Progetto database (da abbellire ovviamente)

Indice dei contenuti

Requisiti di progetto e descrizione dell’ambito applicativo

Si intende realizzare un database a supporto di un social network. Per definire il contesto su cui si va a realizzare la base di dati, tale social network è chiamato “Sententia”, che significa opinione dal latino, il cui scopo è quello di dare la possibilità a chiunque di esprimere la propria opinione su qualunque argomento essi desiderino dandone una recensione. L’idea principale che lo rende unico a tutti gli effetti è quello di non imporre limiti di alcun tipo e organizzare le recensioni in modo tale che siano trovabili e consultabili efficientemente.

Il fulcro del sistema si colloca indubbiamente nell’utente, in quanto essenzialmente tutto ruota attorno alla figura dello stesso, a ciò che decide pubblicare, ai messaggi che invia e riceve, e le sue transazioni. I loro dati sono: Email e Username che sono entrambi univoci, nome, cognome, password, e foto profilo. Una prima funzionalità riguarda la richiesta di amicizia tra utenti. L’amicizia non è necessariamente reciproca, ovvero un utente A può avere B come amico ma non necessariamente è vero per B rispetto ad A. Un’amicizia può essere annullata, ma non è necessario tenere traccia dell’intera catena di amicizie/annullamenti e neanche la data in cui è stata effettuata l’annullazione.

Le recensioni sono rappresentate nel seguente modo: esse riguardano un oggetto, che a sua volta appartiene ad una categoria, e contengono informazioni quali un codice identificativo, una foto, un titolo e un valore. Esistono due tipologie di interazioni tra utenti: quelle pubbliche, che consistono nel commentare una determinata recensione, e quelle private, gestite da un sistema di messaggistica che consente l’invio e ricezione di messaggi testuali, immagini e persino altre recensioni. I commenti riguardanti una determinata recensione sono caratterizzati da un codice identificativo in base alla recensione (per facilità di visualizzazione e ricerca dei vari commenti) e dal testo dello stesso. Si richiede inoltre la possibilità di rispondere a commenti. Un’altra forma di espressione della propria opinione è sotto forma della medaglia, che un utente può conferire (solamente una volta) ad una determinata recensione. Per quanto riguarda invece i messaggi fra utenti, in generale si necessita di un codice identificativo in base alla coppia di utenti coinvolti, inoltre occorre conoscere se siano stati letti o meno, e nel caso in cui essi siano testuali si richiede solamente il testo del messaggio stesso, in caso contrario contengono una descrizione opzionale, affiancata da un’immagine oppure una recensione.

Le pubblicazioni tra utenti necessitano di un controllo per conservare la sicurezza della piattaforma, occorre dunque introdurre una figura il cui compito è quello di scorrere la lista dei contenuti della piattaforma ed eventualmente rimuoverli se si reputa che violano le regolazioni imposte. Tale figura prende il nome di moderatore, e i dati che vengono memorizzati sono gli stessi dell’utente “utilizzatore” a meno della foto profilo. Un moderatore può fare varie cose: può effettuare un “ban” su uno o più utenti che possono essere annullati successivamente se necessario, e può “nascondere” una recensione temporaneamente. Per il fatto che le recensioni vengono considerate con più cautela, dato che può essere una fonte di monetizzazione e può richiedere svariate ore di lavoro, mentre per i commenti ci si aspetta che non siano particolarmente importanti per la piattaforma, motivo per il quale la “cancellazione” di una recensione è in molti casi soltanto temporanea, dando la possibilità al creatore di contestare la rimozione con una motivazione valida. Invece i commenti possono essere rimossi direttamente dalla base di dati, e non è necessario neanche memorizzarsi la data o il moderatore che l’ha fatto. In base al numero di violazioni che ha effettuato un determinato utente, un moderatore può decidere di “bandire” un utente, e tale operazione deve essere memorizzata assieme al moderatore e all’utente coinvolti. È necessario che le informazioni che riguardano un utente bandito restino nel database, compreso il suo indirizzo IP, al fine di individuazione dello stesso nel caso tentasse di creare un nuovo account.

Essendo il progetto una base di dati di supporto ad un social network, è importante monitorare il tempo esatto di ciascuna pubblicazione. Questo include i messaggi, i commenti e le recensioni, ma anche il ban di una recensione, le varie transazioni, le amicizie tra utenti, ecc. Questo processo di controllo dei contenuti viene affidato in parte anche agli utenti utilizzatori, in quanto si permette a questi ultimi di segnalare una o più recensioni. La segnalazione può avere una tipologia tra quelle predefinite, oppure l’utente può scrivere liberamente la sua motivazione se questa non è presente tra quelle selezionabili. Una segnalazione da parte di un utente riferita ad una particolare recensione può essere effettuata soltanto una volta.

Un altro aspetto molto importante è il discorso sulla monetizzazione. Vi sono due tipologie di utenti: gli utenti normali e gli utenti premium. Gli utenti normali sono quelli descritti fino ad ora, mentre quelli premium hanno tutte le caratteristiche degli utenti normali con l’aggiunta del fatto che possono ricevere donazioni monetarie da utenti (sia da utenti normali sia premium).

Una peculiarità degli utenti premium è che hanno la possibilità di definire piani di abbonamento, ciascuno con un determinato periodo e quantità di pagamento (per esempio: 2€ al mese), ai quali una molteplicità di utenti possono iscriversi. Dell’iscrizione occorre memorizzare la data e l’eventuale data di abbandono. Inoltre, gli utenti premium possono anche impostare incentivi per l’iscrizione a determinati piani per mezzo dell’esclusività temporanea di una recensione. Per ciascun piano e ciascuna recensione pubblicata dallo stesso utente si può avere un’esclusività definita come una data a partire dalla quale la visione esclusiva è permessa (solo per gli iscritti a quel piano). Una stessa recensione può fare parte di più esclusività diverse (soltanto per piani diversi, non all’interno dello stesso piano) ciascuna con una data di visualizzazione anticipata differente. Allo stesso tempo un piano può promettere l’esclusività di più recensioni con date diverse. Vi è inoltre una proprietà della recensione che ne indica la data in cui è visibile a tutti gli utenti.

Per questo progetto occorre tenere traccia di tutte le transazioni effettuate dagli utenti, in modo tale che essi stessi possano, sotto determinate condizioni, annullarla, e occorre anche tenere traccia del fatto che una transazione sia già stata annullata o meno. Le transazioni possono essere di due tipologie: le transazioni manuali e quelle automatiche. Quelle manuali possono essere effettuate liberamente da un utente qualsiasi verso un utente premium, con una quantità di denaro a libera scelta, mentre quelle automatiche sono quelle effettuate per effetto dell’iscrizione ad un piano. Esse sono identificate da un CRO, un codice utilizzato universalmente per identificare una transazione, e devono contenere il numero della carta di credito dell’utente donatore e l’identificativo dell’utente premium destinatario. Un utente può avere più carte di credito (anche nessuna) e una carta di credito può essere utilizzata da più utenti. È richiesto di imporre il vincolo che in una certa data, un determinato utente può soltanto effettuare una donazione rivolta ad un determinato utente premium, e questo va garantito sia per le transazioni manuali sia per quelle automatiche, con l’unica differenza che per quanto riguarda quelle automatiche si permette che un utente ne effettui più di una nei confronti di uno stesso utente premium, ma soltanto in piani diversi. In altre parole, si impedisce ad un utente di effettuare più di una transazione automatica riferita ad un unico piano.

