Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

SENTENTIA

PROGETTO BASI DI DATI

A.A. 2021/2022

Leonardo Temperanza (152831)
Amanjot Singh (152792)
Alessandro Dodi (153355)
(Gruppo n.28)

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Indice dei contenuti

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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 di 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, foto profilo e indirizzo IP. 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, un timestamp, e inoltre occorre conoscere se siano stati letti o meno. 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 un collegamento ad una recensione.

Le interazioni tra utenti necessitano di un controllo per preservare la sicurezza della piattaforma, occorre dunque introdurre una figura il cui compito è quello di scorrere la lista dei contenuti della stessa ed eventualmente rimuoverli se si reputa che violino 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 e dell'indirizzo IP. 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. Non rappresenta un problema il fatto che un moderatore e un utente abbiano lo stesso username o la stessa email,

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

perché può capitare in certi casi rari che un moderatore sia anche un utente che fa utilizzo dei servizi erogati.

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, motivo per il quale la "cancellazione" di una recensione è in tutti i casi soltanto temporanea, dando la possibilità al creatore di contestarla 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. 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.

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, le medaglie, i commenti e le recensioni, ma anche il ban di una recensione, le varie transazioni, e le amicizie tra utenti.

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 di effettuazione 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 determinata data di visualizzazione anticipata. 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.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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 TRN, 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, in una certa data.

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 normali, 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 1,000 pubblicazioni (recensioni o commenti) in un'ora.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

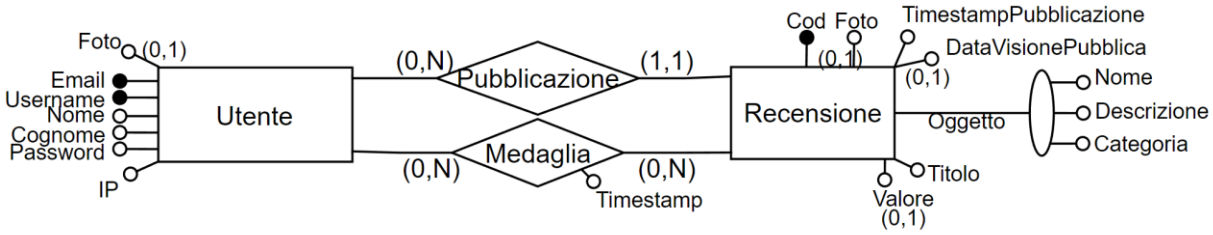
Glossario

Termine	Descrizione	Sinonimi	Legami
Persona	Email, Username, Nome, Cognome, Password	Individuo	Moderatore, Utente
Moderatore		Mod	Persona, Utente, Rimozione
Rimozione	DataEffettuazione, DAnnullamento	Cancellazione	Moderatore, Recensione
Utente	IP, Foto	Utilizzatore	Utente, Messaggio, Report, Commento, Recensione, UPremium, Iscrizione, CartaCredito, TManuale, TAutomatica
Report	Tipo, Altro	Segnalazione	Utente, Recensione
Messaggio	Timestamp, Letto, Cod	Chat	MRecensione, MImmagine, MTesto, Utente
MTesto	Contenuto	Text	Messaggio
MImmagine	Immagine, Descrizione	RiferimentoImmagine	Messaggio
MRecensione	Descrizione	RiferimentoRecensione	Messaggio, Recensione
UPremium	IBAN	UtentePremium	Utente, Piano, TManuale
CartaCredito	Numero, DScadenza, EnteEmittente		Utente, TManuale, TAutomatica
Transazione	TRN	Pagamento	TAutomatica, TManuale
TAutomatica	Data	PagamentoAutomatico	CartaCredito, Piano, Utente, Transazione
TManuale	Data, Quantità	PagamentoManuale	CartaCredito, Utente, UPremium, Transazione
Piano	Quantità, Attivo, Periodo, Cod	Servizio	TAutomatica, Iscrizione, UPremium, Recensione
Commento	Testo, Timestamp, Cod	Opinione	Commento, Recensione, Utente
Recensione	Cod, Foto, TimestampPubblicazione, DataVisionePubblica, Oggetto, Titolo, Valore	Valutazione	Rimozione, Report, Commento, Utente, MRecensione, Piano
Iscrizione	DIscrizione, DAbbandono	Sottoscrizione	Utente, Piano

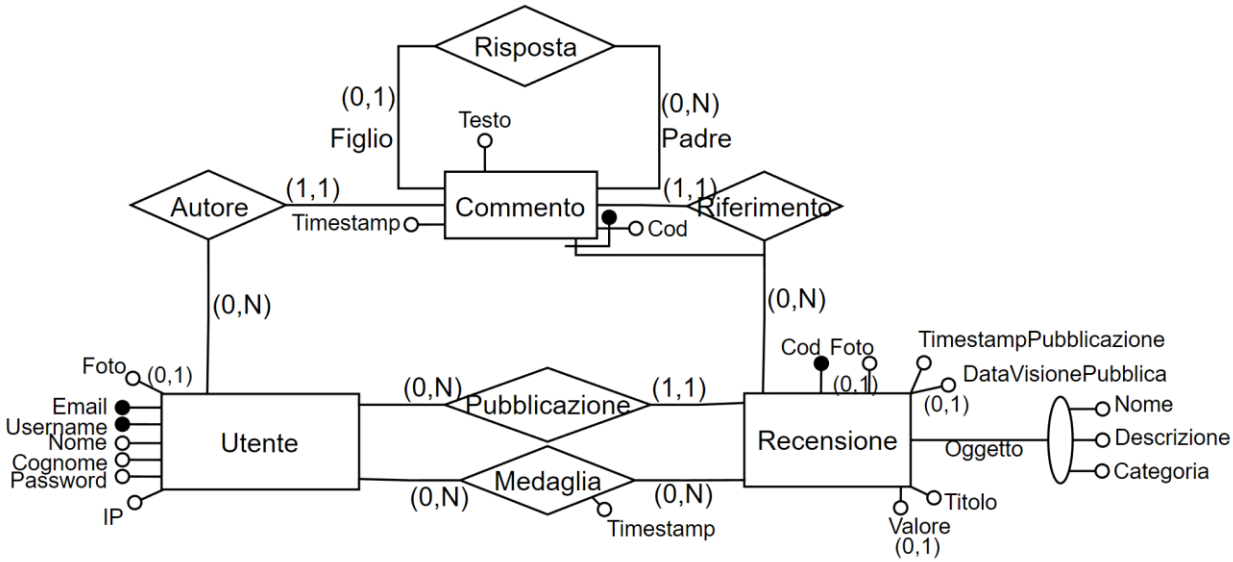
Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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 E/R (estremamente semplificato e privo della maggior parte delle funzionalità richieste):



Gli attributi “TimestampPubblicazione” e “DataVisionePubblica” sono relativi all’esclusività delle recensioni e pertanto verranno spiegati successivamente. 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.



Si applica una strategia di progettazione bottom-up, dove si sviluppano più sottoprogetti separati per poi metterli insieme per ottenere lo schema finale. Il vantaggio sta nel fatto che è più facile da gestire una visione ristretta rispetto ad una globale, mentre il lato negativo è la difficoltà di integrazione, ovvero il fatto che mettere insieme più parti eterogenee può rivelarsi essere

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

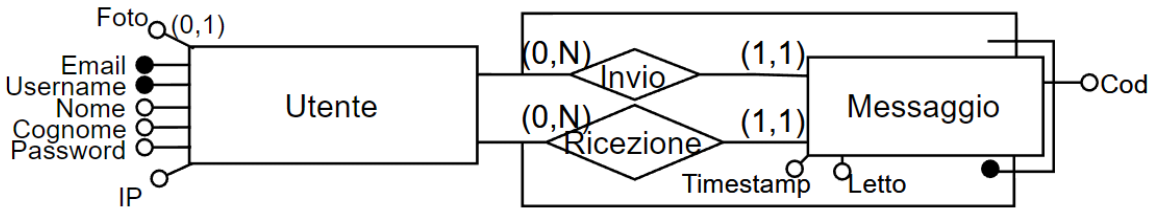
difficoltoso. Si ritiene la scelta adatta per questo progetto in quanto è naturalmente scomponibile in più parti che sono relativamente indipendenti tra di loro.

Per ragionare sul problema risulta quindi efficace scomporlo in più componenti:

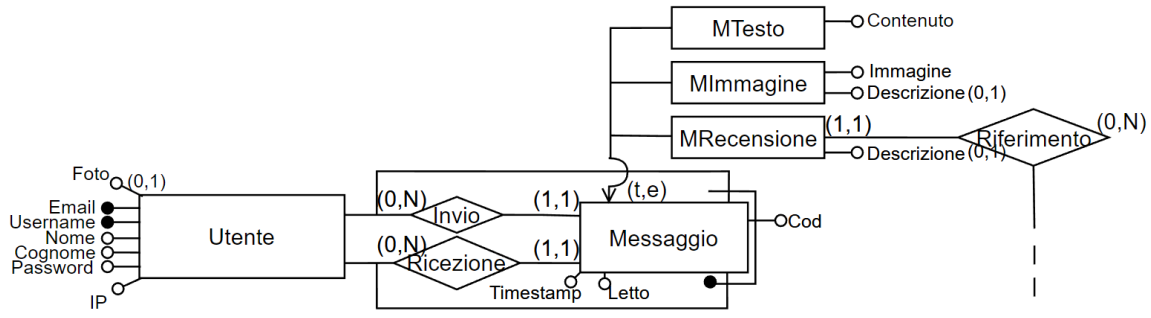
1. La messaggistica e le amicizie tra utenti
2. I moderatori e il controllo dei contenuti
3. Gli utenti premium e le transazioni

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 in entrambe è il 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.

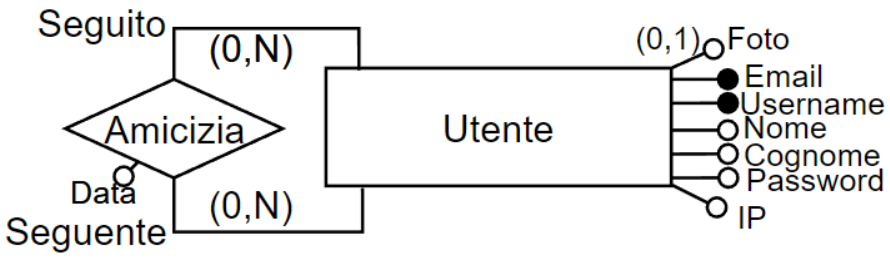


Si tratta di una gerarchia totale ed esclusiva perché un messaggio può essere solamente uno e uno solo di quei 3 tipi. Ci si rende conto del fatto che la descrizione opzionale è comune alle due

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

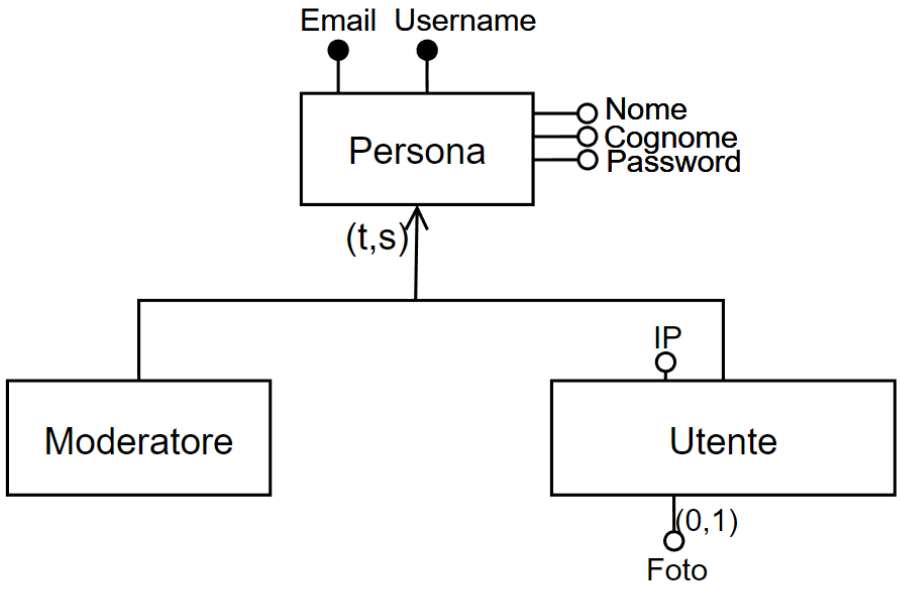
entità MImmagine e MRecensione, si potrebbe dunque riscrivere come un'altra gerarchia che ha come entità padre "MNonTesto" o similmente denominata, tuttavia per questioni di semplicità di rappresentazione si sceglie di evitare di modificare il diagramma in questo modo. In ogni caso le decisioni di traduzione in uno schema logico verranno intraprese nella fase successiva.

Le amicizie vengono modellate attraverso un'auto-associazione molti a molti, dato che un utente può richiedere l'amicizia a più utenti diversi.



I moderatori e gli utenti normali condividono una grande porzione di dati, è possibile introdurre in questo caso una gerarchia totale e sovrapposta, dove l'entità padre viene chiamata "Persona". È totale perché non ci si interessa delle persone che non sono né utenti né moderatori, ed è sovrapposta perché, come indicato dal testo, ci possono essere moderatori che sono al contempo utenti.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

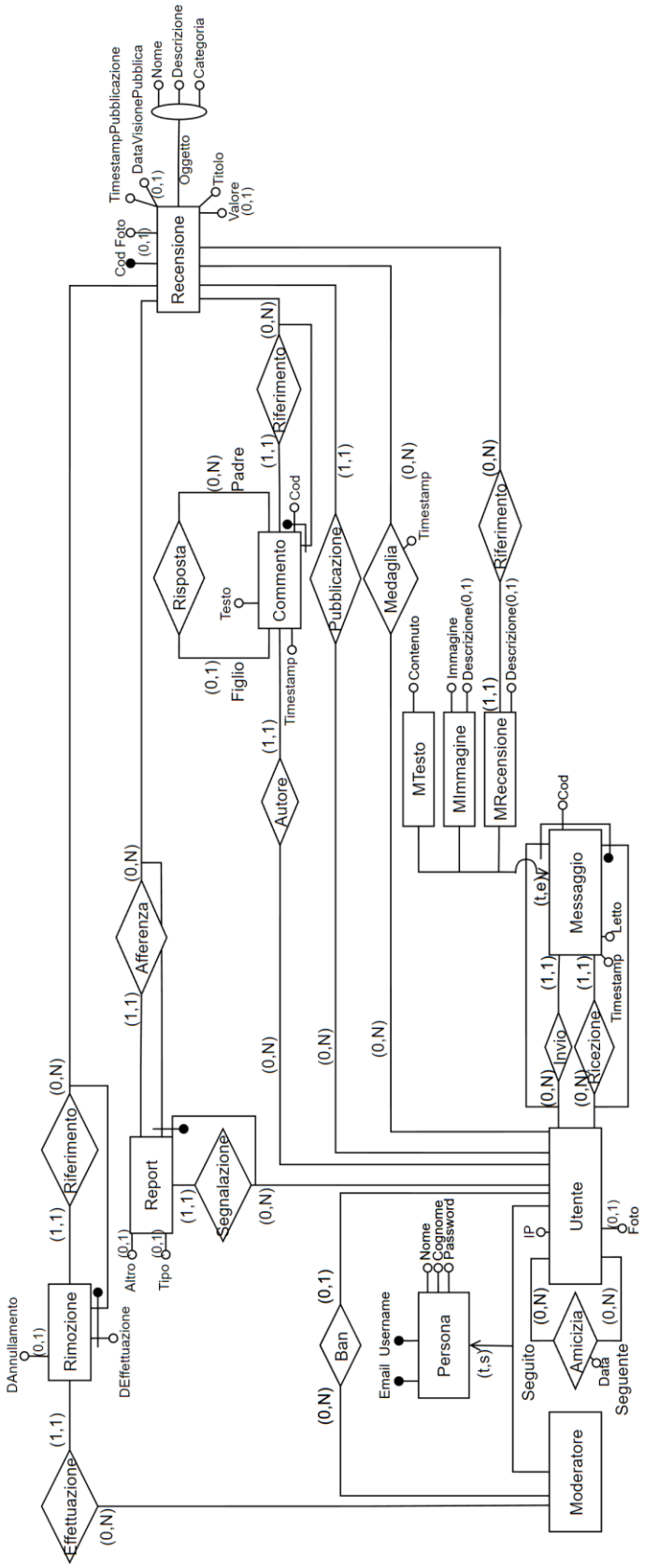


Un moderatore può bandire un determinato utente, e questo si traduce come una semplice associazione uno a molti.

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.

