Basi di Dati

Progetto A.A. 2024/2025

BACHECA ELETTRONICA DI ANNUNCI

Matricola 0307370

Nicolas Oberi

Indice

1. Descrizione del Minimondo

Si vuole realizzare un sistema informativo per la gestione di una bacheca elettronica di annunci. Tale bacheca permette agli utenti del sistema di inserire annunci per la vendita di materiale usato, di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell'oggetto, o di inserire domande (in maniera pubblica) sull'oggetto.

Un utente del sistema si registra scegliendo un username univoco, inserendo tutte le sue informazioni anagrafiche, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di recapiti (telefono, cellulare, email) indicandone uno come mezzo di comunicazione preferito.

I gestori del servizio possono creare una gerarchia di categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell'oggetto. Quando un oggetto inserito in bacheca è stato venduto, l'utente lo indica come tale e questo non viene più visualizzato nella bacheca pubblica.

Un utente del sistema, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all'utente che ha inserito l'annuncio. Similmente, un utente può "seguire" uno degli annunci, venendo così informato ogni volta che su questo viene effettuata una modifica (ad esempio, viene inserita una nuova nota).

I gestori del servizio possono generare un report indicante in forma aggregata per ciascun utente del sistema quale percentuale di annunci pubblicati sono stati contrassegnati come venduti.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
2,5,13, 18	Utente del sistema	Utente	Un utente è ovvio che sia un utente del sistema, perciò "del sistema" è ridondante.
6	Informazioni anagrafiche	Nome, cognome, data di nascita, CF	A valle del contatto con il cliente si è capito che il sistema deve memorizzare esattamente queste informazioni.
3	Materiale usato	Oggetti usati	Coerenza con il resto del testo.
4	Oggetto	Annuncio relativo all'oggetto	La domanda (pubblica) viene inserita sull'annuncio relativo all'oggetto.
7,8	Recapiti	Metodi di contatto	Semanticamente la parola "recapito" può anche assumere il significato di "indirizzo", il che crea confusione in questo caso, essendo l'indirizzo parte delle informazioni memorizzate sull'utente.
11	Lo indica	Indica l'annuncio relativo all'oggetto	Non è l'oggetto ad essere contrassegnato come venduto ma l'annuncio relativo ad esso.
15	"Seguire"	Attivare le notifiche	"Seguire" un oggetto e venire informati quando vengono effettuate modifiche su di esso equivale ad attivare le notifiche per quell'oggetto.
16	Nota	Commento	Con nota si intende commento pubblico.

Specifica disambiguata

Si vuole realizzare un sistema informativo per la gestione di una bacheca elettronica di annunci. Tale bacheca permette agli utenti di inserire annunci per la vendita di oggetti usati, di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell'oggetto, o di inserire domande (in maniera pubblica) sull'annuncio relativo all'oggetto.

Un utente si registra scegliendo un username univoco, inserendo nome, cognome, data di nascita, codice fiscale, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di metodi di contatto (telefono, cellulare, email) indicandone uno come preferito.

I gestori del servizio possono creare una gerarchia di categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell'oggetto. Quando un oggetto inserito in bacheca è stato venduto, l'utente che ha venduto l'oggetto indica l'annuncio relativo all'oggetto come tale e questo non viene più visualizzato nella bacheca pubblica.

Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all'utente che ha pubblicato l'annuncio. Similmente, un utente può

attivare le notifiche per uno degli annunci, venendo così informato ogni volta che su questo viene effettuata una modifica (ad esempio, viene inserito un nuovo commento).
I gestori del servizio possono generare un report indicante in forma aggregata per ciascun utente quale percentuale di annunci pubblicati sono stati contrassegnati come venduti.

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Utente	Persona che si registra autonomamente nel sistema. Può sia pubblicare annunci per vendere i suoi oggetti che visionare gli annunci pubblicati dagli altri utenti per comprare oggetti. Può interagire con gli altri utenti tramite messaggi privati o scrivendo commenti pubblici sugli annunci.		Annuncio
Annuncio	Viene pubblicato da un utente e al suo interno sono specificati l'oggetto in vendita e una descrizione per esso.		Utente
Messaggio privato	Stringa di testo, relativa ad un annuncio, scambiata privatamente tra due utenti.		Utente, Annuncio
Commento pubblico	Stringa di testo, relativa ad un annuncio, visibile pubblicamente a tutti gli utenti.	Domanda (in maniera pubblica)	Utente, Annuncio
Categoria	Categoria di appartenenza di un annuncio		Annuncio
Metodo di contatto	Metodo di contatto fornito dall'utente.	Email, telefono, cellulare.	Utente