Senza addentrarsi troppo nel dettaglio, in quanto si andrebbe fuori dallo scopo del progetto in questione, i pagamenti vengono svolti nella modalità seguente. L’azienda sviluppatrice del social network gestisce entrate e uscite (transazioni dall’utente donatore all’azienda e quelle dall’azienda all’utente destinatario), in modo tale che essa possa incassare una certa percentuale. Per le entrate si utilizza la carta di credito, mentre per quanto riguarda le uscite vengono gestite con un bonifico tramite IBAN. L’IBAN è un’informazione necessaria solo per gli utenti premium, mentre non la si memorizza per gli utenti non-premium, ed entrambi possono fornire zero o più carte di credito (e ne devono avere almeno una se hanno intenzione di effettuare transazioni).

Essendo il progetto in questione un social network, esso necessita di continui cambiamenti e aggiunte, non solo per scopi di manutenzione ma anche per il fine di ritenere l’attenzione del pubblico, quindi è richiesta la possibilità di poter agilmente aggiungere funzionalità.

Il social network deve poter essere robusto ad eventuali sistemi di sovraccarico delle risorse, per esempio l’invio automatizzato di un numero elevato di commenti o recensioni (anche chiamato spam). Questo ha anche lo scopo di migliorare l’esperienza utente. Si richiede che un utente possa effettuare al massimo 2000 pubblicazioni (recensioni o commenti) in un’ora.

È necessario avere a disposizioni determinati dati per soddisfare alcune funzionalità del social network, per esempio il numero di medaglie e di recensione di un utente che è visualizzabile da un menù, e le varie informazioni che riguardano il guadagno degli utenti premium. In particolare, è assolutamente indispensabile poter reperire in maniera efficiente i seguenti dati: il totale incasso di un utente premium, il numero di transazioni effettuate da un particolare utente, il numero di medaglie ricevute, il numero di recensioni effettuato, e il numero di risposte ad un determinato commento. [Ricordarsi poi nella parte sui dati derivati di parlare anche della visibilità della recensione, compresa la visibilità della recensione] Il numero di consultazioni di ciascun dato verrà riportato successivamente nella sezione riguardante i dati derivati.

Progetto concettuale – Schema E/R

I requisiti del progetto rilevano come fulcro il legame tra gli utenti e le recensioni, che viene rappresentata nel seguente schema scheletro (estremamente semplificato e privo della maggior parte delle funzionalità richieste):

[ Immagine schema scheletro ]

È possibile ampliare questo schema con l’obiettivo di implementare ciascuna richiesta. Innanzitutto, i commenti possono rispondere ad altri commenti, viene dunque introdotta un’auto-associazione uno a molti. Non viene gestito, in questa prima fase di progettazione concettuale, il problema che un commento può essere in risposta ad un altro commento pur non riferendosi alla stessa recensione. Questo verrà in effetti gestito nella fase appena successiva, ovvero di progettazione logica.

[ Immagine commento ]

Per ragionare sul problema risulta efficace scomporlo in più parti:

1. La messaggistica e le amicizie tra utenti
2. I moderatori e il controllo dei contenuti
3. Gli utenti premium e le transazioni

Si può notare che queste tre componenti sono strettamente correlate agli utenti e alle recensioni. Per la natura estremamente dinamica della realtà che si va a rappresentare, si pone come obiettivo la semplicità del diagramma ER, lasciando la gestione dei vincoli più complessi ai trigger, discussi in una parte successiva di questo progetto.

La messaggistica tra utenti è gestita attraverso un’entità Messaggio legata all’entità Utente in due relazioni uno a molti (dove il lato molti è il Messaggio).

[ Immagine messaggio ]

Il messaggio contiene un codice che identifica lo stesso a partire dalla coppia di utenti “mittente-destinatario”, e le tipologie differenti vengono modellate con l’uso di una gerarchia, come si può vedere dalla seguente figura.

[ Immagine gerarchia messaggio ]

Si tratta di una gerarchia totale ed esclusiva perché un messaggio può essere solamente uno e uno solo di quei 3 tipi. Ci si rende conto del fatto che la descrizione opzionale è comune alle due entità MImmagine e MRecensione, si potrebbe dunque riscrivere come un’altra gerarchia che ha come entità padre “MNonTesto” o similmente denominata, tuttavia per questioni di semplicità di rappresentazione si sceglie di evitare di modificare il diagramma in questo modo. In ogni caso le decisioni di traduzione in uno schema logico verranno intraprese nella fase successiva.

Le amicizie vengono modellate attraverso un’auto-associazione molti a molti.

I moderatori e gli utenti normali condividono una grande porzione di dati, è possibile introdurre anche in questo caso una gerarchia totale ed esclusiva, dove l’entità padre viene chiamata “Persona”.

[ Immagine gerarchia persona ]

Un moderatore può bandire un determinato utente, e questo si traduce come una semplice associazione uno a molti.

Il moderatore può “disattivare”, ovvero rendere invisibile temporaneamente, le recensioni. Per progettare questo, si può ricorrere all’uso di un’entità “rimozione” che ha come chiave composta la data di effettuazione e il riferimento alla recensione, e contiene un’eventuale data di annullamento. In questo modo una rimozione può essere effettuata più volte in date diverse, ma questo deve essere permesso soltanto se vi sia una data di annullamento non nulla (che indica che sia effettivamente stata annullata), un vincolo che verrà gestito nelle fasi successive. Un utente può effettuare un report, il quale è identificato dall’utente e dalla recensione che sono coinvolti. In questo modo una recensione può essere disattivata e riattivata più volte.

Di seguito è riportato il diagramma E/R aggiornato.

[ Immagine ER aggiornato ]

Per quanto riguarda il sistema di monetizzazione di questo progetto, innanzitutto si può notare che gli utenti premium non sono altro che utente con alcune caratteristiche aggiuntive, in altre parole una specializzazione di utente. Questo si rappresenta all’interno dello schema ER sotto forma di una gerarchia non totale ed esclusiva come segue.

[ Immagine gerarchia Utente ]

La gerarchia è parziale perché un utente può non essere premium (gli utenti normali), e una discussione sull’esclusività in una gerarchia con una sola entità figlia è per certi aspetti irrilevante.

Il piano è in un’associazione uno a molti con l’entità UPremium, dove la partecipazione di UPremium è facoltativa, e l’esclusività di una recensione viene gestita con una semplice associazione molti a molti tra Piano e Recensione contenente un attributo DAnticipata, ovvero la data oltre la quale gli utenti iscritti a quel piano sono abilitati alla visualizzazione della recensione. In questa fase non viene gestito il fatto che un piano può abilitare l’esclusività della recensione pubblicata da un altro utente, persino non premium.

Le iscrizioni vengono gestite con altrettanta semplicità, con un’entità Iscrizione che è identificata dall’utente che la effettua, il piano, e la data. Essa contiene un’eventuale data di abbandono. Similmente al ragionamento fatto sull’entità Rimozione, occorre assicurarsi successivamente che non sia possibile iscriversi due volte ad uno stesso piano senza averlo prima abbandonato. Dato che bisogna memorizzare alcune caratteristiche sulla carta di credito associata ad un particolare utente, si crea un’entità apposita “CartaCredito” che si trova in un’associazione molti a molti con l’entità Utente. Essa è identificata dal numero e contiene informazioni quali la data di scadenza e l’ente emittente. Lo schema E/R fin’ora considerato è il seguente:

[ Immagine E/R aggiornato ]