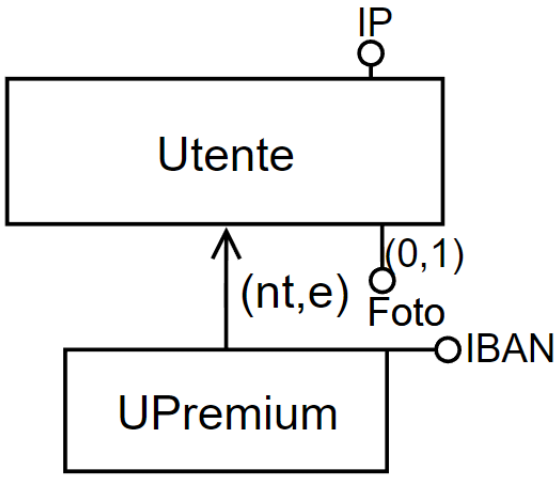
Di seguito è riportato il diagramma E/R aggiornato.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Per quanto riguarda il sistema di monetizzazione di questo progetto, innanzitutto si può notare che gli utenti premium non sono altro che utenti 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.

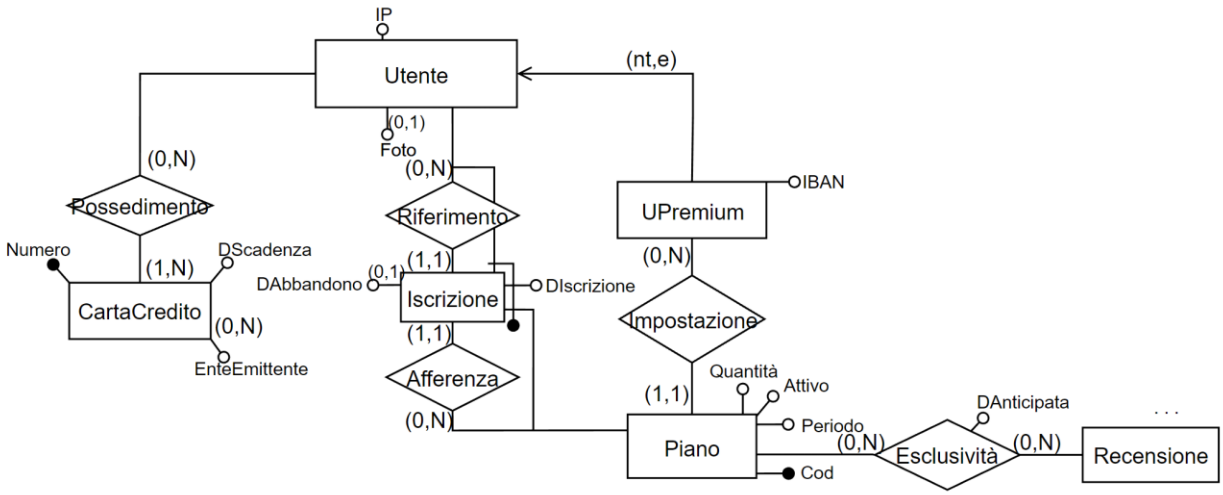


La gerarchia è parziale perché un utente può non essere premium (gli utenti normali), e una discussione sull'esclusività in una gerarchia con una sola entità figlia è per certi aspetti irrilevante.

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. Per riprendere il discorso dell'esclusività, vi sono due proprietà di Recensione: TimestampPubblicazione, ovvero il momento in cui la recensione viene aggiunta alla base di dati, e DataVisionePubblica che indica la data oltre la quale tutti gli utenti sono in grado di visualizzarla. Quest'ultima può essere nulla nel caso in cui non sia stata ancora decisa oppure nel caso in cui si voglia abilitare un'esclusività definitiva della recensione. L'idea è che un utente può accedere ad una recensione nel momento in cui essa ha DataVisionePubblica non nulla e inferiore/uguale alla data attuale, oppure nel caso in cui tale utente sia iscritto ad un piano che ne abilita l'esclusività con DataAnticipata minore/uguale alla data corrente. Una DataVisionePubblica nulla significa che il creatore non l'ha ancora decisa oppure che sia sua intenzione mantenere la recensione privata e solamente visibile se si è iscritti ad un determinato piano. 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.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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 finora considerato è il seguente:



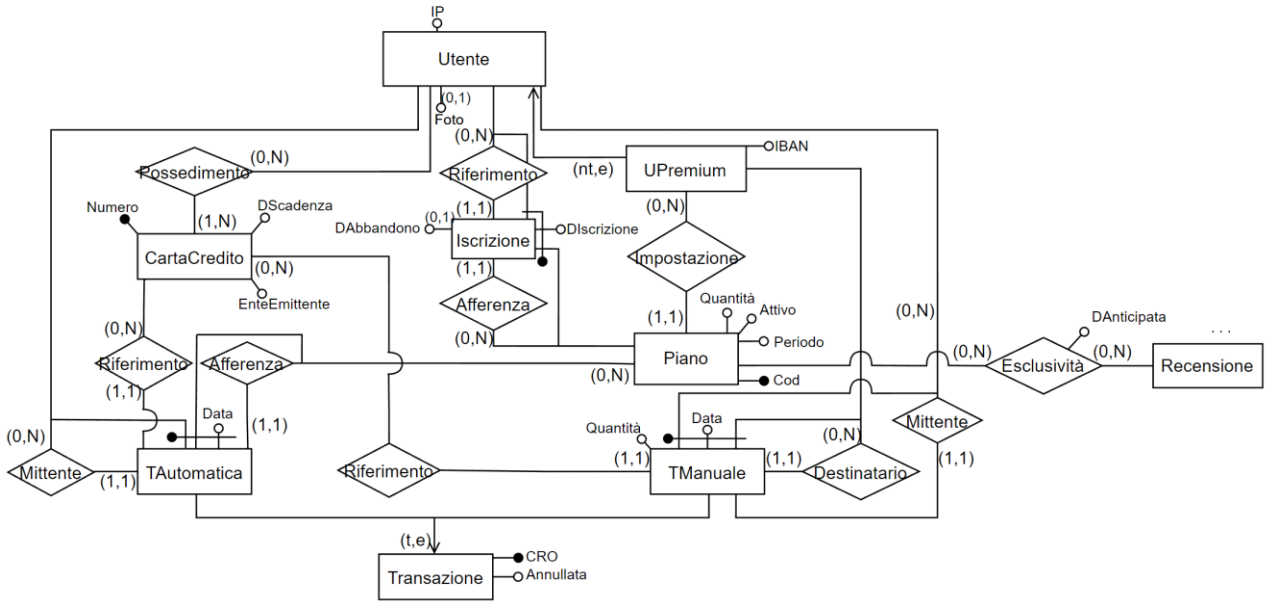
Le due tipologie di transazione sono per certi aspetti differenti, in quanto la transazione manuale deve contenere la somma di denaro donata, la data, l'utente premium destinatario della donazione, la carta di credito utilizzata e l'utente donatore, mentre la transazione automatica si riferisce ad un piano che contiene già l'utente premium e la somma di denaro. Si realizza una gerarchia totale ed esclusiva per raggruppare le caratteristiche comuni delle due tipologie. 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 al 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

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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 ad 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.

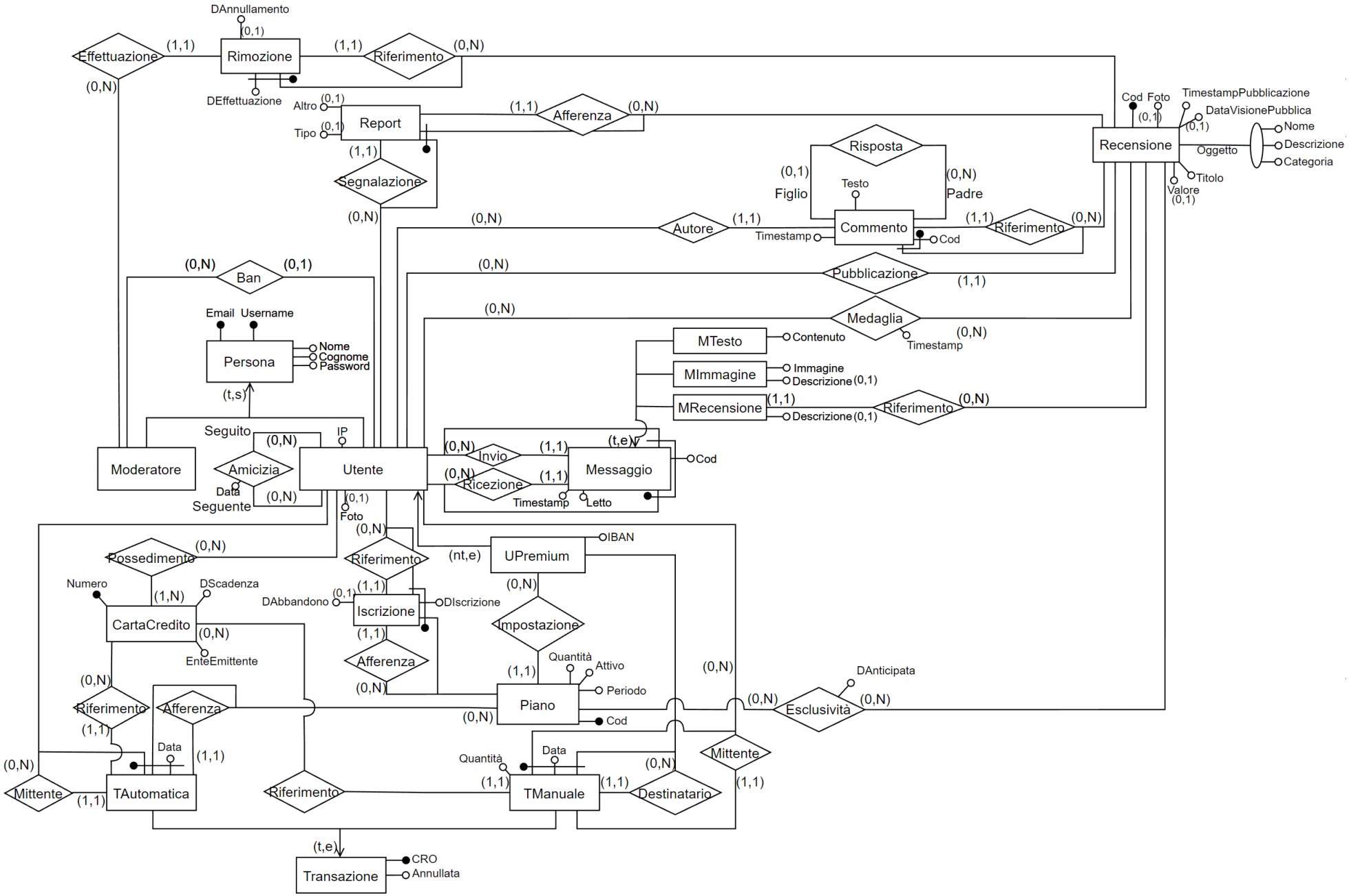
In seguito a queste riflessioni, il diagramma E/R risultante è il seguente:



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.

Schema E/R complessivo

Si presenta dunque lo schema finale:


DAnnullamento
(0,1)
Effettuazione
(1,1)
Rimozione
(1,1)
Riferimento
(0,N)
(0,N)
DEffettuazione
Altro
(0,1)
Tipo
(0,1)
Report
(1,1)
Afferenza
(0,N)
(1,1)
Segnalazione
(0,N)
Risposta
(0,1)
Figlio
Testo
(0,N)
Padre
Autore
(1,1)
Commento
Timestamp
Cod
(1,1)
Riferimento
(0,N)
Recensione
Cod
Foto
(0,1)
TimestampPubblicazione
DataVisionePubblica
(0,1)
Oggetto
Nome
Descrizione
Categoria
Titolo
Valore
(0,1)
(0,N)
Ban
(0,1)
(0,N)
Pubblicazione
(1,1)
(0,N)
Medaglia
Timestamp
(0,N)
Email
Username
Persona
Nome
Cognome
Password
(t,s)
MTesto
Contenuto
MImmagine
Immagine
Descrizione
(0,1)
MRecensione
(1,1)
Descrizione
(0,1)
Riferimento
(0,N)
Seguito
(0,N)
IP
Moderatore
Amicizia
Data
Seguente
(0,N)
Utente
(0,1)
Foto
(0,N)
Invio
(1,1)
(t,e)
Messaggio
Cod
(0,N)
Ricezione
(1,1)
Timestamp
Letto
Possedimento
(0,N)
Riferimento
(0,N)
(nt,e)
UPremium
IBAN
Numero
(1,N)
DScadenza
CartaCredito
(0,N)
EnteEmittente
DAbbandono
(0,1)
(1,1)
Iscrizione
DIscrizione
(1,1)
Afferenza
(0,N)
Impostazione
(0,N)
(1,1)
Quantità
Attivo
Periodo
Piano
Cod
(0,N)
(0,N)
Riferimento
Afferenza
(1,1)
(0,N)
(0,N)
Esclusività
(0,N)
DAnticipata
Data
(1,1)
(0,N)
Mittente
(1,1)
TAutomatica
Riferimento
Quantità
Data
(1,1)
(1,1)
TManuale
(0,N)
Destinatario
(1,1)
Mittente
(t,e)
Transazione
CRO
Annullata

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Progetto logico – Schema relazionale

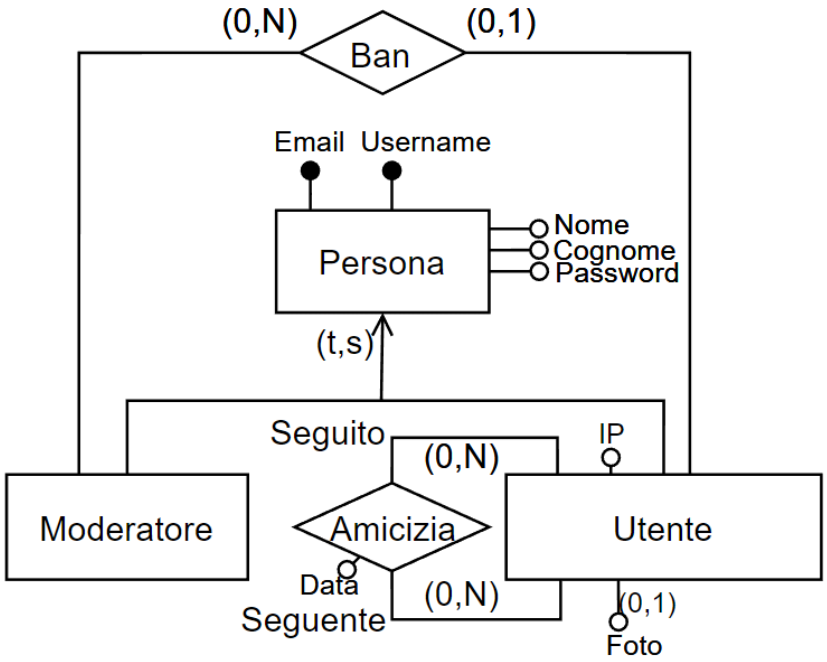
La progettazione logica permette di definire alcuni dettagli implementativi per poter operare nella pratica sui concetti stabiliti durante la fase di progetto concettuale, e viene decomposta nelle seguenti fasi:

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

Si procede nell'ordine descritto, e si concluderà con una sezione sui dati derivati seguita dallo schema logico completo.

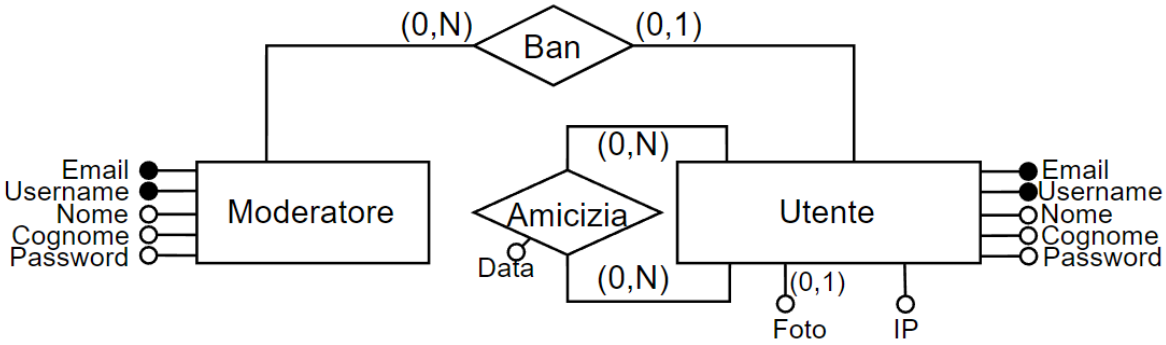
Eliminazione delle gerarchie ISA

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



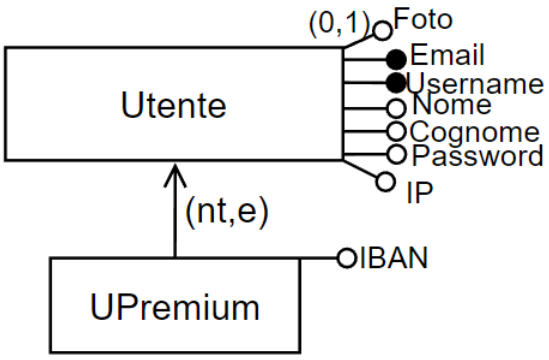
Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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:



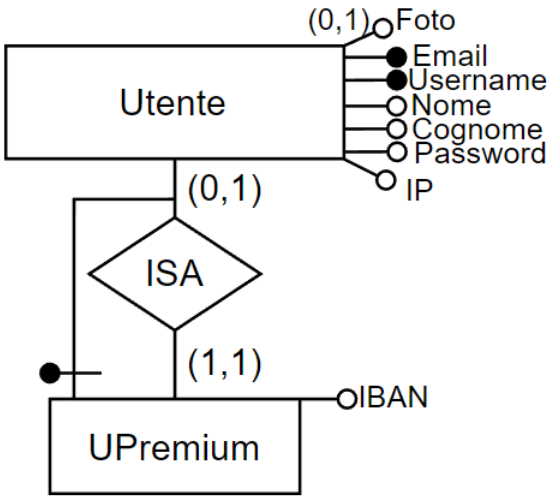
Questo richiederà, a livello implementativo, la creazione di alcuni trigger che si occuperanno di garantire che i valori di "Email" e "Username" siano univoci all'interno di entrambe le relazioni, dato che la gerarchia è esclusiva.