Raggruppamento dei requisiti in insiemi omogenei

Frasi relative a Utente
[La] bacheca permette agli utenti di inserire annunci per la vendita di oggetti usati, di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell'oggetto, o di inserire domande (in maniera pubblica) sull'annuncio relativo all'oggetto. Un utente si registra scegliendo un username univoco, inserendo nome, cognome, data di nascita, codice fiscale, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di metodi di contatto (telefono, cellulare, email) indicandone uno come preferito. Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell'oggetto. Quando un oggetto inserito in bacheca è stato venduto, l'utente che ha inserito l'annuncio indica l'annuncio relativo all'oggetto come tale e questo non viene più visualizzato nella bacheca pubblica. Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all'utente che ha inserito l'annuncio. Similmente, un utente può attivare le notifiche per uno degli annunci, venendo così informato ogni volta che su questo viene effettuata una modifica (ad esempio, viene inserita una nuova nota). I gestori del servizio possono generare un report indicante in forma aggregata per ciascun utente quale percentuale di annunci pubblicati sono stati contrassegnati come venduti.

Frasi relative a Annuncio
[La] bacheca permette agli utenti di inserire annunci per la vendita di oggetti usati, di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell'oggetto, o di inserire domande (in maniera pubblica) sull'annuncio relativo all'oggetto. I gestori del servizio possono creare una gerarchia di categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell'oggetto. Quando un oggetto inserito in bacheca è stato venduto, l'utente che ha inserito l'annuncio indica l'annuncio relativo all'oggetto come tale e questo non viene più visualizzato nella bacheca pubblica. Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico […]. Un utente può attivare le notifiche per uno degli annunci, venendo così informato ogni volta che su questo viene effettuata una modifica (ad esempio, viene inserita una nuova nota). I gestori del servizio possono generare un report indicante in forma aggregata per ciascun utente quale percentuale di annunci pubblicati sono stati contrassegnati come venduti.

Frasi relative a Messaggio privato
[La] bacheca permette agli utenti […] di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell'oggetto. Un utente una volta letto e scelto un annuncio può decidere di […] inviare un messaggio privato all'utente che ha pubblicato l'annuncio.

Frasi relative a Commento pubblico
[La] bacheca permette agli utenti [...] di inserire domande (in maniera pubblica) sull'annuncio relativo all'oggetto. Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico. […] Ogni volta che su questo [Annuncio] viene effettuata una modifica (ad esempio, viene inserito un nuovo commento).

Frasi relative a Categoria

I gestori del servizio possono creare una gerarchia di categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria […].

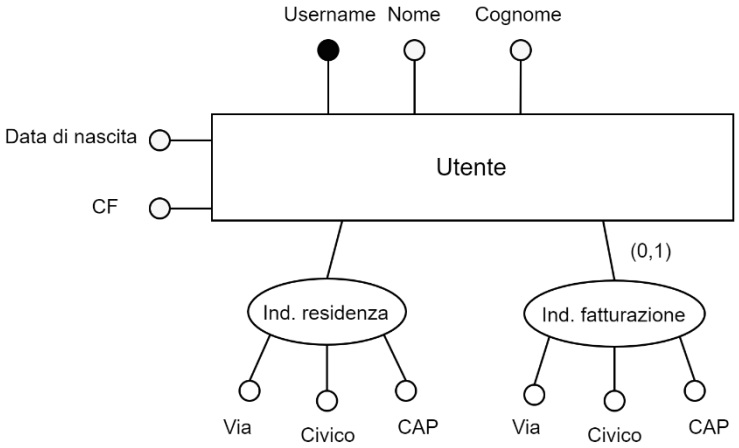
Frasi relative a Metodo di contatto
Un utente si registra […] indicando […] un numero arbitrario di metodi di contatto (telefono, cellulare, email) indicandone uno come preferito.