Le due tipologie di transazione sono per certi aspetti differenti, in quanto la transazione manuale deve contenere la somma di denaro donata, la data, l’utente premium destinatario della donazione, la carta di credito utilizzata e l’utente donatore, mentre la transazione automatica si riferisce ad un piano che contiene già l’utente premium e la somma di denaro. Può essere ragionevole pensare che un piano possa essere modificato nel tempo da un utente premium e persino cancellato. Si può notare, tuttavia, una problematica legata a questa casistica; se di una transazione automatica si tiene solamente traccia del piano che l’ha attivata, allora nel momento in cui quest’ultimo viene eliminato i dati riguardanti la somma pagata e l’utente premium verranno totalmente persi, rendendo conseguentemente lo storico di transazioni inutilizzabile. A questo punto si possono intraprendere due strade, di cui si espongono i rispettivi lati positivi e negativi. Come prima cosa, è possibile salvare all’interno della transazione automatica anche i dati della transazione in aggiunta del piano che l’ha attivata. Questo comporta una certa ridondanza dei dati che implica un piccolo spreco di memoria e i costi associati al mantenimento della consistenza attraverso i trigger, tuttavia si può liberamente modificare la relazione dei piani senza avere alcuna ripercussione significativa sullo storico. D’altra parte, si può impedire la modifica permanente di un piano, sostituendola con l’aggiunta di un nuovo piano con i cambiamenti apportati. Adottando questa strategia si memorizzano quindi tutti i piani che sono mai esistiti. Dato che in questo caso si vuole memorizzare uno storico, e i dati riguardanti i piani passati possono rivelarsi utili, si opta per la seconda scelta.

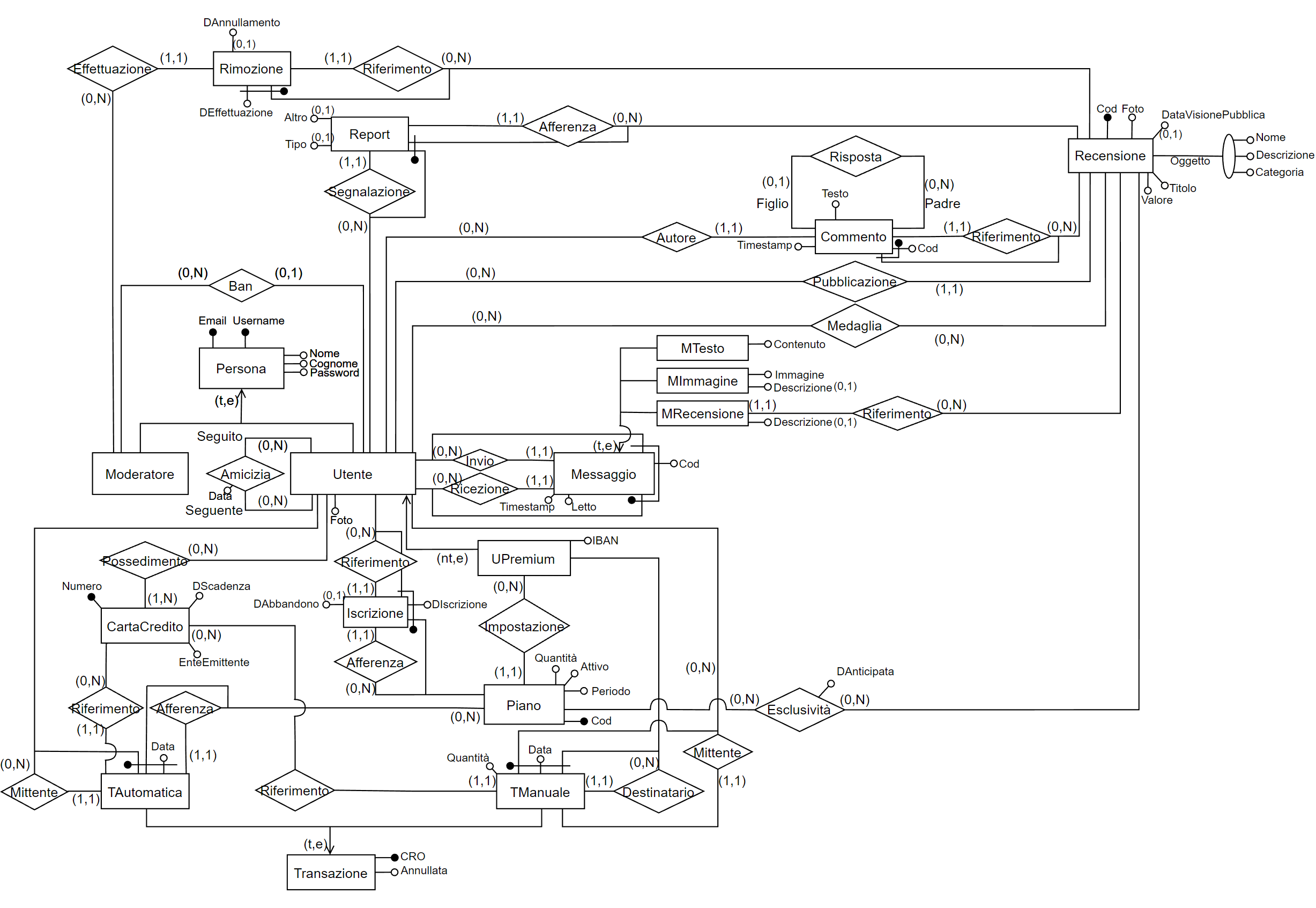
Da qua la progettazione si dirama ancora una volta, in quanto vi sono più modi per gestire la dicotomia tra piani attivi e inattivi. Una possibilità è quella di tenere due entità separate, e al posto di eliminare o modificare i dati già esistenti si crea una copia e la si inserisce nell’entità aggiuntiva “PianoInattivo”. Questo presenta il vantaggio considerevole della semplicità implementativa in certe condizioni, in quanto basta operare sull’entità “PianoAttivo” (o semplicemente “Piano” senza dover controllare ogni volta un attributo booleano e in generale impattare il funzionamento ordinario del database, il che potrebbe essere soggetto a errori. Tuttavia, a differenza della scelta di una semplice aggiunta di un attributo booleano, le relazioni presenti tra un piano (che sia attivo oppure no) e le altre entità potrebbero essere difficili da gestire. Come si vedrà nelle fasi successive, si opta per la presenza di un attributo booleano con l’obiettivo di ovviare il problema della complicazione delle interrogazioni facendo uso di una vista.

In seguito a queste riflessioni, il diagramma E/R risultante è il seguente:

[ Immagine E/R aggiornato ]

Dato che nella transazione viene memorizzata sia la carta di credito sia l’utente donatore, come conseguenza negativa con questo modello E/R è possibile immagazzinare una transazione da parte di un utente che non è in possesso della carta a cui si riferisce la transazione stessa, si tratta di un altro vincolo che verrà gestito in seguito tramite i trigger.