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

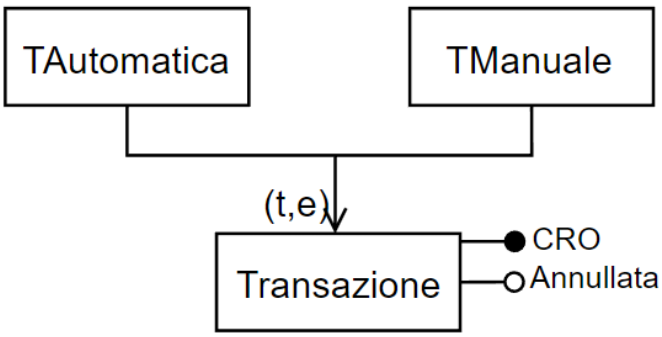


Vi sono numerose interazioni/associazioni che legano sia l'entità padre che l'entità figlia a tutte le altre, inoltre si può notare che vi sono numerose associazioni che legano le due entità Utente e UPremium, per esempio tutti gli utenti possono donare ma solamente gli utenti premium sono in grado di 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:

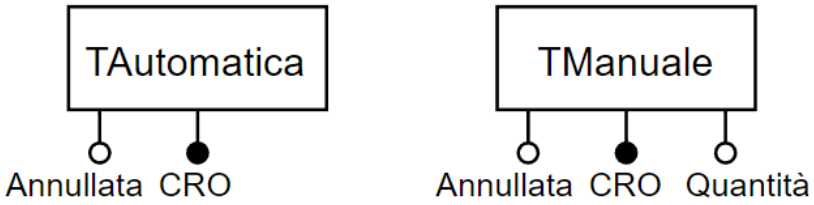
Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



Per quanto riguarda la gerarchia relativa alle transazioni:

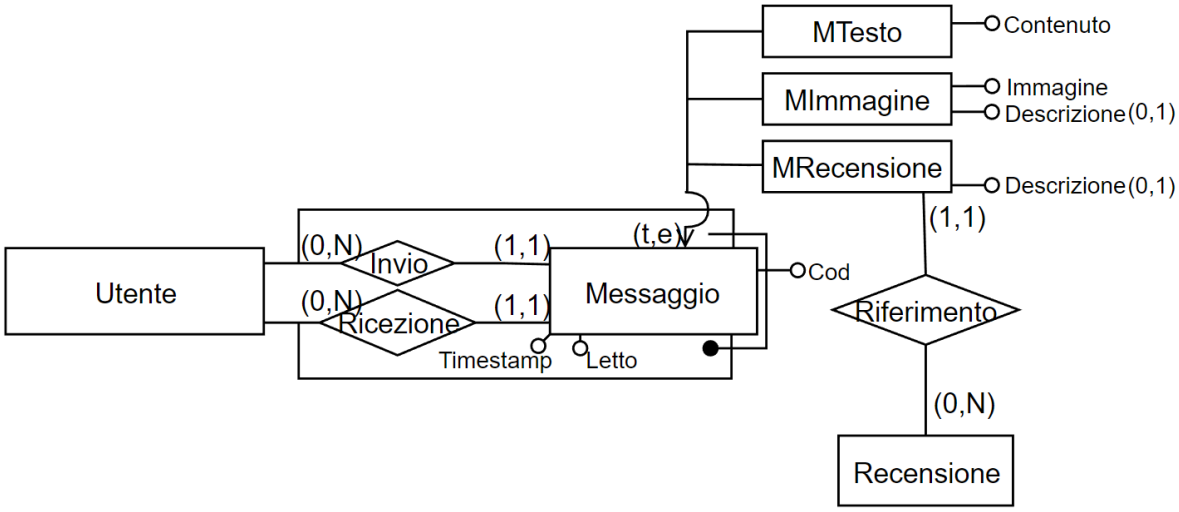


Essa può essere tradotta in più modi, tuttavia per il fatto che l'entità padre contiene solamente due attributi e nessuna associazione che la lega ad altre entità, un semplice collasso verso il basso è sufficiente.



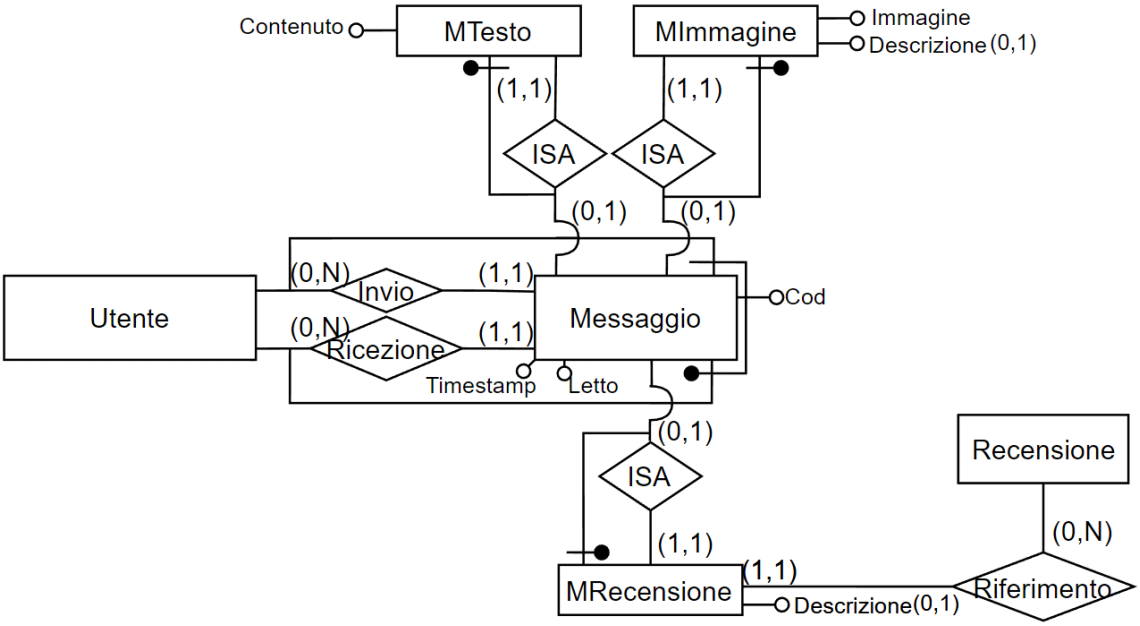
Infine occorre gestire l'eliminazione della gerarchia di Messaggio. Si tratta di una gerarchia totale ed esclusiva, rappresentata nell'immagine seguente:

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



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, e ha il lato negativo che in questo caso si occupa più spazio a causa della ripetizione della chiave composta. È stato scelto un mantenimento delle entità, da cui si ottiene:

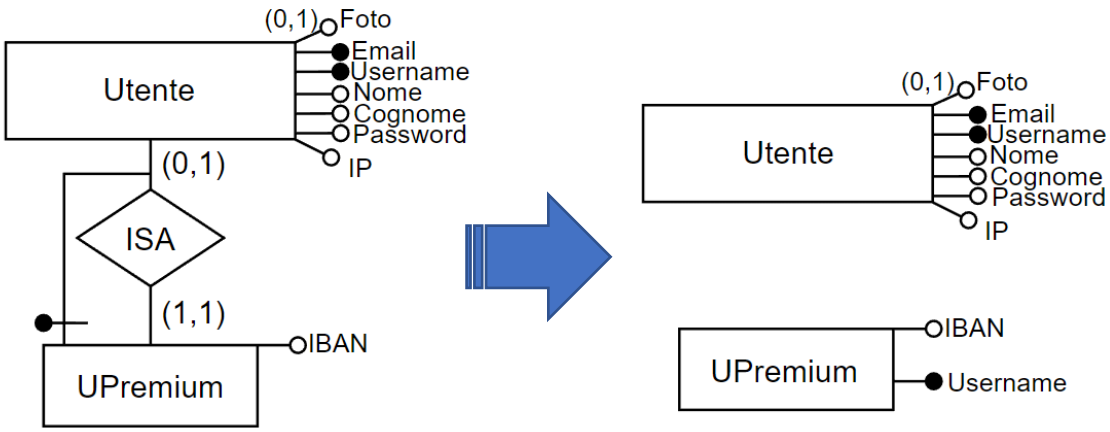
Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



Selezione delle chiavi primarie, eliminazione delle identificazioni esterne

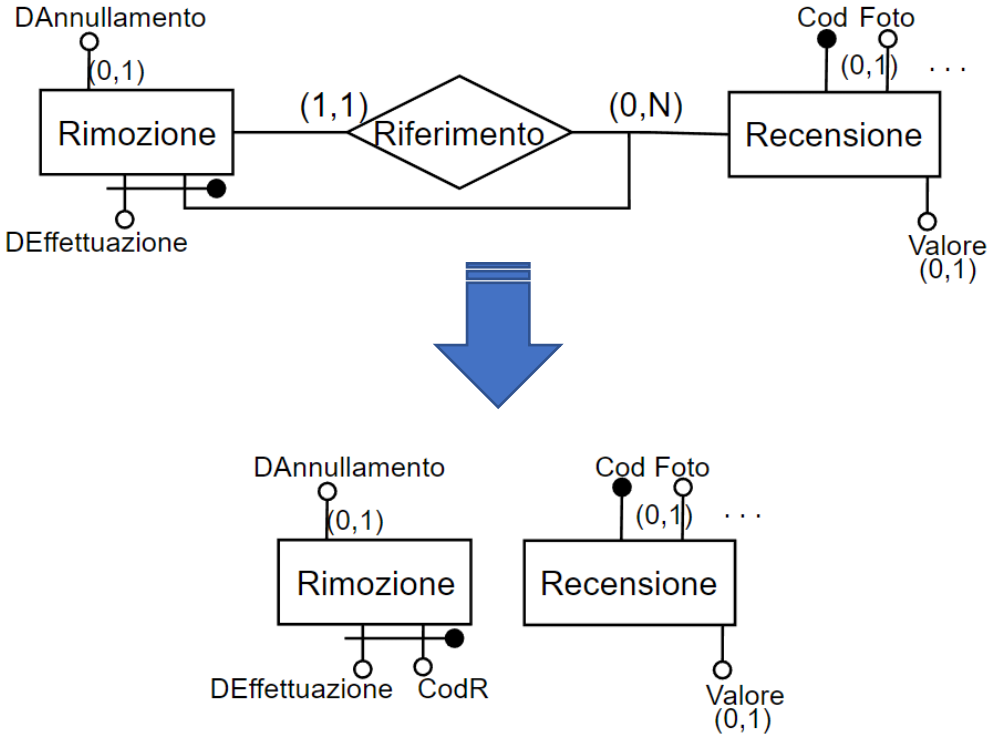
Vi è una scelta di selezione delle chiavi primarie solamente in quattro casi: Utente, Moderatore, TManuale, TAutomatica. Negli ultimi due casi una delle due chiavi è composta ed esterna, quindi occorrerà prima eliminare quella esterna per poi scegliere tra le due. Tra lo username e l'email, è stata scelta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

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



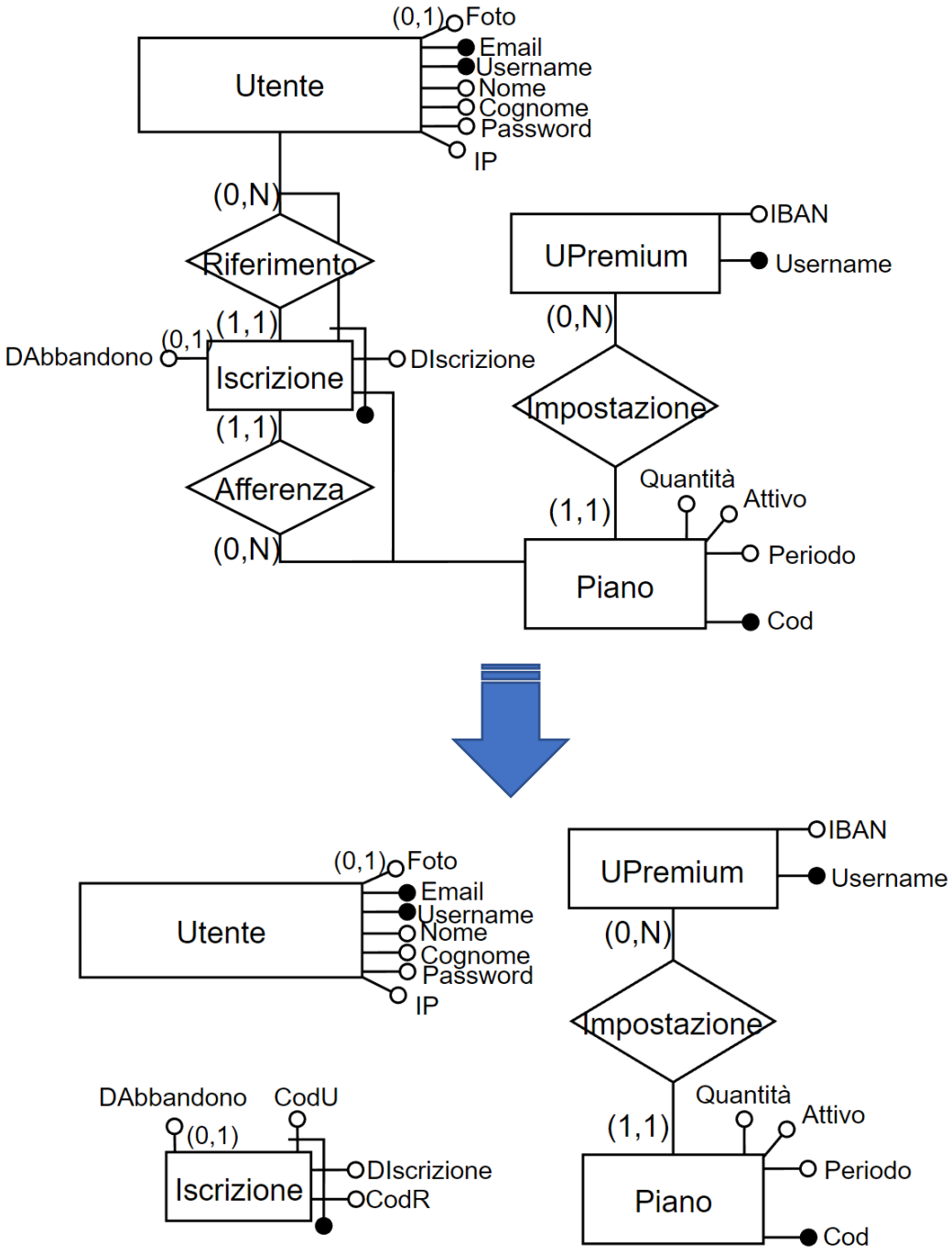
Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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



