
Richiamando perciò i volumi stimati, saranno necessari 8000 accessi alla tabella VotoCommento e 8000 accessi alla tabella Commento.

In mancanza di ridondanza, il numero di operazioni da eseguire risulterebbe essere dato solamente dagli accessi alla tabella Commento.

Totale accessi con ridondanza = 2*8000 + 2*8000 = 32000
Totale accessi senza ridondanza = 2*8000 = 16000
Tutti i volumi risultano raddoppiati poiché si assume costo doppio per le operazioni di scrittura.

Valutazioni finali:
Analizzando nel complesso le due operazioni si ha che:

- **in presenza di ridondanza** si effettuano 100000 + 2*8000 + 2*8000 =132000 operazioni;
- **in assenza di ridondanza** si effettuano 100000 + 5*100000 + 2*8000=616000 operazioni.

Pertanto, si è deciso di conservare anche questo attributo ridondante.

4.1.2 Eliminazione delle generalizzazioni

Nello schema concettuale sono presenti quattro generalizzazioni, tutte di tipo totale.
La prima riguarda l'entità Utente, che si distingue nei due figli Utente attivo ed Utente cancellato. Tali entità figlie possiedono gli stessi attributi del padre: risulta quindi una buona soluzione accorparle all'interno del padre, memorizzando la distinzione tra le due all'interno della base di dati tramite un semplice attributo booleano, il cui valore 0 identifica un utente attivo, mentre il valore 1 un utente cancellato.
Un'analoga risoluzione è stata adottata anche per le entità figlie della generalizzazione Commento (Attivo e Cancellato) e per le entità figlie della generalizzazione Post (Attivo e Cancellato); avverranno anche in questi casi degli accorpamenti delle entità figlie della generalizzazione nei relativi genitori.
Generalizzazione più complessa è invece quella che riguarda l'entità Media e le figlie Foto, Video e Link. Le entità Foto e Video hanno gli stessi attributi del padre come Link ma, a differenza di questo, possiedono anche un ulteriore attributo "tag": la generalizzazione viene quindi risolta, anche in questo caso, accorpando le figlie nel padre, aggiungendo un attributo "tipo" che permette di distinguere i diversi media e consentendo all'attributo tag di avere valore nullo, nel caso in cui il media sia un link.

4.1.3 Scelta degli identificatori primari

Identificatore primario non banale è quello della tabella **Media**: poiché quest'ultima è coinvolta in un'unica relazione di tipo (1,1) con l'entità **Post**, ogni media può essere identificato univocamente tramite l'identificatore del relativo post di appartenenza, che diventa perciò identificatore primario.
Per quanto riguarda la scelta dell'identificatore nella tabella **Award**, possiamo dire che, poiché l'utente può assegnare un solo award ad un commento, l'award può essere identificato univocamente tramite le chiavi primarie dell'utente e del commento.
Tale ragionamento si estende anche al caso della tabella **Moderatore**.
Leggermente differente è l'opzione che si è adottata per la tabella **Ban**: nonostante le associazioni con Sub, Utente e Moderatore siano (1:N), si è scelto di non inserire la chiave di **Moderatore** nella chiave privata, in quanto non aggiunge valore alla chiave; necessariamente diverrà parte della chiave esterna della tabella.
Analogamente, **Feed** nella propria chiave conterrà solo l'identificatore esterno della tabella **Utente**.
Infine, un altro identificatore interessante è quello della tabella **Commento**: una possibile scelta sarebbe stata l'utilizzo del nome dell'utente e dell'identificativo del post, tuttavia dai requisiti non emerge alcun vincolo per cui un utente possa commentare un post una sola volta, pertanto si assume che lo stesso utente possa effettuare più commenti. Si è quindi deciso di utilizzare come chiave primaria un identificativo che incrementa automaticamente.

4.1.4 Partizionamento/accorpamento di entità e relazioni

Entità Award
L'entità Award possiede l'attributo prezzo che dipende dal tipo di Award. Poiché vi sono solo tre categorie di award, tale attributo risulta ridondante; viene quindi effettuata una decomposizione verticale dell'entità Award che porta alla creazione dell'entità *Prezzo*. Tra Award e Prezzo sarà presente una relazione di tipo uno a molti.
Si è scelto inoltre di non limitare le tipologie di Award disponibili, nell'ipotesi di nuove aggiunte future.

Generalizzazione Media
La generalizzazione Media ha tre entità figlie: *Link*, *Foto*, *Video*.
La scelta di accorpare le entità figlie nel padre è dettata da più fattori:

- *Link* non contiene nessun attributo aggiuntivo rispetto all'entità padre;
- *Foto* e *Video* contengono lo stesso attributo che perciò le rende indistinguibili fra loro.