Si presenta dunque lo schema finale:



Progetto logico – Schema relazionale

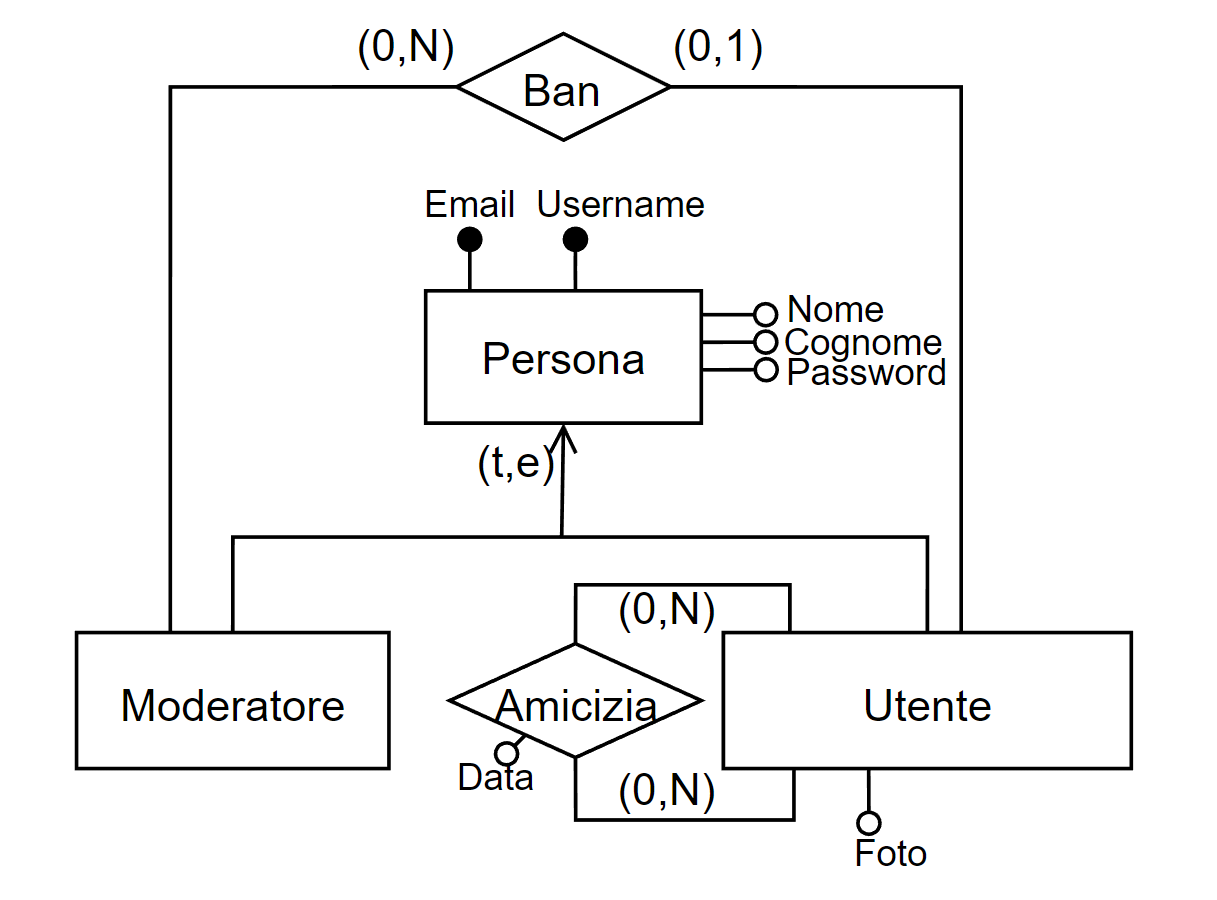
La progettazione logica permette di definire alcuni dettagli implementativi per poter operare nella pratica sui concetti stabiliti durante la fase di progetto concettuale, e viene decomposta nelle seguenti fasi:

* Eliminazione delle gerarchie ISA
* Selezione delle chiavi primarie, eliminazione delle identificazioni esterne
* Trasformazione degli attributi composti o multipli
* Traduzione di entità e associazioni in schemi di relazioni
* Verifica di normalizzazione

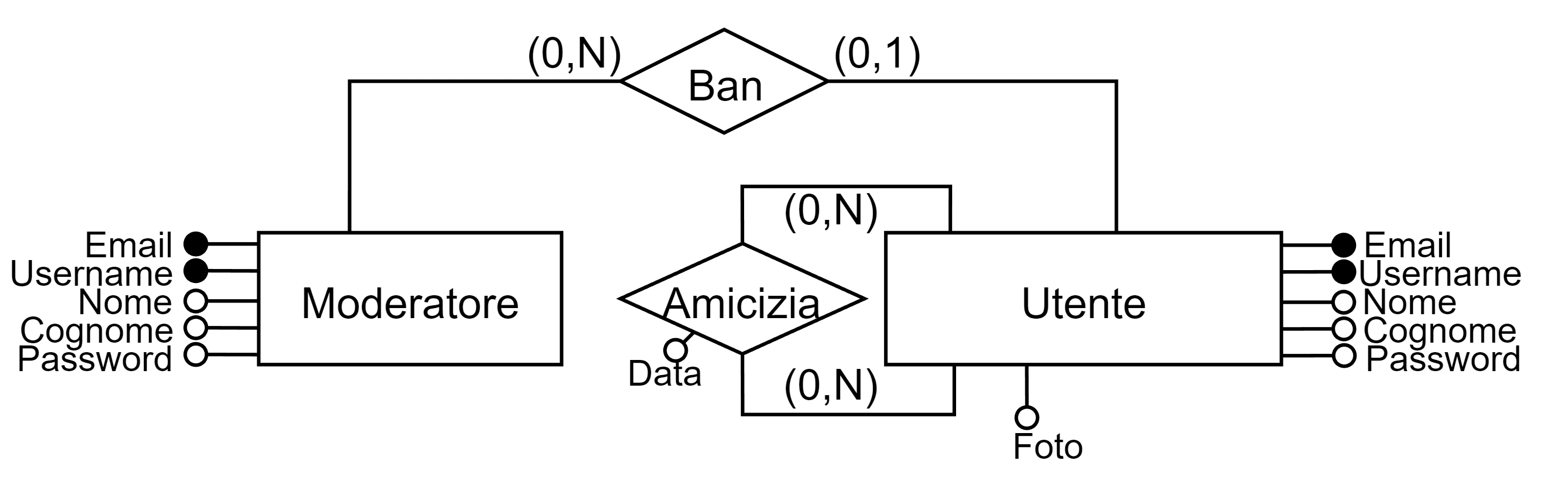
Si procede nell’ordine descritto.

Eliminazione delle gerarchia ISA

Nello schema E/R sono presenti 4 gerarchie ISA, si propone di focalizzare l’attenzione sulla seguente gerarchia per prima:

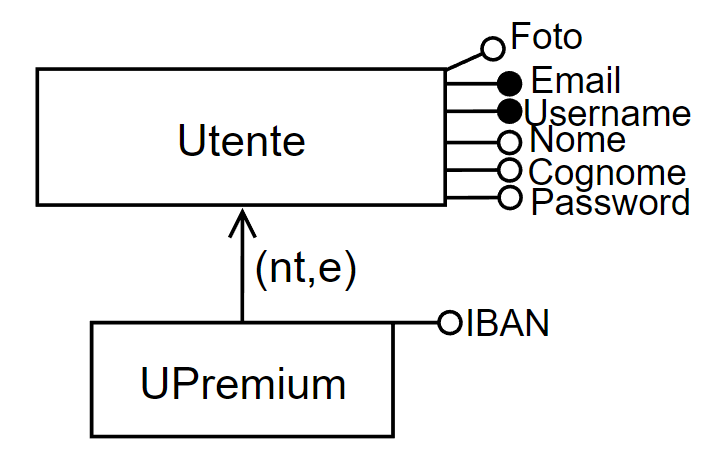


Nell’immagine sono evidenziate alcune associazioni la cui presenza influisce sulla decisione. Per effettuare una decisione appropriata sull’eliminazione, occorre tenere in considerazione che vi è un’ulteriore gerarchia sottostante, con Utente come entità padre. Proprio per questo motivo, e per il fatto che l’entità padre non si interfaccia con alcuna entità nella base di dati, si ritiene che la scelta migliore in questo caso sia un collasso verso il basso, come segue:

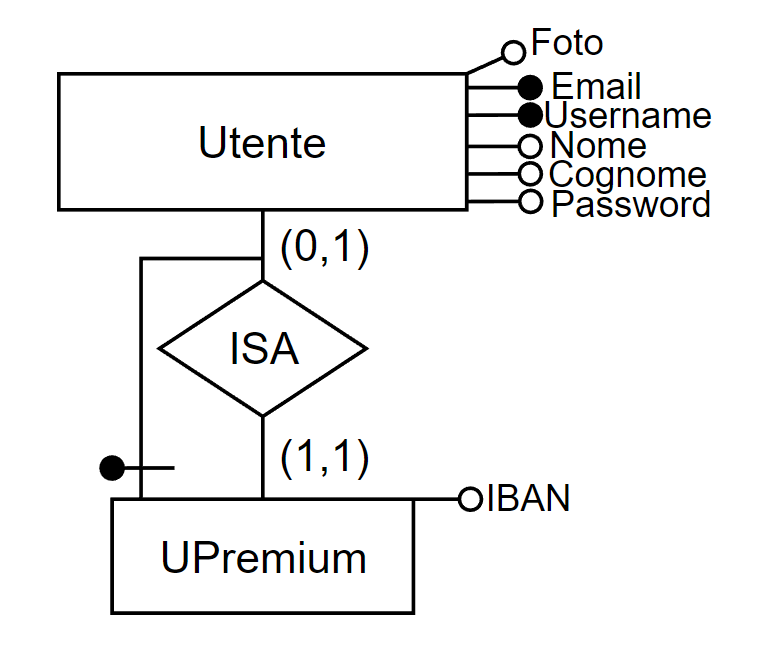


Questo richiederà, a livello implementativo, la creazione di alcuni trigger che si occuperanno di garantire che i valori di “Email” e “Username” siano univoci all’interno di entrambe le relazioni, dato che la gerarchia è esclusiva.

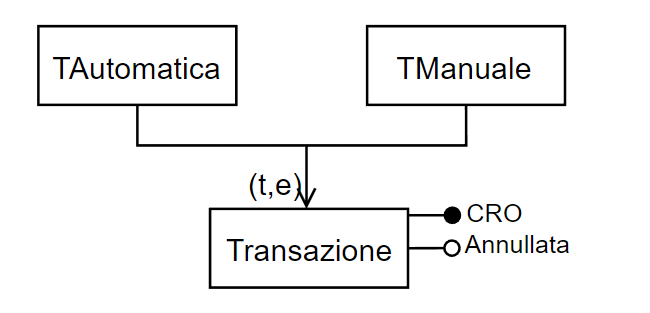
Si prende ora in considerazione la gerarchia immediatamente sottostante (nell’immagine è già stata applicata l’eliminazione della gerarchia già considerata):



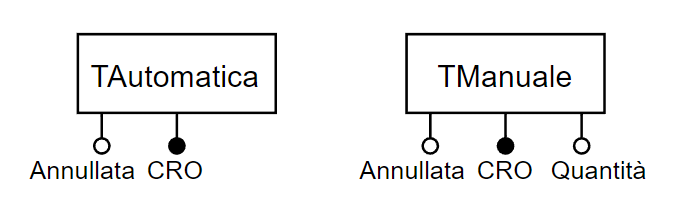
Vi sono numerose interazioni/associazioni che legano sia l’entità padre che l’entità figlia a tutte le altre, inoltre si può notare che vi sono numerose associazioni che legano le due entità Utente e UPremium, per esempio tutti gli utenti possono donare ma solamente gli utenti premium sono in grado di ricevere denaro. Si ritiene quindi che un mantenimento delle entità sia benefico in questo caso, dato che un collasso verso il basso non è possibile a causa della parzialità della gerarchia, e un collasso verso l’alto comporterebbe numerose complicazioni per la gestione delle varie associazioni. La sua traduzione è quindi la seguente:



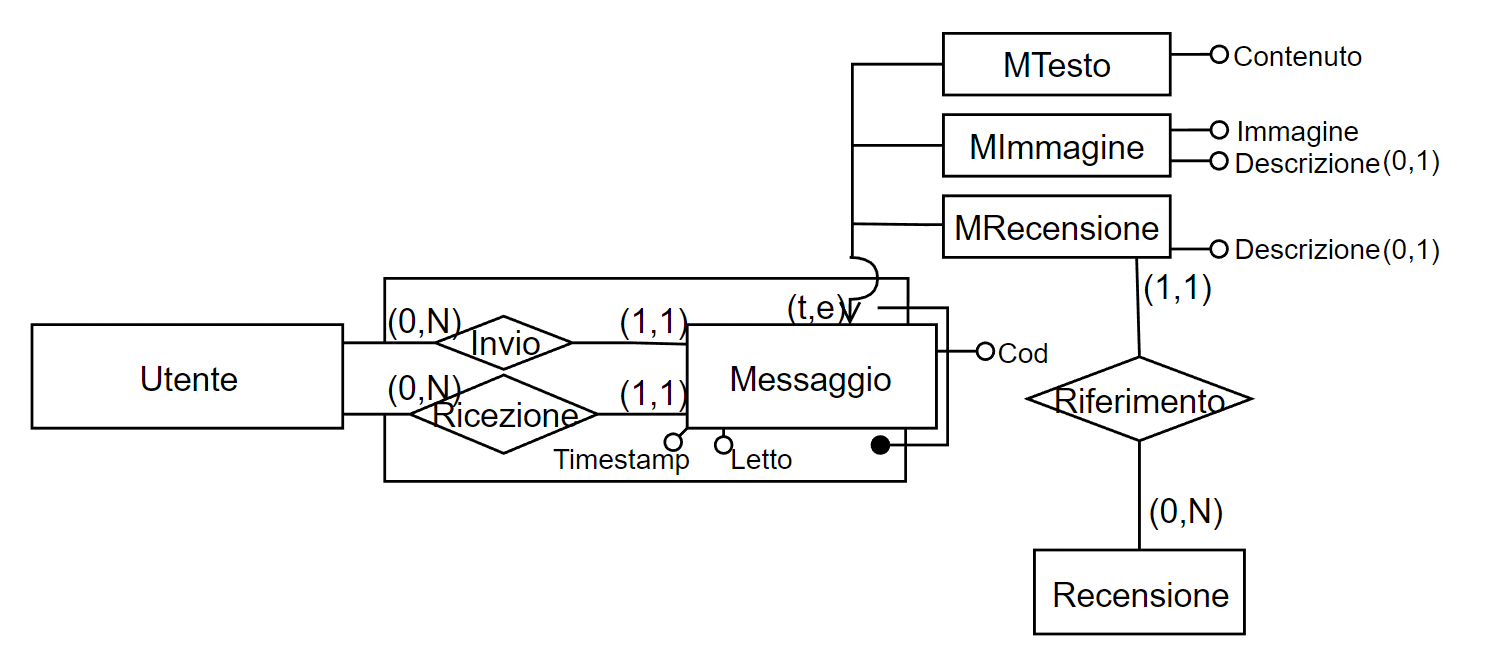
Per quanto riguarda la gerarchia relativa alle transazioni:



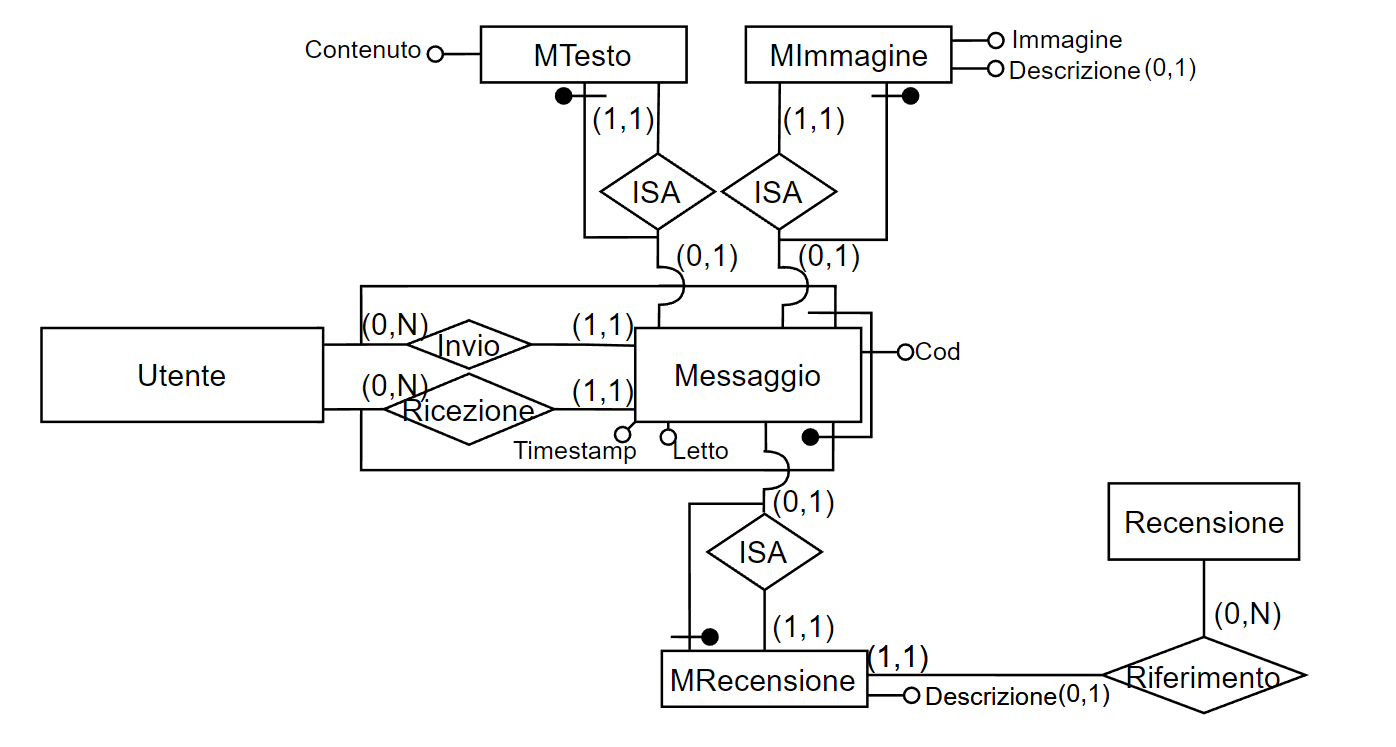
Essa può essere tradotta in più modi, tuttavia per il fatto che l’entità padre contiene solamente due attributi e nessuna associazione che la lega ad altre entità, un semplice collasso verso il basso è sufficiente.



Infine occorre gestire l’eliminazione della gerarchia di Messaggio. Si tratta di una gerarchia totale ed esclusiva, rappresentata nell’immagine seguente:



Innanzitutto occorre specificare che ci si aspetta che la maggior parte dei messaggi siano testuali, perciò non converrebbe effettuare un collasso verso l’alto in quanto si avrebbe un numero di NULL eccessivamente elevato. A questo punto le due alternative sono pressochè interscambiabili, un collasso verso il basso implica il dover realizzare un trigger che garantisce che venga rispettata l’unicità del codice data una coppia mittente-destinatario di utenti, mentre con il mantenimento delle entità occorrerebbe crearne uno per assicurarsi che ad un Messaggio è collegato uno e soltanto uno tra MTesto, MImmagine, e MRecensione (per la totalità della gerarchia). In questo specifico caso, in una chat può essere efficace poter ottenere uno storico dei messaggi in maniera efficiente e veloce senza entrare nel dettaglio e visualizzarne il contenuto. Questa separazione tra gli attributi generali del Messaggio, ovvero “Timestamp” e “Letto”, e i contenuti, può essere desiderata oppure no, a seconda dell’applicazione. È stato scelto un mantenimento delle entità, da cui si ottiene:



Selezione delle chiavi primarie, eliminazione delle identificazioni esterne

Vi è una scelta di selezione delle chiavi primarie solamente in quattro casi: Utente, Moderatore, TManuale, TAutomatica. Negli ultimi due casi una delle due chiavi è composta ed esterna, quindi occorrerà prima eliminare quella esterna per poi scegliere tra le due. Tra lo username e l’email, è stata scelta come chiave primaria lo username per il semplice fatto che è generalmente più corto e che si suppone che verrà utilizzato più spesso per trovare un determinato utente.

UPremium ha una chiave esterna che viene tradotta in questo modo:

[ Immagine della trasformazione ]

L’entità rimozione possiede una chiave composta di cui una parte deriva dall’associazione Riferimento che la lega a recensione.

[ Immagine della trasformazione ]

Allo stesso modo, l’entità Iscrizione ha una chiave che è formata dall’associazione Riferimento, Afferenza, e dalla data nella quale l’utente ha effettuato una determinata iscrizione ad un piano.

[ Immagine della trasformazione ]

L’entità Messaggio è identificata da un codice che è solamente univoco tra i messaggi che hanno una determinata coppia di utenti come mittente-destinatario, che si traduce nel seguente modo:

[ Immagine della trasformazione ]

Le relazioni MTesto, MImmagine e MRecensioni vengono tutte trasformate in questo modo:

[ Immagine della trasformazione ]

Un Report è identificato dall’utente che lo ha effettuato e la recensione a cui si riferisce, per cui si ha:

[ Immagine della trasformazione ]

Un Commento è identificato da un codice univoco all’interno della recensione a cui si riferisce, e questo nella schema logico ha la seguente traduzione:

[ Immagine della trasformazione ]

TAutomatica e TManuale sono simili, la prima è identificata dalla data di effettuazione, dal piano a cui si riferisce e dal mittente, mentre la seconda dalla data, dal mittente e dal destinatario.

[ Immagine della trasformazione ]

Per TAutomatica e TManuale si compie la stessa decisione sulla scelta della chiave primaria, ovvero il CRO, lasciando le due chiavi composte come chiavi alternative. Questo per il semplice fatto che in generale si preferisce una chiave semplice piuttosto che una complessa.

Traduzioni di entità e associazioni in schema di relazioni

[ Spiegare bene tutto, questo occuperà circa 6/7 pagine secondo me ]

Si ritiene opportuno effettuare le analisi dei dati derivati per stabilire quali sono benefici da inserire e mantenere nella base di dati e quali non lo sono.

La prima discussione è svolta sulla visibilità delle recensioni, per essere più precisi è possibile stabilire se una recensione è attualmente nello stato “invisibile” andando a cercare nella relazione Rimozione, e se è presente una rimozione senza data di annullamento allora la recensione è invisibile. Questa procedura può essere semplificata nel caso in cui si utilizzasse un attributo “Visibile” nella recensione.

Tabella dei volumi:

|  |  |  |
| --- | --- | --- |
| Concetto | Tipo | Volume dati |
| Rimozione | E | 500 |
| Recensione | E | 10000 |

Tabella delle operazioni:

|  |  |  |
| --- | --- | --- |
| Operazione | Tipo | Frequenza |
| Rimozione Recensione | I | 10/G |
| Annullamento Rimozione | I | 1/G |
| Accesso Recensione | I | 1000000/G |

Può sembrare a prima vista che la presenza del dato derivato sia estremamente vantaggiosa; tuttavia, per un’analisi rigorosa è opportuno verificare concretamente.

Con l’utilizzo dei trigger è possibile introdurre un vincolo che non permetta un’altra rimozione di una recensione che sia già stata rimossa (cioè per la quale esista un record in “rimozione” che abbia data annullamento nulla).