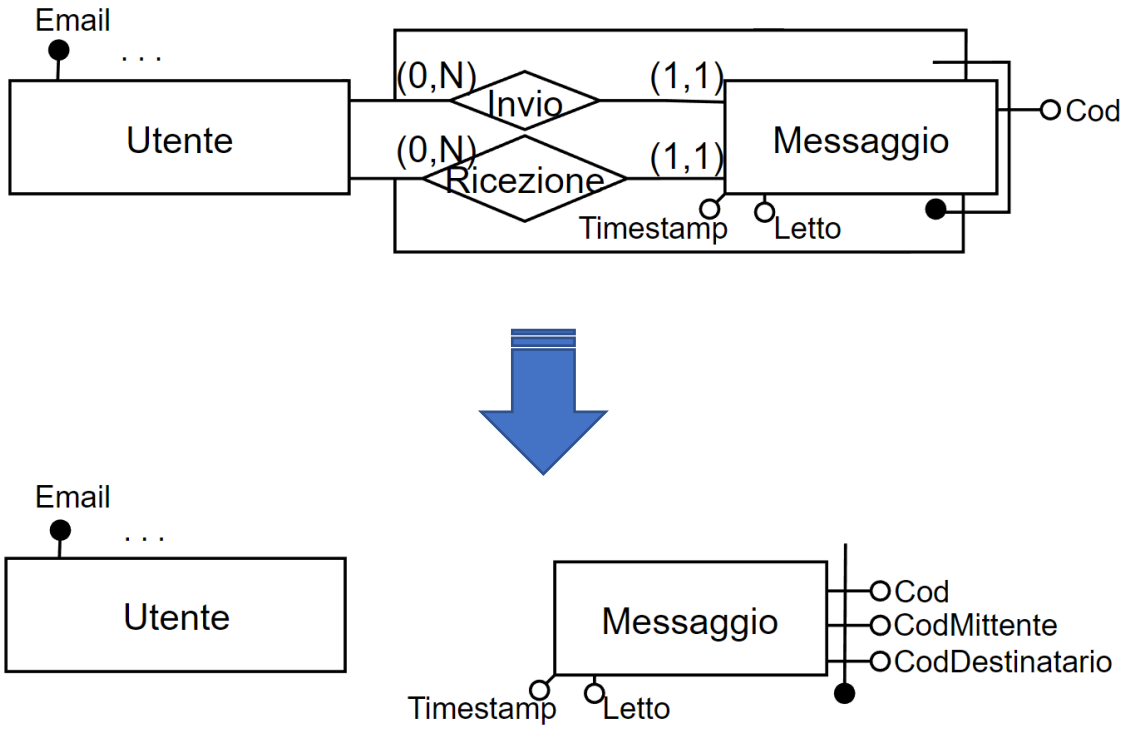
Allo stesso modo, l'entità Iscrizione ha una chiave che è formata dall'associazione Riferimento, Afferenza, e dalla data nella quale l'utente ha effettuato una determinata iscrizione ad un piano.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



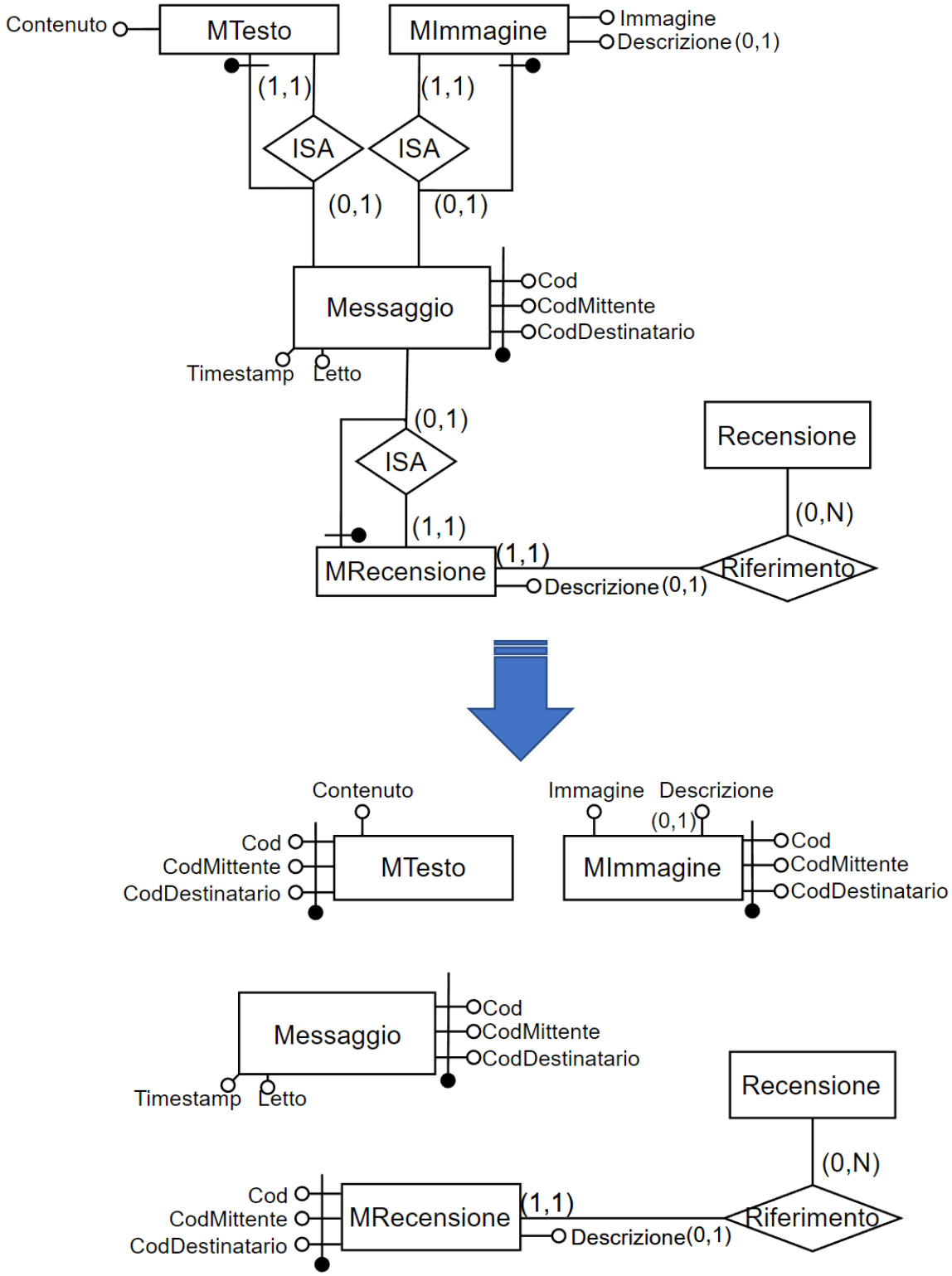
L’entità Messaggio è identificata da un codice che è solamente univoco tra i messaggi che hanno una determinata coppia di utenti come mittente-destinatario, che si traduce nel seguente modo:

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



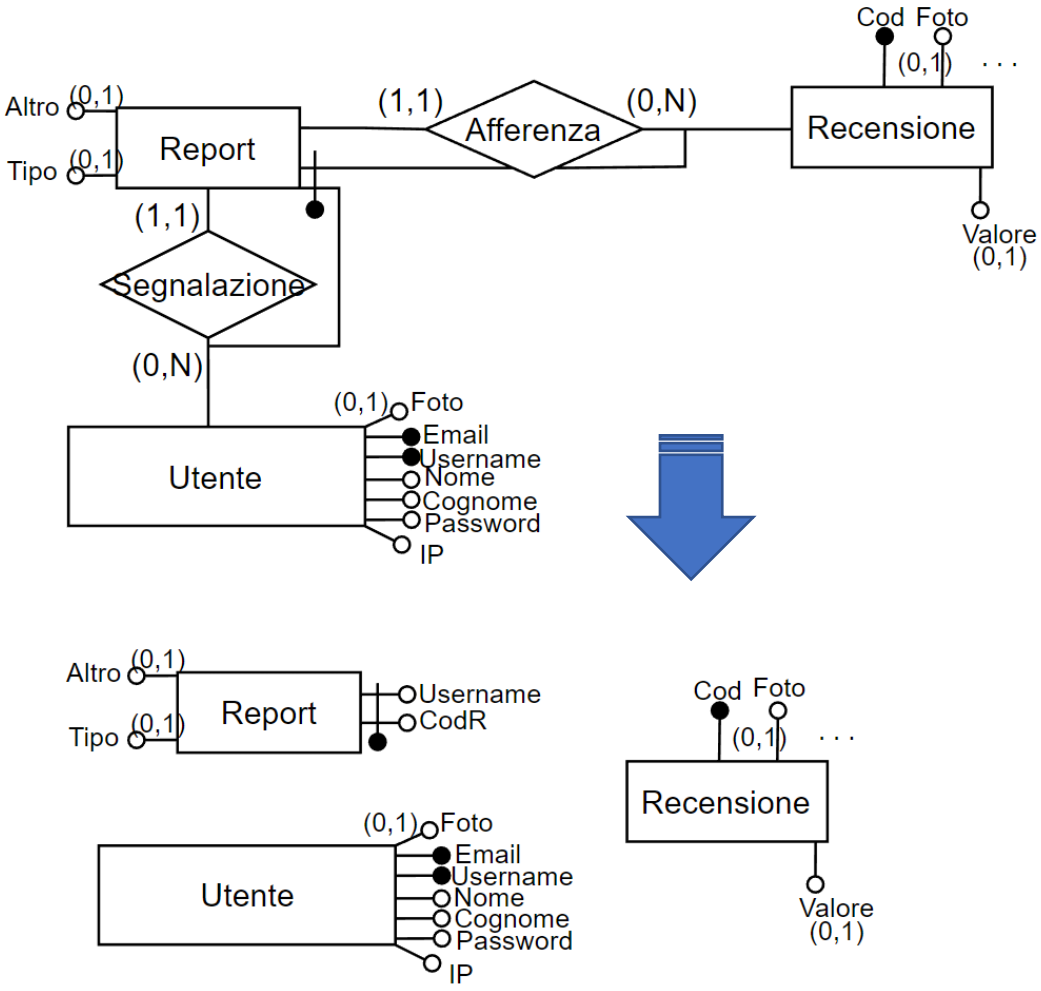
Le relazioni MTesto, MImmagine e MRecensioni vengono tutte trasformate in questo modo:

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



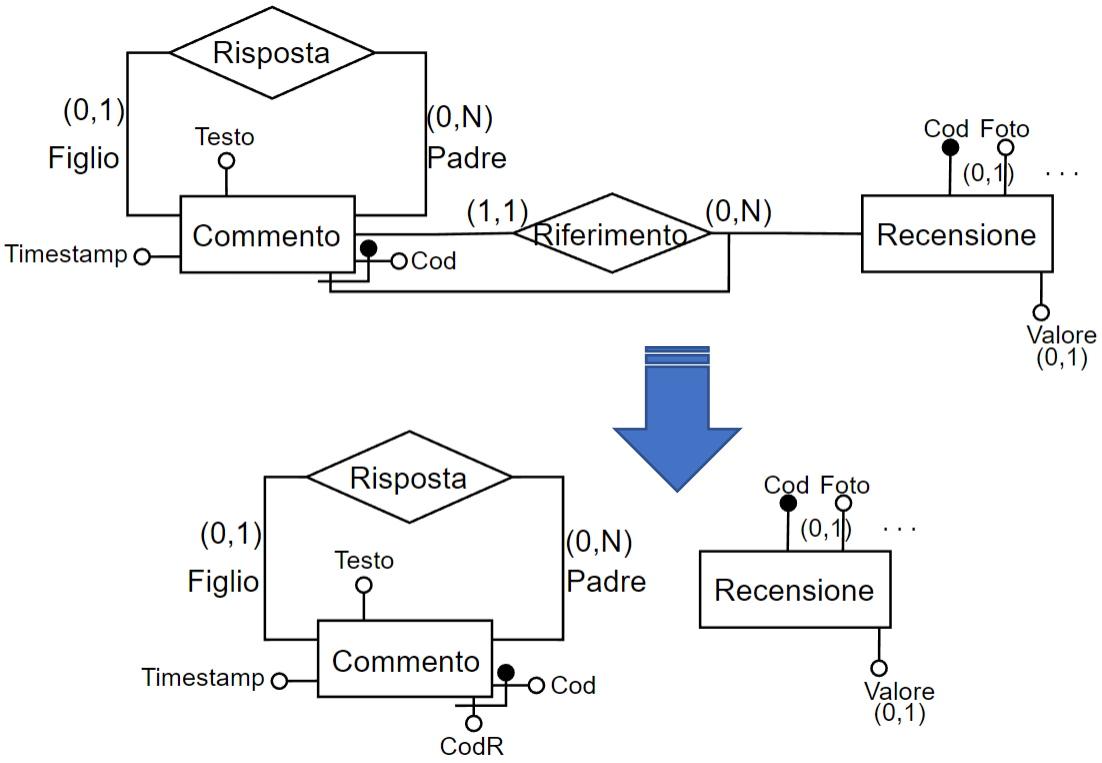
Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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



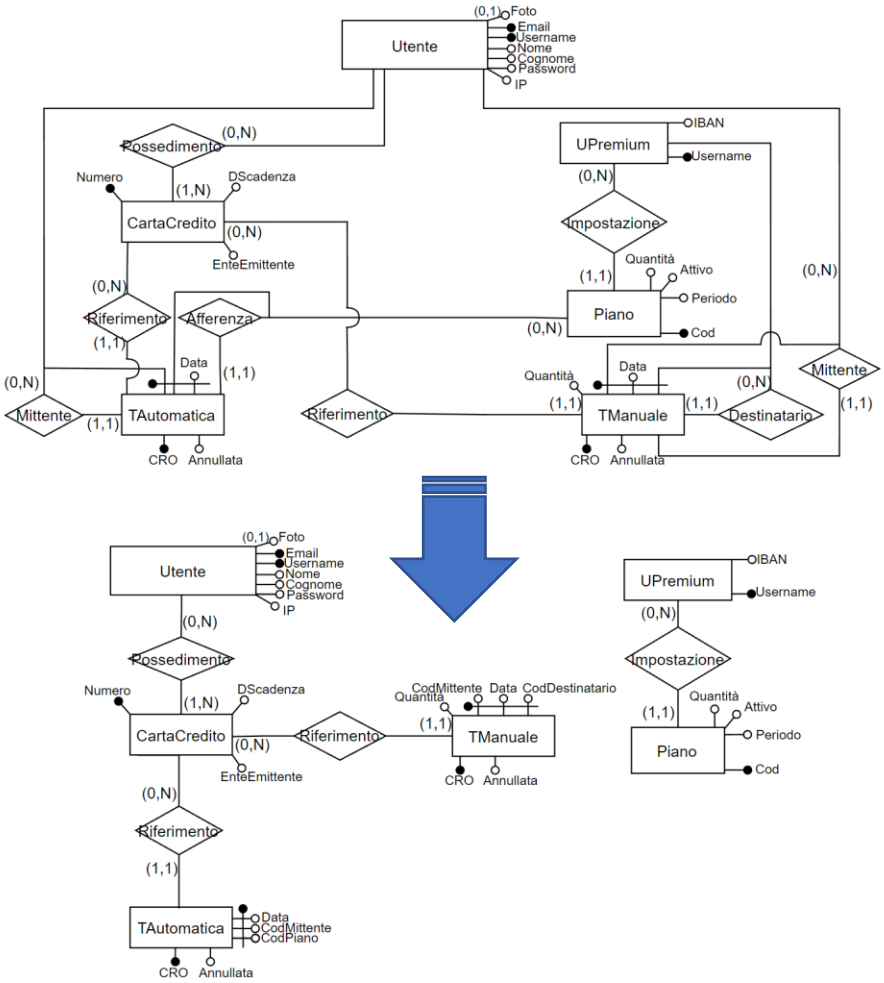
Un Commento è identificato da un codice univoco all'interno della recensione a cui si riferisce, e questo nella schema logico ha la seguente traduzione:

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



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.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



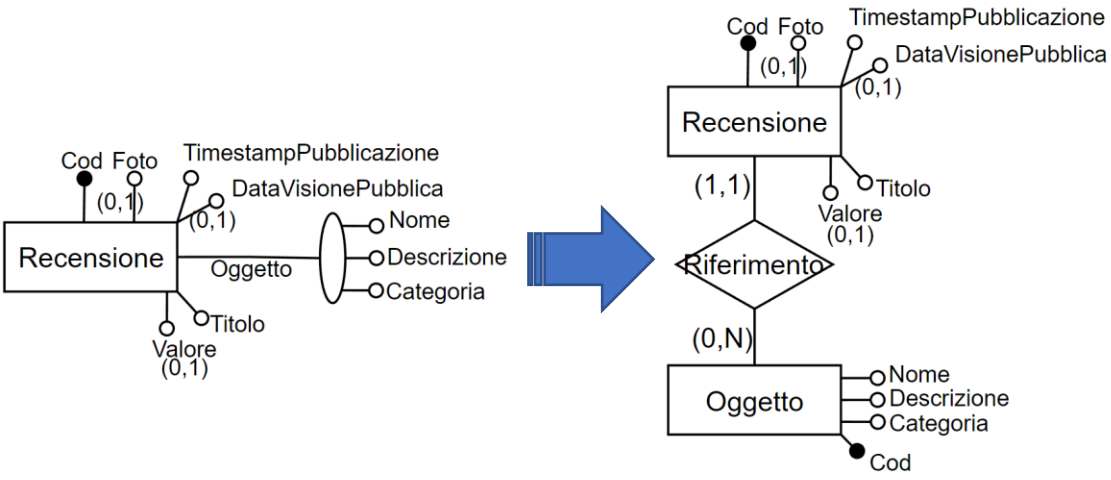
Per TAutomatica e TManuale si compie la stessa decisione sulla scelta della chiave primaria, ovvero il TRN, lasciando le due chiavi composte come chiavi alternative. Questo per il semplice fatto che in generale si preferisce una chiave semplice piuttosto che una complessa.