Si è perciò deciso di accorpare l'attributo "tag" (che può avere valore nullo) all'interno di Media e di aggiungere un attributo "tipo" che contraddistingua il tipo di media (utile per la visualizzazione dello stesso all'interno della piattaforma).

4.1.5 Traduzione verso il modello relazionale
Nella fase di traduzione dal modello logico a quello relazionale vengono apportate delle modifiche al modello logico rispetto alla tipologia di associazione.

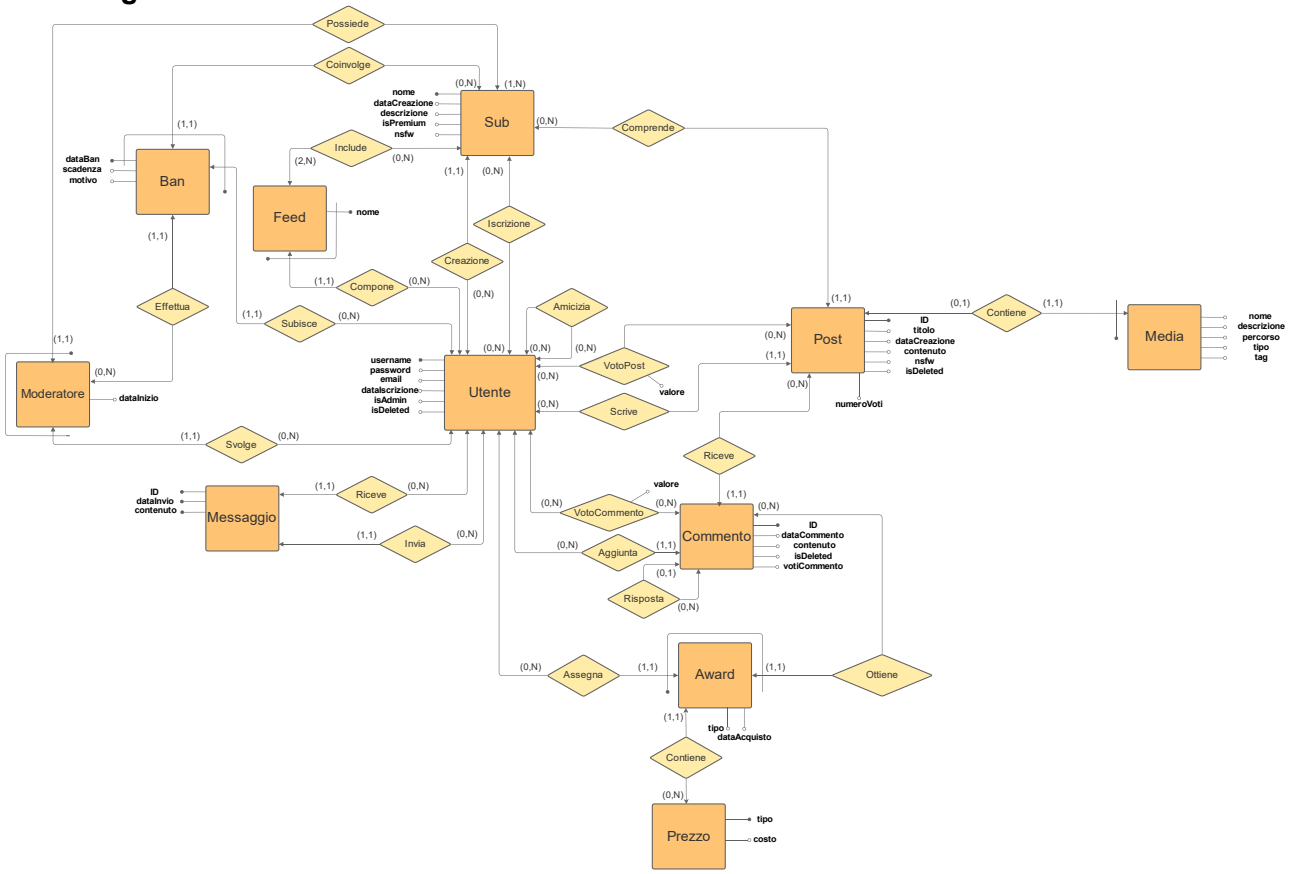
Le relazioni *Amicizia*, *Iscrizione*, *Include*, *VotoPost* e *VotoCommento* costituiscono associazioni **molti a molti;** la loro traduzione nel modello relazionale prevede che, per ogni associazione, si definisca una relazione che abbia lo stesso nome, gli stessi attributi dell'associazione e gli identificatori presi da quelli delle entità coinvolte (che formeranno la chiave primaria della relazione).