3. Progettazione concettuale

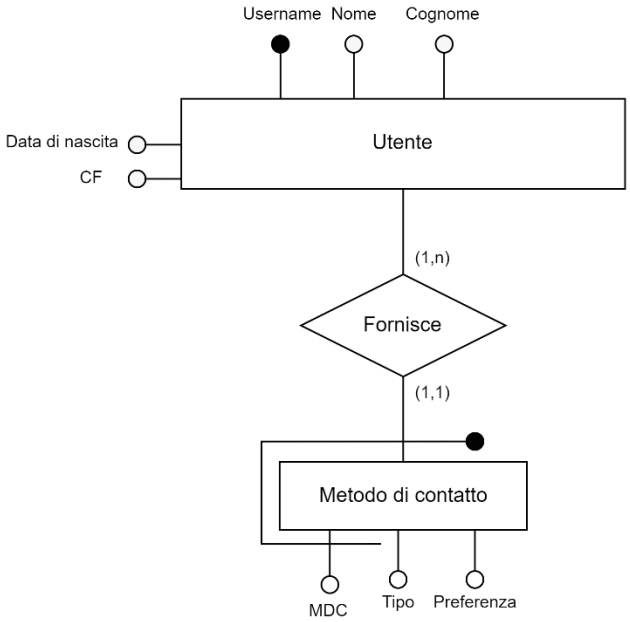
Costruzione dello schema E-R

La costruzione dello schema E-R parte dal concetto di *Utente*, modellato come entità dato il suo ruolo cardine nel sistema, a cui vengono aggiunti gli attributi richiesti nelle specifiche.

È opportuno specificare che anche il CF sarebbe stato valido come identificatore nel caso una stessa persona potesse creare un solo account utente, tuttavia dato che questa limitazione non è esplicitata nelle specifiche e che l'username è più appropriato a individuare un utente nel dominio del sistema, si è scelto l'username come identificatore. Si sceglie di modellare gli indirizzi come due attributi composti e non come entità a sé stanti in quanto non interessanti nel dominio dell'applicazione.

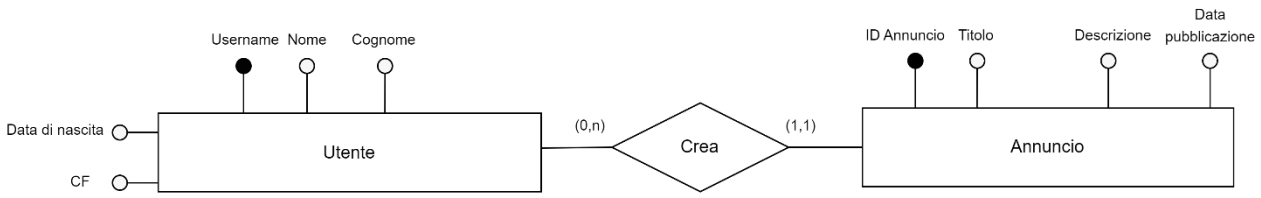


L'utente può fornire un numero arbitrario di metodi di contatto (telefono, cellulare, email) perciò si procede con la modellazione di questa entità che sarà identificata dal metodo di contatto stesso insieme all'username dell'utente. Avrà inoltre un attributo Tipo per distinguere tra telefono, cellulare ed email e un attributo Preferenza attraverso cui l'utente indicherà il metodo di contatto preferito. Viene introdotta una regola aziendale per assicurarsi che un utente abbia un unico metodo di contatto indicato come preferito. La porzione di schema risultante è la seguente.



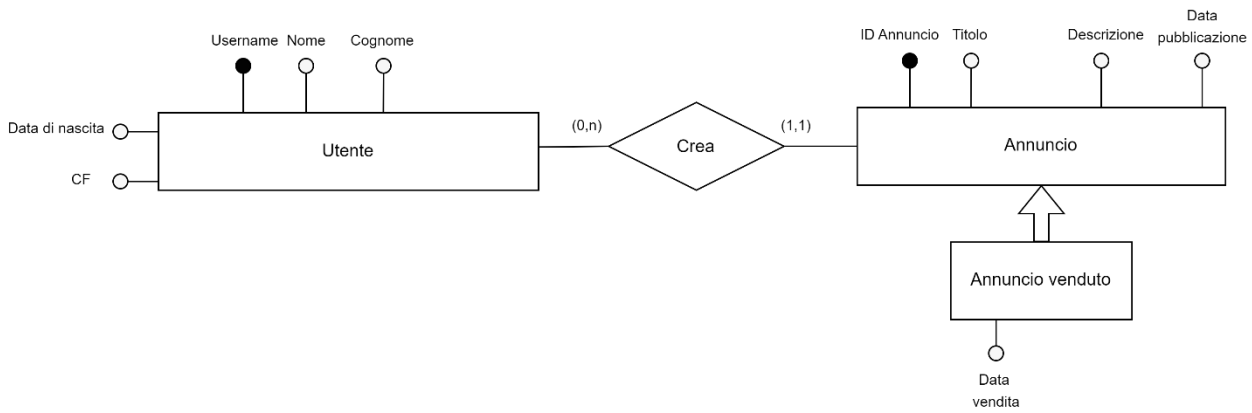
Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Si prosegue modellando l'entità *Annuncio*. L'attributo Titolo non è espressamente richiesto nelle specifiche ma a valle di un'interazione col cliente si è deciso di aggiungerlo per maggiore chiarezza. Si sceglie inoltre di evitare la modellazione di una possibile entità *"Oggetto"* in quanto poco rilevante ai fini del sistema, per cui gli attributi Titolo e Descrizione dell'entità *Annuncio* risultano più che sufficienti. Si utilizza come identificatore un ID sintetico in quanto altrimenti non sarebbe possibile identificare un annuncio univocamente. Si evidenzia comunque che un *Annuncio* non potrebbe esistere senza un *Utente* che lo crea.