Trasformazione degli attributi multipli e/o composti

L'unico attributo composto presente all'interno dello schema E/R è l'oggetto della recensione. Dato che uno stesso oggetto, dotato di un titolo e di una descrizione che può essere in certi casi molto lunga, può ripetersi più volte, per non occupare eccessivamente spazio è stato deciso di creare una relazione che contiene i dati dell'oggetto e un codice intero identificativo che verrà utilizzato dalla recensione qualora volesse associarsi ad esso.

Si può osservare la trasformazione nel seguente schema E/R:

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355



Sono anche presenti alcuni attributi ripetuti, come la categoria dell'oggetto, il tipo di un report e l'ente emittente di una carta di credito; in tutti questi 3 casi è stato scelto di non trasformarli in entità separate in quanto è la scelta più semplice dal punto di vista progettuale, non occorre effettuare accessi in più su relazioni differenti per arrivare a queste informazioni, e lo spreco di spazio è alquanto trascurabile.

Traduzioni di entità e associazioni in schema di relazioni

Si prendono in esame prima di tutto le relazioni Moderatore, Utente, e UtentePremium.

UtentePremium non è legato a nessun'altra entità tramite associazione, perciò si può applicare una traduzione standard, per quanto riguarda invece Moderatore e Utente, sono legati da un'associazione uno a molti. Si suppone in generale che il numero di utenti "banditi" da un moderatore sia vastamente ridotto in proporzione al numero di utenti totale, motivo per il quale è stato deciso di tradurre l'associazione come una relazione a parte. L'amicizia tra utenti è un'auto-associazione N:N con partecipazione facoltativa da entrambe le parti, che viene tradotta come una relazione in possesso di due chiavi esterne che si riferiscono a due utenti separati. Si ricorda che un'amicizia non è necessariamente ricambiata, dunque non sono necessari trigger o quant'altro per imporre l'unicità della coppia (ovvero rendere impossibile gli inserimenti A-B e B-A). Lo schema logico risultante è il seguente:

```
Moderatore(Email, Username, Nome, Cognome, Password)
    PK: Username
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
    AK: Email
Utente(Email, Username, Nome, Cognome, Password, Foto, IP)
    PK: Username
    AK: Email
Ban(CodUtente, CodModeratore)
    PK: CodUtente
    FK: CodUtente REFERENCES Utente
    FK: CodModeratore REFERENCES Moderatore
UtentePremium(CodUtente, IBAN)
    PK: CodUtente
    FK: CodUtente REFERENCES Utente
Amicizia(Seguito, Seguente, Data)
    PK: (Seguito, Seguente)
    FK: Seguito REFERENCES Utente
    FK: Seguente REFERENCES Utente
```

Si passa ora a ciò che riguarda le recensioni. L'associazione 1:N tra Oggetto e Recensione viene tradotta accorpandola all'interno di Recensione, trattandosi di una partecipazione obbligatoria rispetto al lato molti. La separazione in una relazione a parte potrebbe avere determinati vantaggi in termini di compattezza e contiguità dei collegamenti, e nei casi in cui sia estremamente frequente la scansione delle coppie CodR-CodO questo può risultare vantaggioso in termini di efficienza di questa operazione. Tuttavia non si tratta del caso in questione, dunque si opta per una traduzione più comune, come riportato di seguito. Proseguendo con la traduzione, la Medaglia non è altro che un'associazione molti a molti con partecipazione facoltativa da entrambe le parti, che risulta nella creazione di una relazione distinta che contiene gli identificatori dell'utente che l'ha "creata" e della recensione a cui si riferisce.

```
Oggetto(CodO, Nome, Descrizione, Categoria)
    PK: CodO
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
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

L'entità Rimozione è legata alle entità Moderatore e Recensione tramite le associazioni Effettuazione e Riferimento rispettivamente, che sono 1:N e che possono essere accorpate nella relazione Rimozione, come mostra lo schema logico qui di seguito. Gli utenti possono effettuare report e commenti. L'associazione tra Report e Utente è 1:N, e la relativa traduzione consiste nel suo accorpamento nella relazione Report, e lo stesso può essere detto per l'associazione tra Report e Recensione. L'associazione tra Commento e Utente è 1:N, si può quindi accorpare all'entità Commento, e lo stesso ragionamento si può applicare per quanto riguarda l'associazione tra Commento e Recensione. Per quanto riguarda l'auto-associazione, la traduzione standard di accorpamento in Commento sarebbe caratterizzata dalla presenza di due attributi CodCRisposta e CodRRisposta, che costituiscono la chiave primaria dell'entità stessa. Con questa traduzione perfettamente valida, occorrerebbe creare un trigger per fare in modo che un commento non possa rispondere ad una recensione diversa da quella a cui si riferisce. In questo caso specifico, per via di com'è fatta la chiave di Commento, si può gestire questo vincolo a partire da questa fase. Basta eliminare l'attributo CodRRisposta e impostare la coppia (CodCRisposta, CodR) come Foreign Key che si riferisce ad un commento, dove CodR è il codice della recensione a cui si riferisce. In questo modo un commento può rispondere solamente ad un altro commento che si riferisce alla stessa recensione. Di seguito è riportato lo schema relativo a Report e Commento per esteso:

```
Rimozione(DataEffettuazione, CodR, DAnnullamento, CodModeratore)
    PK: (DataEffettuazione, CodR)
    FK: CodR REFERENCES Recensione
    FK: CodModeratore REFERENCES Moderatore
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
```

Tra Messaggio è Utente sono presenti due associazioni 1:N, entrambe con partecipazione facoltativa da parte dell'utente e obbligatoria da parte del messaggio. Queste vengono tradotte accorpandole alla relazione Messaggio. Le varie tipologie di messaggio MTesto, MImmagine e MRecensione, per effetto dell'eliminazione delle gerarchie ISA e delle identificazioni esterne non hanno alcune associazioni da analizzare e vengono quindi tradotte in maniera standard.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
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, Descrizione)
    PK: (CodM, CodMittente, CodDestinatario)
    FK: (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio
    FK: CodR REFERENCES Recensione
```

Tra le entità CartaCredito e Utente vi è un'associazione N:M. Essa viene tradotta come una relazione separata contenente i codici di un utente e delle sue rispettive carte di credito possedute. Tale relazione viene chiamata CartaUtente, per la mancanza di un nome migliore.

```
CartaCredito(Numero, DScadenza, EnteEmittente)
    PK: Numero
CartaUtente(CodCarta, CodU)
    PK: (CodCarta, CodU)
    FK: CodU REFERENCES Utente
    FK: CodCarta REFERENCES Carta
```

Un Piano è impostato da un solo UtentePremium, dunque l'associazione che lega le due entità viene accorpata nella relazione Piano, nello stesso modo in cui Iscrizione si trova in associazioni 1:N con Piano e Recensione. Tra Piano e Recensione vi è un'associazione N:M, da cui risulta naturale estrapolarla in una relazione a parte, denominata appunto Esclusività.

```
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
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
Iscrizione(CodP, CodUtente, DIscrizione, DAbbandono)
    PK: (CodP, CodUtente, DIscrizione)
    FK: CodUtente REFERENCES Utente
    FK: CodP REFERENCES Piano
```

I due tipi di transazioni sono simili e per questo motivo verranno descritti assieme. Le transazioni sono coinvolte in un'associazione 1:N con Utente, con CartaCredito, e nel caso di TransazioneAutomatica Piano, altrimenti UPremium. Tutte queste associazioni vengono accorpate nel rispettivo lato molti (TransazioneAutomatica e TransazioneManuale), come si può vedere nel seguente schema logico:

```
TransazioneAutomatica(TRN, Annullata, Data, CodPiano, CodUtente,
                      NumeroCartaDiCredito)
    PK: TRN
    FK: CodPiano REFERENCES Piano
    FK: CodUtente REFERENCES Utente
    FK: NumeroCartaDiCredito REFERENCES CartaCredito
    AK: (Data, CodPiano, CodUtente)
TransazioneManuale(TRN, Annullata, Data, CodMittente, CodDestinatario,
                   NumeroCartaDiCredito, Quantità)
    PK: TRN
    FK: CodMittente REFERENCES Utente
    FK: CodDestinatario REFERENCES UtentePremium
    FK: NumeroCartaDiCredito REFERENCES CartaCredito
    AK: (Data, CodMittente, CodDestinatario)
```

Verifica della normalizzazione

Una volta effettuata l'analisi di normalizzazione è stato appurato che le relazioni si trovano in forma normale.

Studio di dati derivati

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. Per una misurazione più accurata, gli

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

accessi in lettura si contano solo se strettamente necessari (non prima di ogni scrittura). Inoltre i dati sono puramente esemplificativi, potrebbero non essere completamente realistici.

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.

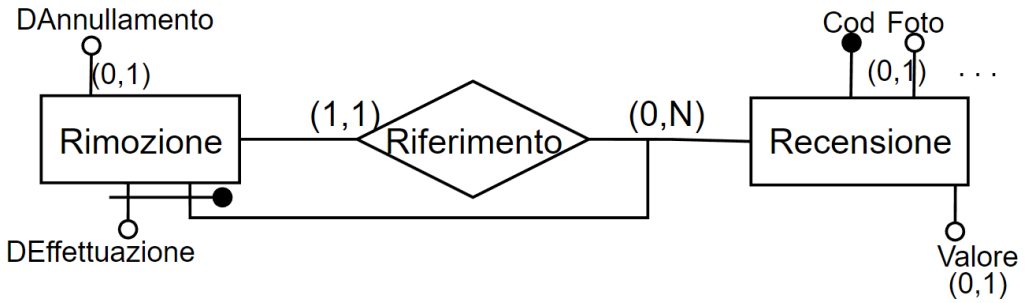


Tabella dei volumi:

Concetto	Tipo	Volume dati
Rimozione	E	50,000
Recensione	E	1,000,000

Tabella delle operazioni:

Operazione	Tipo	Frequenza
Rimozione Recensione	I	100/G
Annullamento Rimozione	I	8/G
Accesso Recensione	I	2,000,000/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). Si ricorda che l'associazione Riferimento è stata tradotta con un accorpamento a Recensione, dunque nello studio del dato derivato non si contano gli accessi di lettura/scrittura in Riferimento perché i suoi dati sono già contenuti all'interno di Rimozione.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Inoltre, si suppone di essere già a conoscenza del codice della recensione in caso di rimozione e di accesso, e nell'operazione di annullamento, oltre al codice della recensione si è a conoscenza della data di effettuazione dell'ultima rimozione (quella da annullare).

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

Rimozione recensione:

Concetto	Accessi	Tipo
Rimozione	1	S
Recensione	1	S

2*2 * 100 = 400/G

Annullamento rimozione:

Concetto	Accessi	Tipo
Rimozione	1	S
Recensione	1	S

2*2 * 8 = 32/G

Accesso recensione:

Concetto	Accessi	Tipo
Recensione	1	L

1* 2,000,000 = 2,000,000/G

Si ha quindi un totale di 2,000,432/G.

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

Rimozione recensione:

Concetto	Accessi	Tipo
Rimozione	1	S

1*2 * 100 = 200/G

Annullamento rimozione:

Concetto	Accessi	Tipo
Rimozione	1	S

1*2 * 8 = 16/G

Accesso recensione:

Concetto	Accessi	Tipo

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Rimozione	50,000/1,000,000	L
Recensione	1	L

(1+50,000/1,000,000) * 2,000,000 = 2,100,000/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, solitamente, su 100 recensioni a cui si vuole accedere, 95 siano visibili, il numero di accessi giornaliero si avvicina di più al seguente risultato:

(95/100+50,000/1,000,000) * 2,000,000 = 2,000,000/G

Si ha quindi un totale di 2,000,216/G.

Dunque, al contrario di quanto ipotizzato, il dato derivato non conviene in questo caso, anche se la differenza del numero di accessi giornalieri nei due casi è trascurabile. La conclusione potrebbe quindi cambiare con una piccola variazione dei dati e delle supposizioni fatte. È anche da esplicitare il fatto che per un'analisi più accurata sarebbe adatto effettuare misurazioni nella pratica, e il risultato potrebbe variare in presenza di dati che rappresentano più accuratamente un social network di medie dimensioni. Tuttavia per lo scopo di questo progetto si suppone che queste approssimazioni rappresentino la realtà con un errore sufficientemente ridotto.

Una richiesta di cui si è parlato precedentemente è la limitazione di un numero massimo di recensioni e di commenti che un utente può effettuare in un'ora. Questo dev'essere senz'altro gestito con l'utilizzo di trigger, tuttavia quest'operazione di controllo (che dev'essere eseguita ad ogni pubblicazione di un commento e di una recensione) può essere molto costosa, motivo per il quale può essere vantaggioso l'utilizzo di un dato derivato che per ogni utente memorizza il numero di pubblicazioni effettuate, e che viene azzerato per tutti gli utenti allo scattare di xx:00. Si studia di seguito il numero di operazioni di lettura e scrittura in entrambi i casi. È chiaro che, nel momento in cui si voglia aggiungere una recensione oppure un commento, è noto il codice dell'utente che la vuole pubblicare.

Tabella dei volumi:

Concetto	Tipo	Volume dati
Utente	E	400,000
Commento	E	1,450,000
Recensione	E	1,000,000

Tabella delle operazioni:

Operazione	Tipo	Frequenza
Aggiunta Recensione	I	80,000/G
Aggiunta Commento	I	120,000/G
Azzeramento contatore	I	24/G

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

L'ultima operazione è soltanto presente nel caso in cui sia memorizzato nel database il dato derivato, la cui frequenza giornaliera è esattamente 24, una volta per ogni ora (da 00:00 a 23:00).

Con l'utilizzo del dato derivato, ad ogni aggiunta occorre aggiornare il contatore del rispettivo utente che ha intenzione di pubblicarla, conseguentemente si ottengono i seguenti risultati:

Aggiunta Recensione:

Concetto	Accessi	Tipo
Recensione	1	S
Utente	1	L
Utente	1	S

(2*2 + 1) * 80,000 = 400,000/G

Aggiunta Commento:

Concetto	Accessi	Tipo
Commento	1	S
Utente	1	L
Utente	1	S

(2*2 + 1) * 120,000 = 650,000/G

Azzeramento Contatore:

Concetto	Accessi	Tipo
Utente	400,000	S

400,000*2 * 24 = 19,200,000/G

Si ha quindi un totale di 20,250,000/G.

Il numero di accessi di scrittura in Utente per l'azzeramento del contatore è pari a 400,000 ovvero pari al numero di utenti, e in generale aumenta con il passare del tempo, in un ambiente in continua crescita come quello di un social network.

Per quanto riguarda il caso in cui non si utilizzi il dato derivato, ad ogni aggiunta o rimozione occorre andare a contare tutte le pubblicazioni che ha effettuato l'utente (quindi recensioni + commenti), perciò:

Aggiunta Recensione:

Concetto	Accessi	Tipo
Recensione	1	S
Recensione	1,000,000/400,000	L
Commento	1,450,000/400,000	L

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

(1*2 + 1,000,000/400,000 + 1,450,000/400,000) * 80,000 = 650,000/G

Aggiunta Commento:

Concetto	Accessi	Tipo
Commento	1	S
Recensione	1,000,000/400,000	L
Commento	1,450,000/400,000	L

(1*2 + 1,000,000/400,000 + 1,450,000/400,000) * 80,000 = 650,000/G

Si ha quindi un totale di 1,300,000/G.

In quest'ultimo caso, in generale nelle due operazioni di inserimento, la prima operazione di scrittura viene in realtà omessa nel caso molto raro in cui l'utente abbia effettuato già 1,000 pubblicazioni nell'attuale ora. Le operazioni di lettura, che hanno lo scopo di contare il numero di recensioni e di commenti, vengono effettuate anche per ricavare l'ora di pubblicazione. Se le recensioni pubblicate dall'utente che hanno ora di pubblicazione maggiore di hh:00, dove hh è l'ora corrente, sono in numero maggiori di 1,000 allora a quel punto si impedisce la pubblicazione e di conseguenza la scrittura su Commento/Recensione.