La relazione *Contiene* rappresenta un'associazione **uno a uno**. La sua traduzione nel modello relazionale può essere effettuata in più modi. Il caso in esame presenta però la necessità di definire un identificatore per l'entità Media; perciò, vista la presenza di un identificatore esterno, l'unica opzione è quella di indentificarlo tramite la chiave dell'entità Post.

Anche nel caso delle entità *Moderatore*, *Ban*, *Feed* e *Award* notiamo la presenza dell'**identificatore esterno**. Come per il caso precedente, l'unica opzione valida di traduzione è quella di determinare l'identificatore esterno per mezzo della chiave delle entità a cui sono associate.

Per tutte le restanti relazioni con associazione **uno a molti** si segue quanto previsto dalle regole di traduzione, pertanto l'entità con cardinalità 1 dovrà contenere, come chiave esterna, la chiave dell'entità con associazione (0,N).

4.1.6 Diagramma ER ristrutturato



4.2 Schema relazionale

Segue la descrizione dello schema relazionale, con gli eventuali vincoli di integrità referenziale.
*Gli attributi segnati con * ammettono valore NULL.*

Utente(<u>username</u>, password, email, dataIscrizione, isAdmin, isDeleted)
Amicizia(<u>utente1</u>, <u>utente2</u>)
Amicizia.utente1 → Utente.username
Amicizia.utente2 → Utente.username
Messaggio(<u>ID</u>, dataInvio, contenuto, mittente, destinatario)
Messaggio.mittente → Utente.username
Messaggio.destinatario → Utente.username
Sub(<u>nome</u>, dataCreazione, descrizione, isPremium, nsfw, creatore)
Sub.creatore → Utente.username
Iscrizione(<u>utente</u>, <u>sub</u>)
Iscrizione.utente → Utente.username
Iscrizione.sub → Sub.nome
Post(<u>ID</u>, titolo, dataCreazione, contenuto, nsfw, creatore, sub, numeroVoti, isDeleted)
Post.creatore → Utente.username
Post.sub → Sub.nome
VotoPost(<u>utente</u>, <u>post</u>, valore)
VotoPost.utente → Utente.username
VotoPost.post → Post.ID
Commento(<u>ID</u>, dataCommento, contenuto, isRisposta*, votoCommento, utente, post, isDeleted)
Commento.isRisposta → Commento.ID
Commento.utente → Utente.username
Commento.post → Post.ID
VotoCommento(<u>utente</u>, <u>commento</u>, valore)
VotoCommento.utente → Utente.username
VotoCommento.commento → Commento.ID
Media(<u>ID</u>, nome*, descrizione*, percorso, tipo, tag*)
Media.ID → Post.ID

Prezzo(tipo, costo)
Award(utente, commento, tipo, dataAcquisto)
Award.utente → Utente.username
Award.commento → Commento.ID
Award.tipo → Prezzo.tipo
Feed(utente, nome)
Feed.utente → Utente.username
Include(feed, utente, sub)
Include.feed → Feed.nome
Include.utente → Feed.utente
Include.sub → Sub.nome
Moderatore(utente, sub, dataInizio)
Moderatore.utente → Utente.username
Moderatore.sub → Sub.nome
Ban(sub, utente, dataBan, moderatore, modSub, scadenza*, motivo)
Ban.sub → Sub.nome
Ban.utente → Utente.username
Ban.moderatore → Moderatore.utente
Ban.modSub → Moderatore.sub

5. Implementazione query ed indici

5.1 Query

Query 1: *selezionare per ciascun utente, in ordine alfabetico, i feed da lui creati ed i sub che lo compongono*

```
select username as "Proprietario", f.nome as "Nome feed", i.sub as "Sub incluso"
from utente u join feed f on u.username=f.utente join include i on
(f.nome=i.feed and f.utente=i.utente)
group by username, f.nome, i.sub
order by username, f.nome;
```

Proprietario character varying (20)	Nome feed character varying (30)	Sub incluso character varying (20)
asawerse	videogames	gaming
asawerse	videogames	PlayStation
kharrow0	subs	gaming
kharrow0	subs	italy
kharrow0	subs	pictures
kharrow0	subs	PlayStation

Query 2: *visualizzare per ogni sub il numero di post e di commenti effettuati al suo interno*

```
select s.nome as "Sub",
(select count(*) from post p where p.sub=s.nome group by s.nome) as "Numero
post",
(select count(*) from commento c, post p where c.post=p.id and p.sub=s.nome
group by s.nome) as "Numero commenti" from sub s;
```

Sub character varying (20)	Numero post bigint	Numero commenti bigint
gold	[null]	[null]
italy	2	3
gaming	2	5
pictures	1	2
PlayStation	[null]	[null]