Con l’utilizzo del dato derivato si ottengono i seguenti risultati:

Rimozione recensione:

|  |  |  |
| --- | --- | --- |
| Concetto | Accessi | Tipo |
| Rimozione | 1 | S |
| Riferimento | 1 | S |
| Recensione | 1 | S |

2\*3 \* 10 = 60/G

Annullamento rimozione:

|  |  |  |
| --- | --- | --- |
| Concetto | Accessi | Tipo |
| Rimozione | 1 | S |
| Riferimento | 1 | L |
| Recensione | 1 | S |

(2\*2 + 1) \* 1 = 5/G

Accesso recensione:

|  |  |  |
| --- | --- | --- |
| Concetto | Accessi | Tipo |
| Recensione | 1 | L |

1\* 1000000 = 1000000/G

Si ha quindi un totale di 1000065/G.

Per quanto riguarda il caso in cui non si utilizzi il dato derivato:

Rimozione recensione:

|  |  |  |
| --- | --- | --- |
| Concetto | Accessi | Tipo |
| Rimozione | 1 | S |
| Riferimento | 1 | S |

2\*2 \* 10 = 40/G

Annullamento rimozione:

|  |  |  |
| --- | --- | --- |
| Concetto | Accessi | Tipo |
| Rimozione | 1 | S |

2 \* 1 = 2/G

Accesso recensione:

|  |  |  |
| --- | --- | --- |
| Concetto | Accessi | Tipo |
| Riferimento | 500/10000 | L |
| Rimozione | 500/10000 | L |
| Recensione | 1 | L |

(1+500/10000+500/10000) \* 1000000 = 1100000/G

In realtà l’accesso a recensione si verifica soltanto nel caso in cui sia visibile, altrimenti l’accesso a tale recensione viene annullato. Ipotizzando che delle 10000 recensioni, 9500 siano visibili si ha in media:

(9500/10000+500/10000+500/10000) \* 1000000 = 1050000/G

Si ha quindi un totale di 1050042/G.

Dunque, come ipotizzato, il dato derivato conviene in questo caso, ma non era del tutto ovvio.

Lo schema logico complessivo è il seguente:

Moderatore(Email, Username, Nome, Cognome, Password)

PK: Username

AK: Email

Utente(Email, Username, Nome, Cognome, Password, Foto, IP)

PK: Username

AK: Email

UtentePremium(CodUtente, IBAN)

PK: CodUtente

FK: CodUtente REFERENCES Utente

Amicizia(Seguito, Seguente, Data)

PK: (Seguito, Seguente)

FK: Seguito REFERENCES Utente

FK: Seguente REFERENCES Utente

Categoria(CodC, Nome)

PK: CodC

Oggetto(CodO, Nome, Descrizione, CodC)

PK: CodO

FK: CodC REFERENCES Categoria

Recensione(CodR, Foto, DataVisionePubblica, Titolo, Valore,

Descrizione, DataPubblicazione, CodO, CodUtente)

PK: CodR

FK: CodO REFERENCES Oggetto

FK: CodUtente REFERENCES Utente

Medaglia(CodUtente, CodR, Timestamp)

PK: (CodUtente, CodR)

FK: CodUtente REFERENCES Utente

FK: CodR REFERENCES Recensione

Report(CodUtente, CodR, Tipo, Altro)

PK: (CodUtente, CodR)

FK: CodUtente REFERENCES Utente

FK: CodR REFERENCES Recensione

Commento(CodC, CodR, CodU, Timestamp, Testo, CodCRisposta)

PK: (CodC, CodR)

FK: CodR REFERENCES Recensione

FK: CodU REFERENCES Utente

FK: (CodCRisposta, CodR) REFERENCES Commento

Messaggio(CodM, CodMittente, CodDestinatario, Letto, Timestamp)

PK: (CodM, CodMittente, CodDestinatario)

FK: CodMittente REFERENCES Utente

FK: CodDestinatario REFERENCES Utente

MTesto(CodM, CodMittente, CodDestinatario, Contenuto)

PK: (CodM, CodMittente, CodDestinatario)

FK: (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio

MImmagine(CodM, CodMittente, CodDestinatario, Immagine, Descrizione)

PK: (CodM, CodMittente, CodDestinatario)

FK: (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio

MRecensione(CodM, CodMittente, CodDestinatario, CodR)

PK: (CodM, CodMittente, CodDestinatario)

FK: (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio

FK: CodR REFERENCES Recensione

CartaCredito(Numero, DScadenza, EnteEmittente)

PK: Numero

CartaUtente(CodCarta, CodU)

PK: (CodCarta, CodU)

FK: CodU REFERENCES Utente

FK: CodCarta REFERENCES Carta

Piano(CodP, Quantita, Periodo, Attivo, CodUtentePremium)

PK: CodP

FK: CodUtentePremium REFERENCES UtentePremium

Esclusivita(DAnticipata, CodP, CodR)

PK: (CodP, CodR)

FK: CodP REFERENCES Piano

FK: CodR REFERENCES Recensione

Iscrizione(CodP, CodUtente, DIscrizione, DAbbandono)

PK: (CodP, CodUtente, DIscrizione)

FK: CodUtente REFERENCES Utente

FK: CodP REFERENCES Piano

TransazioneAutomatica(CRO, Annullata, Data, CodPiano, CodUtente,

NumeroCartaDiCredito)

PK: CRO

FK: CodPiano REFERENCES Piano

FK: CodUtente REFERENCES Utente

FK: NumeroCartaDiCredito REFERENCES CartaCredito

AK: (Data, CodPiano, CodUtente)

TransazioneManuale(CRO, Annullata, Data, CodMittente, CodDestinatario,

NumeroCartaDiCredito)

PK: CRO

FK: CodMittente REFERENCES Utente

FK: CodDestinatario REFERENCES Utente

FK: NumeroCartaDiCredito REFERENCES CartaCredito

AK: (Data, CodMittente, CodDestinatario)

Ban(CodUtente, CodModeratore)

PK: CodUtente

FK: CodUtente REFERENCES Utente

FK: CodModeratore REFERENCES Moderatore

Rimozione(DataEffettuazione, CodR, DAnnullamento, CodModeratore)

PK: (DataEffettuazione, CodR)

FK: CodR REFERENCES Recensione

FK: CodModeratore REFERENCES Moderatore

Verifica della normalizzazione

Una volta effettuata l’analisi di normalizzazione si può notare che le relazioni sono in forma normale.

Operazioni previste dalla base di dati – Descrizione e relativo codice SQL

Vi sono alcuni vincoli che vengono gestiti con appositi CHECK, in particolare: il formato delle email, degli username e degli indirizzi IP, il vincolo del valore della recensione che deve essere compreso nell’intervallo 0-100, e infine la data di annullamento di una rimozione e la data di abbandono di un piano, che devono essere rispettivamente maggiore della data di effettuazione della rimozione e della data di iscrizione a tale piano. I rimanenti verranno gestiti con appositi TRIGGER, in aggiunta al mantenimento dei dati derivati.