In questo caso specifico si può osservare che il dato derivato semplifica le due operazioni di inserimento, tuttavia il costo di mantenimento dello stesso sorpassa notevolmente il risparmio che esso porta. Nel caso in cui il rapporto tra il volume dei commenti o delle recensioni e il volume degli utenti sia molto maggiore, si può ipotizzare che l'analisi possa portare a risultati differenti.

Schema logico complessivo

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)
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
    FK: Seguito REFERENCES Utente
    FK: Seguente REFERENCES Utente
Oggetto(CodO, Nome, Descrizione, Categoria)
    PK: CodO
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, Descrizione)
    PK: (CodM, CodMittente, CodDestinatario)
    FK: (CodM, CodMittente, CodDestinatario) REFERENCES Messaggio
    FK: CodR REFERENCES Recensione
CartaCredito(Numero, DScadenza, EnteEmittente)
    PK: Numero
CartaUtente(CodCarta, CodU)
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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(TRN, Annullata, Data, CodPiano, CodUtente,
NumeroCartaDiCredito)

PK: TRN

FK: CodPiano REFERENCES Piano

FK: CodUtente REFERENCES Utente

FK: NumeroCartaDiCredito REFERENCES CartaCredito

AK: (Data, CodPiano, CodUtente)

TransazioneManuale(TRN, Annullata, Data, CodMittente, CodDestinatario,
NumeroCartaDiCredito, Quantità)

PK: TRN

FK: CodMittente REFERENCES Utente

FK: CodDestinatario REFERENCES UtentePremium

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

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

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; 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; infine l'esclusività del tipo di un report e del campo "Altro" (la presenza di uno esclude la presenza dell'altro). I rimanenti verranno gestiti con appositi TRIGGER. Si suppone che le date immesse precedano o siano uguali alla data e timestamp attuali, in quanto risparmia la creazione e l'esecuzione di numerosi trigger, diminuendo di conseguenza il tempo di sviluppo e aumentando l'efficienza della base di dati, che non deve effettuare controlli superflui. Questo può essere garantito tramite l'utilizzo di apposite procedure, inoltre può esserci in gioco un meccanismo di permessi che impediscano l'inserimento di una data arbitraria, ma al contempo permettano l'aggiunta della attuale.

Query di creazione

Si hanno le seguenti query di creazione:

```
CREATE TABLE Moderatore(
	Email varchar(320) NOT NULL UNIQUE,
	CHECK(Email ~* '^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
	Username varchar(55) PRIMARY KEY,
	CHECK(Username !~* '.*[^A-Za-z0-9].*'),
	Nome varchar(55) NOT NULL,
	Cognome varchar(55) NOT NULL,
	Psw varchar(72) NOT NULL
);
CREATE TABLE Utente(
	Email varchar(320) NOT NULL UNIQUE,
	CHECK(Email ~* '^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
	Username varchar(55) PRIMARY KEY,
	CHECK(Username !~* '.*[^A-Za-z0-9].*'),
	Nome varchar(55) NOT NULL,
	Cognome varchar(55) NOT NULL,
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	IP char(15) NOT NULL,
	CHECK(IP ~ '^(\d{3}\.){3}\d{3}$'),
	Psw varchar(72) 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 date NOT NULL
);
CREATE TABLE Oggetto(
	CodO int PRIMARY KEY,
	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 timestamp NOT NULL,
	DataVisionePubblica timestamp,
	CHECK(DataVisionePubblica IS NULL OR DataVisionePubblica >= DataPubblicazione),
	Titolo varchar(155) NOT NULL,
	Valore int,
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	Descrizione varchar(1555) NOT NULL,
	CHECK(Valore IS NULL OR (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 date 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,
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	Testo varchar(1555),
	CodCRisposta int,
	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) NOT NULL,
	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) NOT NULL,
	Descrizione varchar(1555),
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	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,
	CodUtentePremium varchar(55) NOT NULL,
	Quantita int NOT NULL,
	Periodo periodoPiano NOT NULL,
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	Attivo bool NOT NULL,
	FOREIGN KEY (CodUtentePremium) REFERENCES UtentePremium(CodUtente)
);
CREATE TABLE Esclusivita(
	DataAnticipata timestamp 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,
	CHECK(DAbbandono IS NULL OR (DAbbandono >= DIscrizione)),
	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,
	CodMittente varchar(55) NOT NULL,
	NumeroCartaDiCredito char(16) NOT NULL,
	FOREIGN KEY (NumeroCartaDiCredito) REFERENCES CartaCredito(Numero),
	FOREIGN KEY (CodPiano) REFERENCES Piano(CodP),
	FOREIGN KEY (CodMittente) REFERENCES Utente(Username),
	UNIQUE(Data, CodPiano, CodMittente)
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
);
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,
	Quantita INTEGER NOT NULL,
	FOREIGN KEY (NumeroCartaDiCredito) REFERENCES CartaCredito(Numero),
	FOREIGN KEY (CodMittente) REFERENCES Utente(Username),
	FOREIGN KEY (CodDestinatario) REFERENCES UtentePremium(CodUtente),
	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 Moderatore(Username)
);
CREATE TABLE Rimozione(
	DataEffettuazione date,
	CodR int,
	DAnnullamento date,
	CHECK(DAnnullamento IS NULL OR (DataEffettuazione <= DAnnullamento)),
	CodModeratore varchar(55) NOT NULL,
	PRIMARY KEY(DataEffettuazione, CodR),
	FOREIGN KEY (CodR) REFERENCES Recensione(CodR),
	FOREIGN KEY (CodModeratore) REFERENCES Moderatore(Username)
);
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Viste

Vista contenente tutti i piani attivi:

```
CREATE VIEW PianiAttivi AS
SELECT *
FROM Piano
WHERE Attivo = TRUE;
```

Vista contenente tutte le recensioni visibili (controlla solamente che la recensione non sia stata rimossa da un Moderatore):

```
CREATE VIEW RecensioniVisibili AS
SELECT R1.*
FROM Recensione AS R1
WHERE R1.CodR NOT IN (
       SELECT R2.CodR
       FROM Recensione AS R2, Rimozione
       WHERE R2.CodR = Rimozione.CodR
         AND Rimozione.DAnnullamento IS NULL
);
```

Vista contenente tutte le recensioni disponibili per ciascun utente:

```
CREATE VIEW RecDisponibiliPerUtente AS
SELECT Utente.Username, R.CodR
FROM Utente, RecensioniVisibili AS R
WHERE R.DataVisionePubblica <= CURRENT_DATE OR
Utente.Username = R.CodUtente OR
EXISTS (
       SELECT *
       FROM UtentePremium, Iscrizione, Piano, Esclusivita
       WHERE UtentePremium.CodUtente = R.CodUtente AND
       Piano.CodUtentePremium = UtentePremium.CodUtente AND
       Iscrizione.CodUtente = Utente.Username AND
       Iscrizione.CodP = Piano.CodP AND
       Iscrizione.DAbbandono IS NULL AND
       Piano.CodP = Esclusivita.CodP AND
       Esclusivita.CodR = R.CodR AND
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	DataAnticipata <= CURRENT_DATE
);
```

Trigger – Imposizione di vincoli

Totalità della gerarchia di Transazione:

```
/* Il primo argomento è vero se si tenta di inserire una transazione manuale, falso altrimenti
*/
CREATE FUNCTION TotalitaGerarchiaTrans() RETURNS TRIGGER AS $$
DECLARE
	Cond BOOL;
BEGIN
	IF OLD.TRN = NEW.TRN THEN RETURN NEW;
	END IF;

	IF TG_NARGS <> 1 THEN RAISE EXCEPTION 'Argomenti forniti erroneamente';
	END IF;

	IF TG_ARGV[0] = 'true'
	THEN
		Cond = EXISTS(SELECT TRN
				FROM TransazioneAutomatica
				WHERE TRN = NEW.TRN);
	ELSE
		Cond = EXISTS(SELECT TRN
				FROM TransazioneManuale
				WHERE TRN = NEW.TRN);
	END IF;

	IF Cond
	THEN RAISE EXCEPTION 'Questo codice TRN e'' gia'' presente nel database.';
	END IF;
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneInserTransManuale
BEFORE INSERT OR UPDATE ON TransazioneManuale
FOR EACH ROW EXECUTE PROCEDURE TotalitaGerarchiaTrans(TRUE);

CREATE TRIGGER GestioneInserTransAuto
BEFORE INSERT OR UPDATE ON TransazioneAutomatica
FOR EACH ROW EXECUTE PROCEDURE TotalitaGerarchiaTrans(FALSE);
```

Totalità della gerarchia di Messaggio:

```
CREATE FUNCTION TotalitaGerarchiaMessaggio() RETURNS TRIGGER AS $$
BEGIN
	IF (OLD.CodM = NEW.CodM AND
		OLD.CodMittente = NEW.CodMittente AND
		OLD.CodDestinatario = NEW.CodDestinatario) THEN RETURN NEW;
	END IF;

	IF EXISTS (SELECT *
			FROM (SELECT CodM, CodMittente, CodDestinatario
				FROM MTesto
				UNION
				SELECT CodM, CodMittente, CodDestinatario
				FROM MImmagine
				UNION
				SELECT CodM, CodMittente, CodDestinatario
				FROM MRecensione) AS CodMessaggi
			WHERE CodM = NEW.CodM
			AND CodMittente = NEW.CodMittente
			AND CodDestinatario = NEW.CodDestinatario)
	THEN RAISE EXCEPTION 'La tripla ("%", "%", "%") esiste gia'' in un''altra
specializzazione di Messaggio', NEW.CodM, NEW.CodMittente, NEW.CodDestinatario;
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneInserMTesto
BEFORE INSERT OR UPDATE ON MTesto
FOR EACH ROW EXECUTE PROCEDURE TotalitaGerarchiaMessaggio();

CREATE TRIGGER GestioneInserMImmagine
BEFORE INSERT OR UPDATE ON MImmagine
FOR EACH ROW EXECUTE PROCEDURE TotalitaGerarchiaMessaggio();

CREATE TRIGGER GestioneInserMRecensione
BEFORE INSERT OR UPDATE ON MRecensione
FOR EACH ROW EXECUTE PROCEDURE TotalitaGerarchiaMessaggio();
```

Trigger che permette di imporre che un utente premium possa offrire esclusività soltanto si proprie recensioni:

```
CREATE FUNCTION VincoloEsclusivita() RETURNS TRIGGER AS $$
DECLARE
CodUtenteP varchar(55);
BEGIN
        IF (OLD.CodP = NEW.CodP AND
                OLD.CodR = NEW.CodR) THEN RETURN NEW;
        END IF;

        SELECT CodUtentePremium INTO CodUtenteP
        FROM Piano WHERE CodP = NEW.CodP;

        IF NOT EXISTS (SELECT *
                        FROM Recensione
                        WHERE CodR = NEW.CodR AND CodUtente = CodUtenteP)
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
        THEN RAISE EXCEPTION 'La recensione non appartiene all''utente in questione';
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneEsclusivita
BEFORE INSERT OR UPDATE ON Esclusivita
FOR EACH ROW EXECUTE PROCEDURE VincoloEsclusivita();
```

Trigger che impone che un utente possa effettuare una transazione solamente con una carta di credito di cui è in possesso:

```
CREATE FUNCTION VincoloCartaCredito() RETURNS TRIGGER AS $$
BEGIN
        IF (OLD.NumeroCartaDiCredito = NEW.NumeroCartaDiCredito AND
            OLD.CodMittente = NEW.CodMittente) THEN RETURN NEW;
        END IF;

        IF NOT EXISTS (SELECT *
                       FROM CartaUtente
                       WHERE CodU = NEW.CodMittente AND
                       NumeroC = NEW.NumeroCartaDiCredito)
        THEN RAISE EXCEPTION 'L''utente non può utilizzare la carta di credito in questione';
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneTransManCarta
BEFORE INSERT OR UPDATE ON TransazioneManuale
FOR EACH ROW EXECUTE PROCEDURE VincoloCartaCredito();
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
CREATE TRIGGER GestioneTransAutoCarta
BEFORE INSERT OR UPDATE ON TransazioneAutomatica
FOR EACH ROW EXECUTE PROCEDURE VincoloCartaCredito();
```

Trigger che vincola che non si possa rimuovere una recensione che attualmente si trova già nello stato disattivato:

```
CREATE FUNCTION VincoloRimozione() RETURNS TRIGGER AS $$
BEGIN
	IF OLD.DAnnullamento = NEW.DAnnullamento THEN RETURN NEW;
	END IF;

	IF EXISTS (SELECT *
		FROM Rimozione
		WHERE CodR = NEW.CodR AND DAnnullamento IS NULL)
	THEN RAISE EXCEPTION 'La recensione e'' gia'' attualmente invisibile';
	END IF;

	RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneRimozione
BEFORE INSERT OR UPDATE ON Rimozione
FOR EACH ROW EXECUTE PROCEDURE VincoloRimozione();
```

Trigger che vincola che non ci si possa iscrivere ad un piano a cui si è già attualmente abbonati:

```
CREATE FUNCTION VincoloIscrizione() RETURNS TRIGGER AS $$
BEGIN
	IF OLD.DAbbandono = NEW.DAbbandono THEN RETURN NEW;
	END IF;

	IF EXISTS (SELECT *
		FROM Iscrizione
		WHERE CodP = NEW.CodP AND
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
                CodUtente = NEW.CodUtente AND
                DAbbandono IS NULL)
        THEN RAISE EXCEPTION 'L''utente e'' gia'' attualmente iscritto al piano';
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneIscrizione
BEFORE INSERT OR UPDATE ON Iscrizione
FOR EACH ROW EXECUTE PROCEDURE VincoloIscrizione();
```

Vincolo che permette di imporre che la DataVisionePubblica di una recensione sia successiva rispetto alla data di visione anticipata che offre un determinato piano:

```
CREATE FUNCTION VincoloDataEsclusiva() RETURNS TRIGGER AS $$
BEGIN
        IF (OLD.DataAnticipata = NEW.DataAnticipata AND
            OLD.CodR = NEW.CodR) THEN RETURN NEW;
        END IF;

        IF EXISTS (SELECT *
                FROM Recensione
                WHERE DataVisionePubblica IS NOT NULL AND
                CodR = NEW.CodR AND NEW.DataAnticipata > DataVisionePubblica)
        THEN RAISE EXCEPTION 'La data anticipata e'' cronologicamente dopo la data di visione
pubblica';
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneDataEsclusiva
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
BEFORE INSERT OR UPDATE ON Esclusivita

FOR EACH ROW EXECUTE PROCEDURE VincoloDataEsclusiva();


CREATE FUNCTION VincoloDataVisionePubblica() RETURNS TRIGGER AS $$

BEGIN

	IF OLD.DataVisionePubblica = NEW.DataVisionePubblica THEN RETURN NEW;

	END IF;


	IF EXISTS (SELECT *

		FROM Esclusivita

		WHERE CodR = NEW.CodR AND NEW.DataVisionePubblica < DataAnticipata)

	THEN RAISE EXCEPTION 'La data di visione pubblica immessa precede una data di visione
anticipata';

	END IF;


	RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';


CREATE TRIGGER GestioneDataVisionePubblica

BEFORE INSERT OR UPDATE ON Recensione

FOR EACH ROW EXECUTE PROCEDURE VincoloDataVisionePubblica();
```

Imposizione del numero massimo di pubblicazioni orarie:

```
-- Si può utilizzare la stessa procedura per entrambi i trigger, in quanto hanno l'attributo
CodUtente con il medesimo nome

CREATE FUNCTION VincoloMaxPubblicazioni() RETURNS TRIGGER AS $$

DECLARE

    -- Contiene l'ora attuale priva di minuti e secondi

    OraNonMin timestamp;

BEGIN

    OraNonMin = date_trunc('hour', Now()::timestamp);


    IF (SELECT (SELECT COUNT(*)
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
                FROM Recensione
                WHERE CodUtente = NEW.CodUtente AND
                DataPubblicazione >= OraNonMin)
               +
               (SELECT COUNT(*)
                FROM Commento
                WHERE CodUtente = NEW.CodUtente AND
                Commento.Data >= OraNonMin)) > 1000
    THEN RAISE EXCEPTION 'E'' stato superato il numero massimo di pubblicazioni che si possono
effettuare in un''ora.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestioneMaxPubblicazioniRec
BEFORE INSERT ON Recensione
FOR EACH ROW EXECUTE PROCEDURE VincoloMaxPubblicazioni();