Query 3: *query che calcoli il punteggio "karma" di tutti gli utenti, ovvero la somma dei punteggi ottenuti dai rispettivi post e commenti*

```
drop view if exists votiPostUtente;
create view votiPostUtente as select username, sum(valore) as votiPost from
utente u join votoPost v on u.username=v.utente group by username;

drop view if exists votiCommentoUtente;
create view votiCommentoUtente as select username, sum(valore) as votiCommento
from utente u join votoCommento v on u.username=v.utente group by username;

select u.username, coalesce(votiPost+votiCommento, 0) as karma from utente u
left join votiPostUtente vp on u.username=vp.username
left join votiCommentoUtente vc on u.username=vc.username group by u.username,
votiPost, votiCommento order by karma desc;
```

username [PK] character varying (20)	karma bigint
vferrierif	3
kharrow0	2
mslineyc	0
rblackhallh	0

Query 4: *selezionare per ogni sub gli utenti che hanno scritto un post il cui voto è maggiore della media dei post all'interno di quel sub*

```
drop view if exists mediaVotiSub;
create view mediaVotiSub as select nome, cast(avg(numeroVoti)as decimal(100,2))
as media from sub s join post p on s.nome=p.sub group by s.nome;

select s.nome, (select media from mediaVotiSub m where m.nome=s.nome), username,
p.numeroVoti as voto
from utente u join post p on u.username=p.creatore join sub s on p.sub=s.nome
where p.numeroVoti>(select media from mediaVotiSub m where m.nome=s.nome)
order by (s.nome, p.numeroVoti;
```

nome character varying (20)	media numeric (100,2)	username character varying (20)	voto integer
gaming	0.00	vferrierif	3
italy	2.00	kharrow0	3

Query 5: *produrre una lista ordinata per sub dei primi 3 moderatori che hanno effettuato più ban*

```
drop view if exists numeroBanPerModeratore;
create view numeroBanPerModeratore as
select m.sub, m.utente, count(*) as numeroBan from moderatore m join ban b on
(m.utente=b.moderatore and m.sub=b.modSub)
group by m.sub, m.utente;

select s.nome, m.utente as moderatore, numeroBan
from sub s join moderatore m on s.nome=m.sub join numeroBanPerModeratore n on
(n.utente=m.utente and n.sub=s.nome)
order by s.nome, m.utente;
```

nome character varying (20)	moderatore character varying (20)	numeroban bigint
gaming	kharrow0	3
italy	gbleything7	1
pictures	kharrow0	1

Query 6: *cercare i media che contengono tag specificati dall'utente e, se presenti, mostrarne il tipo, il titolo del relativo post, l'utente che l'ha creato, i voti ricevuti, il numero di commenti*

```
drop view if exists numeroCommentiPost;
create view numeroCommentiPost as
select p.id, count(*) as "commenti"
from post p join commento c on p.id=c.post
group by p.id;

select m.tipo, titolo, creatore, numeroVoti as "voti", commenti
from numeroCommentiPost n join post p
on n.id=p.id join media m on p.id=m.id
where tag like '%funny%'; (si assume il termine "funny" come input dell'utente)
```

tipo character varying (5)	titolo character varying (100)	creatore character varying (20)	voti integer	commenti bigint
video	Primo post	kharrow0	3	3
foto	Funny dog pic	ptolotti1	5	2

5.2 Indici

5.2.1 Indice per i ban

Ogni volta che un utente accede ad un sub è necessario verificare che egli non sia stato bannato, per garantirgli la possibilità di creare post e commentarli. L'operazione di ban è però piuttosto rara, pertanto l'utilizzo di un indice nella tabella Ban può rendere più efficiente l'operazione precedentemente descritta.

```
drop index if exists indiceBan;
create index indiceBan on Ban(utente);
```

5.2.1 Indice per utenti premium

Un utente premium è un utente che ha acquistato almeno un award. La piattaforma contiene dei sub a cui può accedere solamente questa categoria di utenti, perciò la loro identificazione deve essere quanto più possibile efficiente. Si decide quindi di creare un indice di tipo hash all'interno della tabella Award, poiché si desidera accedere ad una tupla contenente uno specifico valore della chiave.

```
drop index if exists indicePerUtentiPremium;
create index indicePerUtentiPremium on Award(utente);
```

6. Implementazione in C++

Per quanto riguarda l'implementazione delle query in C++, si è deciso di scrivere un unico sorgente che esegue le diverse query in successione.
Per essere compilato, il sorgente richiede la presenza di una cartella "dependencies", con al suo interno le cartelle "lib", contenente i file "libpq.dll" e "libpq.lib", e "include" contenente i file "libpq-fe.h", "pg_config_ext.h" e "postgres_ext.h".
Una volta compilato, il programma può essere eseguito da riga di comando specificando, in ordine, username, password, nome del database, indirizzo del database, numero di porta.

```
C:\Users\Utente\Progetto>query.exe postgres password leddit 127.0.0.1 5432
```