Si hanno le seguenti query di inserimento:

CREATE TABLE Moderatore(

Email varchar(320) UNIQUE,

Username varchar(55) PRIMARY KEY,

Nome varchar(55) NOT NULL,

Cognome varchar(55) NOT NULL,

Psw char(64) NOT NULL

);

CREATE TABLE Utente(

Email varchar(320) UNIQUE,

Username varchar(55) PRIMARY KEY,

Nome varchar(55) NOT NULL,

Cognome varchar(55) NOT NULL,

IP char(15) NOT NULL,

Psw char(64) NOT NULL,

Foto varchar(55)

);

CREATE TABLE UtentePremium(

CodUtente varchar(55) PRIMARY KEY,

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

IBAN char(27) NOT NULL

);

CREATE TABLE Amicizia(

Seguente varchar(55),

Seguito varchar(55),

FOREIGN KEY (Seguente) REFERENCES Utente(Username),

FOREIGN KEY (Seguito) REFERENCES Utente(Username),

PRIMARY KEY (Seguente, Seguito),

Data timestamp NOT NULL

);

CREATE TABLE Oggetto(

CodO int PRIMARY KEY AUTO\_INCREMENT,

Nome varchar(55) NOT NULL,

Descrizione varchar(320) NOT NULL,

Categoria varchar(55) NOT NULL,

);

CREATE TABLE Recensione(

CodR int PRIMARY KEY,

Foto varchar(55),

DataPubblicazione date NOT NULL,

DataVisionePubblica date NOT NULL,

Titolo varchar(155) NOT NULL,

Valore int,

Descrizione varchar(1555) NOT NULL,

CHECK(Valore >= 0 AND Valore <= 100),

CodUtente varchar(55) NOT NULL,

CodO int NOT NULL,

FOREIGN KEY (CodO) REFERENCES Oggetto(CodO),

FOREIGN KEY (CodUtente) REFERENCES Utente(Username)

);

CREATE TABLE Medaglia(

CodUtente varchar(55),

CodR int,

Data timestamp NOT NULL,

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

FOREIGN KEY (CodR) REFERENCES Recensione(CodR),

PRIMARY KEY (CodUtente, CodR)

);

CREATE TYPE tipoR AS ENUM('Contenuti di natura sessuale', 'Contenuti violenti o ripugnanti', 'Azioni dannose o pericolose', 'Spam o ingannevole');

CREATE TABLE Report(

CodUtente varchar(55),

CodR int,

TipoReport tipoR,

Altro varchar(55),

CHECK((TipoReport IS NULL AND Altro IS NOT NULL) OR (TipoReport IS NOT NULL AND Altro IS NULL))

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

FOREIGN KEY (CodR) REFERENCES Recensione(CodR),

PRIMARY KEY (CodUtente, CodR)

);

CREATE TABLE Commento(

CodC int,

CodR int,

CodUtente varchar(55) NOT NULL,

Data timestamp NOT NULL,

Testo varchar(1555),

CodCRisposta int NOT NULL,

PRIMARY KEY(CodC, CodR),

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

FOREIGN KEY (CodR) REFERENCES Recensione(CodR),

FOREIGN KEY (CodCRisposta, CodR) REFERENCES Commento,

PRIMARY KEY (CodC, CodR)

);

CREATE TABLE Messaggio(

CodM int,

CodMittente varchar(55),

CodDestinatario varchar(55),

Letto boolean NOT NULL,

Data timestamp NOT NULL,

FOREIGN KEY (CodDestinatario) REFERENCES Utente(Username),

FOREIGN KEY (CodMittente) REFERENCES Utente(Username),

PRIMARY KEY (CodM, CodMittente, CodDestinatario)

);

CREATE TABLE MTesto(

CodM int,

CodMittente varchar(55),

CodDestinatario varchar(55),

Contenuto varchar(1555),

FOREIGN KEY (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio(CodM, CodMittente, CodDestinatario)

PRIMARY KEY (CodM, CodMittente, CodDestinatario)

);

CREATE TABLE MImmagine(

CodM int,

CodMittente varchar(55),

CodDestinatario varchar(55),

Immagine varchar(55),

Descrizione varchar(1555),

FOREIGN KEY (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio(CodM, CodMittente, CodDestinatario)

PRIMARY KEY (CodM, CodMittente, CodDestinatario)

);

CREATE TABLE MRecensione(

CodM int,

CodMittente varchar(55),

CodDestinatario varchar(55),

codR int NOT NULL,

Descrizione varchar(1555),

FOREIGN KEY (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio(CodM, CodMittente, CodDestinatario)

FOREIGN KEY (CodR) REFERENCES Recensione(CodR),

PRIMARY KEY (CodM, CodMittente, CodDestinatario)

);

CREATE TABLE CartaCredito(

Numero char(16) PRIMARY KEY,

DScadenza char(5) NOT NULL,

EnteEmittente varchar(30) NOT NULL,

);

CREATE TABLE CartaUtente(

CodU varchar(55),

NumeroC char(16),

PRIMARY KEY(CodU, NumeroC),

FOREIGN KEY (CodU) REFERENCES Utente(Username),

FOREIGN KEY (NumeroC) REFERENCES CartaCredito(Numero)

);

CREATE TYPE periodoPiano as ENUM('settimana', 'mese', 'trimestre', 'semestre', 'anno');

CREATE TABLE Piano(

CodP int PRIMARY KEY,

CodR int NOT NULL,

CodUtentePremium varchar(55) NOT NULL,

Quantita int NOT NULL,

Periodo periodoPiano NOT NULL,

FOREIGN KEY (CodUtentePremium) REFERENCES UtentePremium(CodUtente),

FOREIGN KEY (CodR) REFERENCES Recensione(CodR)

);

CREATE TABLE Esclusivita(

DataAnticipata date NOT NULL,

CodP int,

CodR int,

FOREIGN KEY (CodR) REFERENCES Recensione(CodR),

FOREIGN KEY (CodP) REFERENCES Piano(CodP),

PRIMARY KEY (CodP, CodR)

);

CREATE TABLE Iscrizione(

CodP int,

CodUtente varchar(55),

DIscrizione date,

DAbbandono date,

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

FOREIGN KEY (CodP) REFERENCES Piano(CodP),

PRIMARY KEY (CodP, CodUtente, DIscrizione)

);

CREATE TABLE TransazioneAutomatica(

TRN char(30) PRIMARY KEY,

Annullata bool NOT NULL,

Data date NOT NULL,

CodPiano int NOT NULL,

CodUtente varchar(55) NOT NULL,

NumeroCartaDiCredito char(16) NOT NULL,

FOREIGN KEY (NumeroCartaDiCredito) REFERENCES CartaCredito(Numero),

FOREIGN KEY (CodPiano) REFERENCES Piano(CodP),

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

UNIQUE(Data, CodPiano, CodUtente)

);

CREATE TABLE TransazioneManuale(

TRN char(30) PRIMARY KEY,

Annullata bool NOT NULL,

Data date NOT NULL,

CodMittente varchar(55) NOT NULL,

CodDestinatario varchar(55) NOT NULL,

NumeroCartaDiCredito char(16) NOT NULL,

FOREIGN KEY (NumeroCartaDiCredito) REFERENCES CartaCredito(Numero),

FOREIGN KEY (CodMittente) REFERENCES Utente(Username),

FOREIGN KEY (CodDestinatario) REFERENCES Utente(Username),

UNIQUE(Data, CodMittente, CodDestinatario)

);

CREATE TABLE Ban(

CodUtente varchar(55) PRIMARY KEY,

CodModeratore varchar(55) NOT NULL,

FOREIGN KEY (CodUtente) REFERENCES Utente(Username),

FOREIGN KEY (CodModeratore) REFERENCES Utente(Username)

);

CREATE TABLE Rimozione(

DataEffettuazione date NOT NULL,

CodR int NOT NULL,

DAnnullamento date,

CodModeratore varchar(55) NOT NULL,

PRIMARY KEY(DataEffettuazione, CodR),

FOREIGN KEY (CodR) REFERENCES Recensione(CodR),

FOREIGN KEY (CodModeratore) REFERENCES Utente(Username)

);

Note sulla progettazione fisica

[ Dire la roba sulla sicurezza, permessi, studiare il costo di alcune query, valutare indici ]