CREATE TRIGGER GestioneMaxPubblicazioniCom
BEFORE INSERT ON Commento
FOR EACH ROW EXECUTE PROCEDURE VincoloMaxPubblicazioni();
```

Trigger – Mantenimento dati derivati

Non sono presenti dati derivati all'interno della base di dati in quanto lo studio dei dati derivati ha portato alla conclusione che entrambi quelli considerati sono svantaggiosi.

Query di inserimento

Si hanno le seguenti query di inserimento:

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
INSERT INTO Moderatore
  (Email, Username, Nome, Cognome, Psw)
VALUES
  ('dodialessandro3@gmail.com', 'AlleDodi', 'Alessandro', 'Dodi', 'alleella314253'),
  ('amanjotSingh@gmail.com', 'jot', 'Amanjot', 'Singh', 'singh392015'),
  ('LeonardoTemperanza@gmail.com', 'lollo', 'Leonardo', 'Temperanza', 'leotemp126'),
  ('luigifini@gmail.com', 'luigi', 'Luigi', 'Fini', 'luigiluigi'),
  ('mariacostanzi@gmail.com', 'meri', 'Maria', 'Costanzi', 'MariaCostanzi24152');


INSERT INTO Utente
  (Email, Username, Nome, Cognome, Psw, IP, Foto)
VALUES
  ('sarafornaciari@gmail.com', 'sara', 'Sara', 'Fornaciari', 'passwordsicurasara',
'127.000.000.001', 'images/users/sara.png'),
  ('mariorossi@gmail.com', 'MarioRossi', 'Mario', 'Rossi', 'mariomario', '127.000.000.002',
'images/users/MarioRossi.png'),
  ('lauracastiello@gmail.com', 'lalla', 'Laura', 'Castiello', 'lauraCastiello1207',
'127.000.000.003', 'images/users/default.png'),
  ('jimmy@gmail.com', 'jimmy', 'Jimmy', 'Brown', 'lljimllbrownll', '127.000.000.004',
'images/users/jimmy.png'),
  ('groot@gmail.com', 'groot', 'Groot', 'Avenger', 'grootaven123', '127.000.000.005',
'images/users/default.png'),
  ('frankestrizza@gmail.com', 'frank', 'Frank', 'Salami', 'ffrraannkk122346',
'127.000.000.006', 'images/users/default.png'),
  ('ligabuemarcon@gmail.com', 'lmarco', 'Marco', 'Ligabue', 'ligabmarc', '127.000.000.007',
'images/users/default.png'),
  ('marcusspin@gmail.com', 'marcus', 'Marcus', 'Spin', 'spinmarcusspin', '127.000.000.008',
'images/users/default.png');


INSERT INTO UtentePremium
  (CodUtente, IBAN)
VALUES
  ('sara', 'IT45X0442834101000000654321'),
  ('lalla', 'IT60X0542811101000000123456'),
  ('groot', 'IT60X0542811101000000123453');


INSERT INTO Amicizia
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
  (Seguente, Seguito, Data)
VALUES
  ('sara', 'jimmy', '2022-01-09'),
  ('sara', 'lalla', '2022-02-04'),
  ('sara', 'groot', '2022-03-10'),
  ('marcus', 'marcus', '2022-03-21'),
  ('marcus', 'jimmy', '2021-04-07'),
  ('lalla', 'sara', '2021-12-02'),
  ('jimmy', 'sara', '2022-05-12'),
  ('jimmy', 'groot', '2022-05-12'),
  ('jimmy', 'frank', '2022-05-12'),
  ('jimmy', 'MarioRossi', '2022-05-12'),
  ('groot', 'sara', '2022-06-10'),
  ('groot', 'lmarco', '2022-06-10'),
  ('lmarco', 'sara', '2022-07-01');


INSERT INTO Oggetto
  (CodO, Nome, Descrizione, Categoria)
VALUES
  ('1', 'Interstellar', 'Un gruppo di scienziati appartenenti un tempo alla NASA, sfruttando
un "wormhole" per superare le limitazioni fisiche del viaggio spaziale e coprire le immense
distanze del viaggio interstellare, organizza una serie di missioni spaziali alla ricerca di
un pianeta abitabile.', 'Film'),
  ('2', 'Will hunting', 'In un quartiere povero di Boston, Will Hunting (Matt Damon), venti
anni, vive in modo precario e scombinato insieme ad alcuni amici, tra i quali spicca il suo
migliore amico Chuckie, e guadagna qualcosa pulendo i pavimenti nel dipartimento di matematica
del famoso Massachusetts Institute of Technology (MIT).', 'Film'),
  ('3', 'Cane', 'l cane è un quadrupede ed ha il corpo coperto di pelo. Le sue zampe sono
lunghe, forti, snelle, atte alla corsa. Il cane cammina sulla punta delle dita, come il gatto,
ma il suo passo non è silenzioso, perché i suoi artigli non sono retrattili e quando cammina
li batte sul suolo.', 'Animali'),
  ('4', 'Joe Rogan', 'The Joe Rogan Experience è un podcast ospitato dal comico, presentatore
e commentatore a colori UFC americano Joe Rogan. È stato lanciato il 24 dicembre 2009 su
YouTube da Rogan e dal comico Brian Redban, che ne è stato co-conduttore e produttore fino al
2013, quando è stato sostituito da Jamie Vernon.', 'Podcast'),
  ('5', 'The genius of the crowd', 'Famosa poesia di Charles Bukowski', 'Poesia'),
  ('6', 'Bitcoin website', '', 'Tecnologia');


INSERT INTO Recensione
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
  (CodR, Foto, DataPubblicazione, Titolo, Valore, Descrizione, CodUtente, CodO,
DataVisionePubblica)

VALUES

  ('1', 'images/reviews/lalla/1.png', '2021-04-01 10:12:24', 'Migliore film di sempre', '100',
'Semplicemente il miglior film di sempre', 'lalla', '1', '2021-05-01 05:01:10'),

  ('2', 'images/reviews/jimmy/2.png', '2021-04-03 02:52:26', 'Poesia estremamente profonda',
'100', 'Questa è la mia poesia preferita di Bukowski. La trovo molto profonda.', 'jimmy', '5',
'2021-04-03 15:15:15'),

  ('3', 'images/reviews/sara/3.png', '2022-05-13 05:42:23', 'Miglior film di Robin Williams',
'100', 'Film meraviglioso. A mio parare miglior prestazione di Robin Williams', 'sara', '2',
'2022-05-14 03:30:12'),

  ('4', 'images/reviews/jimmy/4.png', '2022-05-14 12:16:56', 'Il mio podcast preferito', '95',
'Questo è il mio podcast preferito. Joe è molto intelligente e fa ottime domande. Mi piace
molto la varietà nel tipo delle persone chiamate', 'jimmy', '4', '2022-05-14 18:12:30'),

  ('5', 'images/reviews/marcus/5.png', '2022-05-15 08:16:34', 'Comprate sul mio sito', '90',
'Comprate bitcoin sul mio sito comprabitcoin.com', 'marcus', '6', '2022-05-15 09:10:21'),

  ('6', 'images/reviews/sara/5.png', '2022-05-16 22:22:22', 'Imperdibile', '94', 'Sarò breve:
questo film entra di diritto nel club dei film unici da riguardare più e più volte. la storia,
almeno inizialmente, non appare imprevedibile e ricca di sorprese: Terra che sta diventando
inabitabile, quindi serve un altro mondo dove abitare, quindi ecco il progetto dell viaggio
interstellare. Ma ecco la rivoluzione: non ci si sofferma più nel mostrare le solite pompose
scene tipiche dei film spaziali (piloti che si preparano, musiche eroiche, sfrenata
dettagliatura delle navicelle spaziali), ma si vada dritti al cuore della narrazione. Alcuni
dei vari accadimenti nello spazio non sono di immediata comprensione per lo spettatore, quasi
una conseguenza logica vista la mole di nozioni tecniche su cui si basano: ma questo mi è
piaciuto, attriubuisce una certo realismo e una logica che sfuggono allo spettatore medio, ma
che vengono più che esaurientemente rese afferrabili col proseguire della trama.', 'sara',
'1', '2022-10-01 18:20:00'),

  ('7', 'images/reviews/lalla/5.png', '2022-05-17 23:13:42', 'Stravolgente', '100','Questo
film di psicologia ci mostra come non si debba mai dar per scontato un attore:robin
williams,solitamente effervescenza e vitalità allo stato puro-l''attimo fuggente,mrs
doubtfire,hook-qui è nei panni di uno psicologo,brillante e amante della vita ma non troppo al
di sopra delle righe-e sia detto che considero l''attimo fuggente come uno dei film capolavoro
della storia del cinema e keatting come il professore che avrei voluto avere-tuttavia il
rientrare nelle righe di williams non mi dispiace affatto,ed lo s9i ritrova solo in risvegli
con de niro-il cambiamento di williams è tale in questo flm che può essere paragonato a quello
di fabrizi in roma città aperta, nel quale da comico si trasforma in un serio rappresentante
di dio.', 'lalla', '2', null),

  ('8', 'images/reviews/groot/5.png', '2022-05-15 00:12:51', 'Bitcoin', '90', 'Sei vuoi capire
cosa sono i bitcoin, segui il link in allegato', 'groot', '6', '2022-05-15 10:00:00');


INSERT INTO Medaglia

  (CodUtente, CodR, Data)

VALUES

  ('lalla', '1', '2022-05-20'),
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
  ('marcus', '2', '2022-05-21'),
  ('lalla', '3', '2022-05-22'),
  ('marcus', '3', '2022-05-23'),
  ('jimmy', '5', '2022-05-24'),
  ('sara', '5', '2022-05-25');

INSERT INTO Report
  (CodUtente, CodR, TipoReport, Altro)
VALUES
  ('MarioRossi', '5', 'Spam o ingannevole', NULL),
  ('sara', '5', 'Spam o ingannevole', NULL),
  ('lalla', '8', 'Spam o ingannevole', NULL),
  ('jimmy', '5', 'Spam o ingannevole', NULL);

INSERT INTO Commento
  (CodC, CodR, CodUtente, Data, Testo, CodCRisposta)
VALUES
  ('1', '5', 'MarioRossi', '2022-06-01 21:00:13', 'Questa è spam, ti ho riportato', null),
  ('2', '4', 'marcus', '2022-06-02 08:00:32', 'Hai ragione', null),
  ('3', '3', 'lalla', '2022-06-03 05:25:53', 'Concordo', null),
  ('4', '4', 'lalla', '2022-06-03 18:41:37', 'la vedo diversamente', null);

INSERT INTO Messaggio
  (CodM, CodMittente, CodDestinatario, Letto, Data)
VALUES
  ('1', 'sara', 'jimmy', 'TRUE', '2022-06-10 10:02:36'),
  ('2', 'jimmy', 'sara', 'TRUE', '2022-06-10 10:02:56'),
  ('3', 'sara', 'jimmy', 'TRUE', '2022-06-10 10:03:10'),
  ('4', 'jimmy', 'sara', 'TRUE', '2022-06-10 10:04:41'),
  ('5', 'sara', 'jimmy', 'FALSE', '2022-06-10 10:04:59'),
  ('6', 'sara', 'jimmy', 'FALSE', '2022-06-10 10:05:59'),
  ('1', 'lalla', 'sara', 'FALSE', '2022-06-10 11:05:50'),
  ('1', 'marcus', 'lalla', 'TRUE', '2022-06-10 11:05:50'),
  ('2', 'lalla', 'marcus', 'TRUE', '2022-06-10 12:34:21'),
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
  ('3', 'marcus', 'lalla', 'TRUE', '2022-06-10 12:03:51'),
  ('4', 'lalla', 'marcus', 'TRUE', '2022-06-10 15:22:21'),
  ('1', 'groot', 'lmarco', 'TRUE', '2022-06-10 12:34:21'),
  ('2', 'lmarco', 'groot', 'TRUE', '2022-06-10 12:35:51'),
  ('3', 'groot', 'lmarco', 'FALSE', '2022-06-10 15:36:21');

INSERT INTO MTesto
  (CodM, CodMittente, CodDestinatario, Contenuto)
VALUES
  ('1', 'sara', 'jimmy', 'Ciao Jimmy'),
  ('2', 'jimmy', 'sara', 'Ciao Sara'),
  ('3', 'sara', 'jimmy', 'Come stai?'),
  ('4', 'jimmy', 'sara', 'Mai stato meglio, tu?'),
  ('5', 'sara', 'jimmy', 'Idem'),
  ('6', 'sara', 'jimmy', 'Usciamo mercoledí?'),
  ('1', 'lalla', 'sara', 'Ci vediamo oggi pomeriggio per studiare?'),
  ('1', 'groot', 'lmarco', 'Come é andata la sessione estiva?'),
  ('2', 'lmarco', 'groot', 'Bene. E a te?'),
  ('3', 'groot', 'lmarco', 'Anche a me. Vieni alla festa di compleanno di frank?');

INSERT INTO MImmagine
  (CodM, CodMittente, CodDestinatario, Immagine)
VALUES
  ('1', 'marcus', 'lalla', '/img/198271928312982.png'),
  ('2', 'lalla', 'marcus', '/img/091823012983179.png');

INSERT INTO MRecensione
  (CodM, CodMittente, CodDestinatario, codR)
VALUES
  ('3', 'marcus', 'lalla', '1'),
  ('4', 'lalla', 'marcus', '2');


INSERT INTO CartaCredito
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
  (Numero, DScadenza, EnteEmittente)
VALUES
  ('5365192748368000', '12/24', 'Mastercard'),
  ('5243906784563827', '11/23', 'Mastercard'),
  ('5328087472838500', '11/25', 'Mastercard'),
  ('5785097781567052', '01/25', 'Mastercard'),
  ('5785097781567051', '02/25', 'Visa'),
  ('5785097781560332', '03/25', 'Mastercard');

INSERT INTO CartaUtente
  (NumeroC, CodU)
VALUES
  ('5365192748368000', 'lalla'),
  ('5243906784563827', 'jimmy'),
  ('5328087472838500', 'MarioRossi'),
  ('5785097781567052', 'marcus'),
  ('5785097781567051', 'sara'),
  ('5785097781560332', 'sara');

INSERT INTO Piano
  (CodP, CodUtentePremium, Quantita, Periodo, Attivo)
VALUES
  ('1', 'sara', '1', 'trimestre', 'true'),
  ('2', 'lalla', '2', 'anno', 'true');

INSERT INTO Esclusivita
  (DataAnticipata, CodP, CodR)
VALUES
  ('2022-05-10 19:20:00', '1', '6'),
  ('2022-05-11 10:00:00', '2', '7');

INSERT INTO Iscrizione
  (CodP, CodUtente, DIscrizione, DAbbandono)
VALUES
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
  ('1', 'MarioRossi', '2022-03-11', '2022-06-08'),
  ('2', 'MarioRossi', '2022-04-21', '2022-07-13'),
  ('1', 'jimmy', '2022-04-22', '2022-06-11'),
  ('2', 'jimmy', '2022-05-05', '2022-05-30'),
  ('1', 'jimmy', '2022-07-26', null),
  ('2', 'jimmy', '2022-07-26', null);


INSERT INTO TransazioneAutomatica
  (TRN, Annullata, Data, CodPiano, CodMittente, NumeroCartaDiCredito)
VALUES
  ('KSBSJUP2130PNISUEHBJ98KNJLIT00', 'false', '2022-03-11', '1', 'MarioRossi',
'5328087472838500'),
  ('LKNSKJDO0921KJBKJSBD91JKSDIT00', 'false', '2022-04-21', '2','MarioRossi',
'5328087472838500');


INSERT INTO TransazioneManuale
  (TRN, Annullata, Data, CodMittente, CodDestinatario, NumeroCartaDiCredito, Quantita)
VALUES
  ('J98NKSDLKBDKASLB78912ASDJKIT00', 'false', '2022-06-1', 'jimmy', 'sara',
'5243906784563827', '20'),
  ('55FBQSFGLJZ4PEKPX2XWR59UCHYFY3', 'false', '2022-05-30', 'jimmy', 'groot',
'5243906784563827', '30'),
  ('J98NKSDLKBDKASLB78912ASDJKIT22', 'false', '2022-06-2', 'sara', 'lalla',
'5785097781567051', '20');


INSERT INTO Ban
  (CodUtente, CodModeratore)
VALUES
  ('marcus', 'AlleDodi');


INSERT INTO Rimozione
  (DataEffettuazione, CodR, DAnnullamento, CodModeratore)
VALUES
  ('2022-05-20', '5', null, 'AlleDodi'),
  ('2022-05-22', '8', '2022-05-23', 'lollo');
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Query di interrogazione

Si propongono alcune interrogazioni che possono essere utilizzate in condizioni pratiche:

Visualizzare tutti gli amici di “sara”:

```
SELECT *
FROM Amicizia
WHERE Seguente = 'sara';
```

Contare il numero di amici di “sara”:

```
SELECT COUNT(*)
FROM Amicizia
WHERE Seguente = 'sara';
```

Visualizzare tutti gli utenti che hanno come amica “sara”:

```
SELECT *
FROM Amicizia
WHERE Seguito = 'sara';
```

Contare tutti gli utenti che hanno come amica “sara”:

```
SELECT COUNT(*)
FROM Amicizia
WHERE Seguito = 'sara';
```

Visualizzare una chat fra due utenti:

```
SELECT
	Messaggio.CodMittente,
	Messaggio.CodDestinatario,
	Messaggio.Letto,
	Messaggio.Data,
	MTesto.Contenuto,
	MImmagine.Immagine,
	MImmagine.Descrizione,
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	MRecensione.CodR,
	MRecensione.Descrizione
FROM
	Messaggio
	FULL JOIN MTesto ON Messaggio.CodM = MTesto.CodM
		AND Messaggio.CodMittente = MTesto.CodMittente
		AND Messaggio.CodDestinatario = MTesto.CodDestinatario
	FULL JOIN MImmagine ON Messaggio.CodM = MImmagine.CodM
		AND Messaggio.CodMittente = MImmagine.CodMittente
		AND Messaggio.CodDestinatario = MImmagine.CodDestinatario
	FULL JOIN MRecensione ON Messaggio.CodM = MRecensione.CodM
		AND Messaggio.CodMittente = MRecensione.CodMittente
		AND Messaggio.CodDestinatario = MRecensione.CodDestinatario
WHERE (Messaggio.CodMittente = 'sara' AND
	Messaggio.CodDestinatario = 'jimmy') OR
      (Messaggio.CodMittente = 'jimmy' AND
	Messaggio.CodDestinatario = 'sara')
ORDER BY Messaggio.data DESC
```

Visualizzare tutti gli utenti della chat di “sara”:

```
SELECT DISTINCT Messaggio.CodDestinatario
FROM Messaggio
WHERE (Messaggio.CodMittente = 'sara')
UNION
SELECT DISTINCT Messaggio.CodMittente
FROM Messaggio
WHERE (Messaggio.CodDestinatario = 'sara')
```

Visualizzare le proprie carte di credito salvate:

```
SELECT
	CartaCredito.Numero,
	CartaCredito.DScadenza,
	CartaCredito.EnteEmittente,
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
        Utente.Nome,
        Utente.Cognome
FROM CartaCredito JOIN
     CartaUtente ON CartaCredito.NUmero = CartaUtente.NumeroC JOIN
     Utente ON CartaUtente.CodU = Utente.Username
WHERE
        Utente.Username = 'lalla'
```

Visualizzare lo storico dei propri abbonamenti (sia attivi che non):

```
SELECT
        Iscrizione.DIscrizione,
        Piano.CodUtentePremium,
        Piano.Periodo,
        Piano.Quantita
FROM
        Iscrizione
        JOIN Piano ON Iscrizione.CodP = Piano.CodP
WHERE
        Iscrizione.CodUtente = 'jimmy'
```

Visualizzare i propri abbonamenti attuali (sia attivi che non):

```
SELECT
        Iscrizione.DIscrizione,
        Piano.CodUtentePremium,
        Piano.Periodo,
        Piano.Quantita
FROM
        Iscrizione
        JOIN Piano ON Iscrizione.CodP = Piano.CodP
WHERE
        Iscrizione.CodUtente = 'jimmy'
        AND Iscrizione.DAbbandono = NULL
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Visualizzare tutte le recensioni nel feed:

```
SELECT *
FROM RecDisponibiliPerUtente
WHERE RecDisponibiliPerUtente.Username = '?'
```

Visualizzare le notifiche dei messaggi non letti:

```
SELECT CodMittente
FROM Messaggio
WHERE CodDestinatario = 'sara' AND Letto = 'FALSE'
ORDER BY Data DESC
```

Visualizzare tutte le recensioni pubblicate da un utente:

```
SELECT *
FROM RecDisponibiliPerUtente, RecensioniVisibili
WHERE RecDisponibiliPerUtente.CodR = RecensioniVisibili.CodR AND
RecDisponibiliPerUtente.Username = '?' AND
RecensioniVisibili.CodUtente = '?'
```

Elencare tutte le transazioni effettuate da un utente:

```
SELECT TRN, Annullata, Data, NumeroCartaDiCredito, Piano.Quantita
FROM TransazioneAutomatica, Piano
WHERE TransazioneAutomatica.CodPiano = Piano.CodP AND
TransazioneAutomatica.CodMittente = ?
UNION
SELECT TRN, Annullata, Data, NumeroCartaDiCredito, Quantita
FROM TransazioneManuale
WHERE TransazioneManuale.CodMittente = ?
```

Totale transazioni ricevute di un determinato utente premium:

```
SELECT SUM(Total)
FROM ( SELECT SUM(Quantita) AS Total
       FROM TransazioneManuale
       WHERE CodDestinatario = '?'
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
	UNION
	SELECT SUM(Piano.Quantita) AS Total
	FROM TransazioneAutomatica, Piano
	WHERE TransazioneAutomatica.CodPiano = Piano.CodP AND
	Piano.CodUtentePremium = '?'
)
```

Calcolo rapporto (numero moderatori) / (numero utenti):

```
SELECT (
	SELECT COUNT(*)
	FROM Moderatore
) / (
	SELECT COUNT(*)
	FROM Utente
)
```

Elencare tutti gli utenti che possono visualizzare tutte le recensioni della categoria “Film” pubblicate dall’utente con username “lalla”:

```
SELECT U1.*
FROM Utente AS U1
WHERE NOT EXISTS (
	SELECT *
	FROM Recensione AS R2, Oggetto AS O2
	WHERE R2.CodUtente = 'lalla' AND
	R2.CodO = O2.CodO AND
	O2.Categoria = 'Film' AND
	NOT EXISTS (
		SELECT *
		FROM RecDisponibiliPerUtente AS RDPU
		WHERE RDPU.Username = U1.Username AND
		RDPU.CodR = R2.CodR
));
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Queste due ultime interrogazioni non sono presenti nel progetto Java in quanto si ritiene che siano più a scopo didattico, piuttosto che essere utilizzate frequentemente in un progetto reale.

Query di modifica

Modifica del messaggio 6 inviato da “sara” a “jimmy”:

```
UPDATE MTesto
SET Contenuto = 'Usciamo sabato?'
WHERE CodM = '6' AND CodMittente = 'sara'
    AND CodDestinatario = 'jimmy';
```

Modifica della data di visione di modifica di tutte le recensioni scritte da “sara”:

```
UPDATE Recensione
SET DataVisionePubblica = '2022-06-30 10:10:10'
WHERE CodUtente = 'sara'
```

Modifica dell’IBAN dell’utente premium “groot”:

```
UPDATE UtentePremium
SET IBAN = 'IT60X0542811101000093108800'
WHERE CodUtente = 'groot'
```

Query di eliminazione

Rimozione del like dato dall’utente “lalla” alla recensione con codice 1:

```
DELETE FROM Medaglia WHERE CodUtente = 'lalla' and CodR = '1'
```

Rimozione del commento con codice 1 della recensione con codice 4:

```
DELETE FROM Commento WHERE CodC = '1' and CodR = '4'
```

Rimozione del follow all’utente “frank” da parte dell’utente “jimmy”:

```
DELETE FROM Amicizia WHERE Seguente = 'jimmy' and Seguito = 'frank'
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Progettazione fisica

Si vuole effettuare l'analisi del costo delle query, per comprendere se conviene o no l'utilizzo di un indice. In seguito verranno riportati alcuni esempi di casi di studio.

Un primo caso di studio può essere rappresentato dalla seguente query: dato un utente, visualizzarne gli abbonamenti attuali (sia attivi che non).

```
SELECT Iscrizione.DIscrizione, Piano.CodUtentePremium, Piano.Periodo, Piano.Quantita
FROM Iscrizione, Piano
WHERE Iscrizione.CodP = Piano.CodP AND
      Iscrizione.CodUtente = 'jimmy' AND
      Iscrizione.DAbbandono = NULL
```

Si supponga di avere i seguenti dati relativi alle tabelle:

Tabella	NT	NB
Iscrizione	100000	10000
Piano	20000	1000

Si vogliono studiare i seguenti indici e si considera che siano tutti unclustered.

Indice	NF	NK
Iscrizione.CodP	80	20000
Iscrizione.CodUtente	50	6000
Piano.CodP	60	20000

Si effettua l'analisi della selettività degli indici e degli predicati rimanenti

$F_{iscrizione.CodP} = 1/20000$

$F_{iscrizione.CodUtente} = 1/6000$

$F_{piano.CodP} = 1/20000$

$F_{piano.Dabbondono} = 1/10$

Siccome all'interno della query è presente un join occorre suddividere l'analisi del costo in due casi, andando ad analizzare come varia il costo della query al variare della direzione di join.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Caso 1: Iscrizione → Piano

Iscrizione

C_{seq} =10000

$C_{iscrizione.CodUtente}$ = ceil(50/6000) + ceil(10000/6000) = 3

$E_{iscrizione}$ = ceil(100000/(50 * 6000) = 4

Piano

C_{seq} = 1000

$C_{piano.CodP}$ = ceil(60/20000) + ceil(20000/20000) = 2

CostTot = 3 + 4 * 2 = 11

Caso 2: Piano → Iscrizione

Piano

C_{seq} = 1000

E_{piano} = ceil(20000 / 10) = 2000

Iscrizione

C_{seq} = 10000

$C_{iscrizione.CodUtente}$ = 3

$C_{iscrizione.CodP}$ = ceil(80/20000) + ceil(100000/20000) = 7

CostTot = 1000 + 2000 * 3 = 7000

In questo primo caso di studio possiamo concludere che per accelerare la query occorre considerare il caso Iscrizione → Piano e creare gli indici su Iscrizione.CodUtente e Piano.CodP.

Un secondo caso di studio può essere rappresentato dalla query che, dato un utente, restituisce tutti i messaggi che non ha letto.

```
SELECT CodMittente
FROM Messaggio
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
WHERE CodDestinatario = 'sara' AND Letto = 'FALSE'
```

Si supponga di avere i seguenti dati relativi alla tabella Messaggio:

Tabella	NT	NB
Messaggio	1000000	40000

Si vuole studiare il seguente indice unclustered.

Indice	NF	NK
CodDestinatario	150	20000

Si considera che il predicato **Letto = 'FALSE'** abbia una selettività pari a 1/10. Dall'analisi della selettività dell'indice si ottiene che $F_{CodDestinatario}$ è pari a 1/20000.

Si può procedere con l'analisi del costo della query:

C_{seq} = 40000

$C_{CodDestinatario}$ = ceil(150/20000) + ceil(1000000/20000) = 51

E = ceil(100000/(20000*10)) = 5

Infine, si può concludere che conviene la creazione dell'indice CodDestinatario e che il numero di tuple attese è pari a 5.

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Analisi della sicurezza

Un aspetto importante della progettazione fisica è riguardante alcune considerazioni sulla sicurezza del database e lo studio dei permessi da attribuire ad un utente che non sia amministratore.

L'elemento critico del sistema informativo che si vuole studiare è l'autenticazione dell'utente, che permette di fruire dei servizi offerti dal social network. Nel database in questione vengono memorizzati inoltre alcune informazioni sensibili riguardanti all'utente, ad esempio la carta di credito. Tali informazioni possono essere rese inaccessibili se si progetta un adeguato meccanismo di autenticazione al sistema.

Per l'autenticazione di un utente, occorre memorizzare nel sistema le credenziali di accesso, che sono lo username e la password. Quest'ultima non puó essere memorizzata in chiaro nel database, in quanto permetterebbe l'accesso all'account di un utente nel caso in cui vi sia una fuga di dati. Il metodo che si vuole adottare è quello di poter cifrare la password e memorizzarne nel database la versione cifrata.

Per poter utilizzare le funzionalità di cifratura fornite dal DBMS POSTGRESQL, occorre abilitare l'estensione pgcrypto.

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;
```

A fronte delle analisi effettuate si è deciso di creare un trigger, che permette di sostituire la password con la sua versione cifrata a seguito dell'aggiornamento delle credenziali memorizzate nella tabella 'utente'. È stata prevista inoltre una procedura che permette di convalidare le credenziali di un utente. Le precedenti funzionalità sono state implementate anche per il moderatore.

```
CREATE FUNCTION moderatore_login(IN usr VARCHAR(55), IN pswd VARCHAR(72)) RETURNS BOOLEAN AS $$
    DECLARE
        successful BOOLEAN;
    BEGIN
        SELECT (psw = crypt(pswd, psw)) INTO successful
        FROM moderatore
        WHERE username = usr;

        RETURN successful;
    END;
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
$$ LANGUAGE 'plpgsql';

CREATE FUNCTION user_login(IN usr VARCHAR(55), IN pswd VARCHAR(72)) RETURNS BOOLEAN AS $$
    DECLARE
        successful BOOLEAN;
    BEGIN
        SELECT (psw = crypt(pswd, psw)) INTO successful
        FROM utente
        WHERE username = usr;

        RETURN successful;
    END;
$$ LANGUAGE 'plpgsql';

CREATE FUNCTION GestionePassword() RETURNS TRIGGER AS $$
BEGIN
       IF OLD.psw = NEW.psw THEN RETURN NEW;
       END IF;

       NEW.psw := crypt(NEW.psw, gen_salt('bf'));

       RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER GestionePasswordUtente
BEFORE INSERT OR UPDATE ON utente
FOR EACH ROW EXECUTE PROCEDURE GestionePassword();

CREATE TRIGGER GestionePasswordModeratore
BEFORE INSERT OR UPDATE ON moderatore
FOR EACH ROW EXECUTE PROCEDURE GestionePassword();
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

Effettuando un'analisi più approfondita, sono state individuate delle criticità nelle operazioni di eliminazione di un utente, e di modifica di password. Tali operazioni devono essere protette da un meccanismo di autenticazione, per la verifica dell'identità di un utente.

Innanzitutto, occorre creare un nuovo utente del database, che a differenza dell'amministratore sará in possesso di permessi diversi.

```
CREATE USER client WITH ENCRYPTED PASSWORD 'topolino2022';
GRANT ALL ON ALL TABLES IN SCHEMA "public" TO client;
```

A tale utente si decide di assegnare i permessi di SELECT, INSERT, e UPDATE sulla tabella utente. In particolare, sulle colonne 'username' e 'psw' il nuovo utente avrà solo i permessi di SELECT e di INSERT. In questo modo si impedisce l'eliminazione di un utente oppure il cambiamento di una password. Tali operazioni potranno essere eseguite tramite procedure, le quali prima di eseguire le operazioni in questione effettuano un controllo di autenticazione. Dato che l'utente del database 'client' non può effettuare l'eliminazione di una tupla della tabella utente oppure aggiornare la password, tali procedure devono poter essere eseguite con i permessi dell'utente che le ha create (ciò si ottiene mediante la direttiva SECURITY DEFINER).

```
GRANT SELECT, INSERT, UPDATE ON utente to client;

GRANT SELECT(email, username, nome, cognome, ip, foto, psw),
      INSERT(email, username, nome, cognome, ip, foto, psw),
      UPDATE(email, nome, cognome, ip, foto) ON utente to client;

CREATE FUNCTION delete_user(IN usr VARCHAR(55), IN pswd VARCHAR(72)) RETURNS BOOLEAN SECURITY
DEFINER AS $$
        BEGIN
            IF NOT (SELECT user_login(usr, pswd)) = TRUE
            THEN
                RETURN FALSE;
            END IF;

            DELETE FROM utente WHERE username = usr;
            RETURN TRUE;
        END
$$ LANGUAGE 'plpgsql';
```

Leonardo Temperanza – 152831
Amanjot Singh – 152792
Alessandro Dodi – 153355

```
CREATE FUNCTION change_psw(IN usr VARCHAR(55), IN new_pswd VARCHAR(72), IN old_pswd
VARCHAR(72))
    RETURNS BOOLEAN SECURITY DEFINER AS $$
        BEGIN
            IF NOT (SELECT user_login(usr, old_pswd)) = TRUE
            THEN
                RETURN FALSE;
            END IF;

            UPDATE utente SET psw = new_pswd WHERE psw = crypt(old_pswd, psw);
            RETURN TRUE;
        END
$$ LANGUAGE 'plpgsql';
```

Un altro aspetto importante riguarda la figura del moderatore. L'inserimento di un moderatore all'interno del social network, e in questo caso nel database, deve poter essere effettuato solo dall'utente amministratore, in quanto i moderatori sono delle persone su cui si fa affidamento per far rispettare le regole del social network, e inoltre deve poter avere accesso a tutte le segnalazioni e recensioni pubblicate nel database. Per queste ragioni occorre regolamentare la creazione degli utenti moderatori, in quanto l'account moderatore può essere usato in modo non convenzionale. Perciò, l'utente 'client' avrà solamente il permesso di SELECT sulla tabella moderatore.

```
GRANT SELECT ON moderatore to client;
```

L'utente 'client' sará l'utente che verrá utilizzato dall'applicazione o sito web dell'applicazione per connettersi al database.