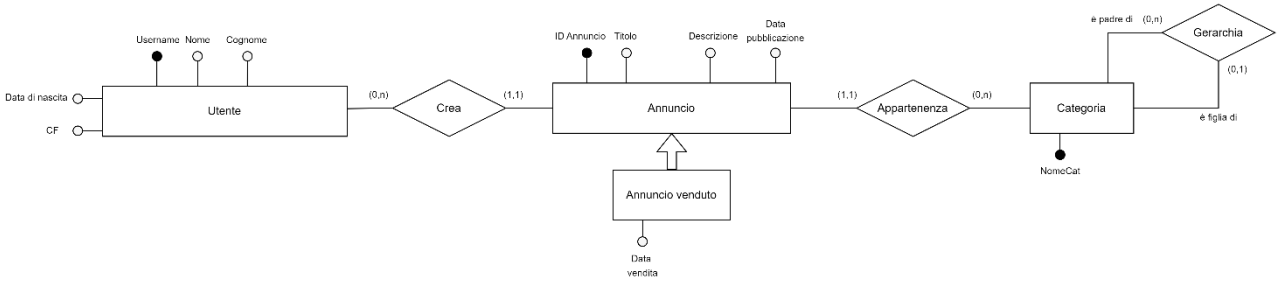
Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Dalle specifiche si evince la necessità di contrassegnare un annuncio come "venduto" quando l'oggetto relativo all'annuncio viene appunto venduto. Si utilizza quindi una generalizzazione parziale sull'entità *Annuncio*, per distinguere gli annunci contrassegnati come venduti da quelli ancora attivi.



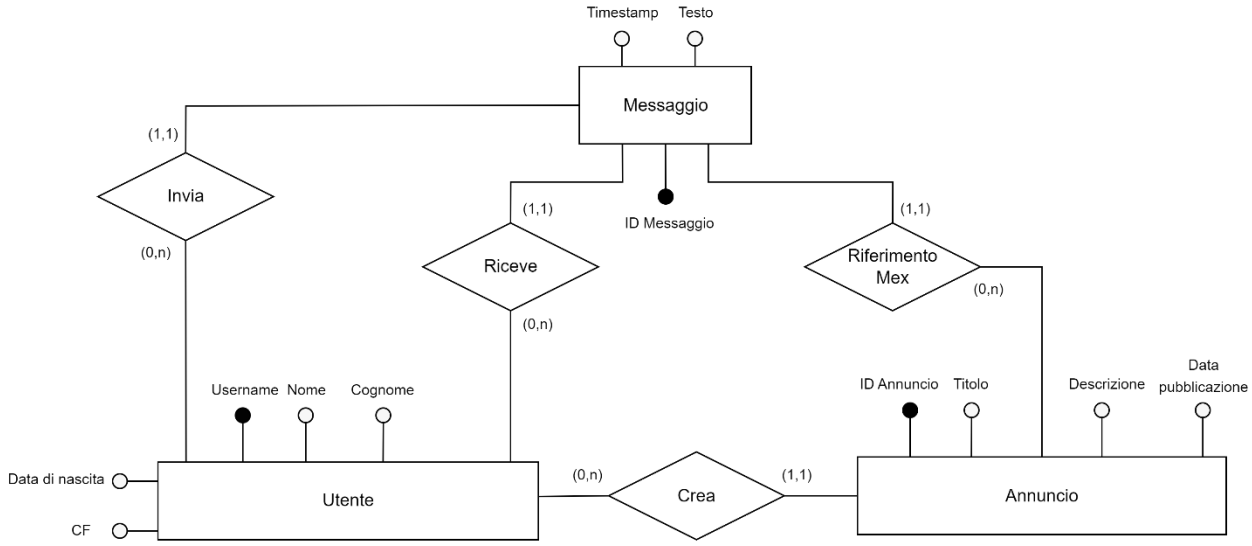
Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Si prosegue implementando la gerarchia di categorie come richiesto nelle specifiche, giungendo al seguente risultato. Si specifica che le categorie sono organizzate in stile "breadcrumbs".



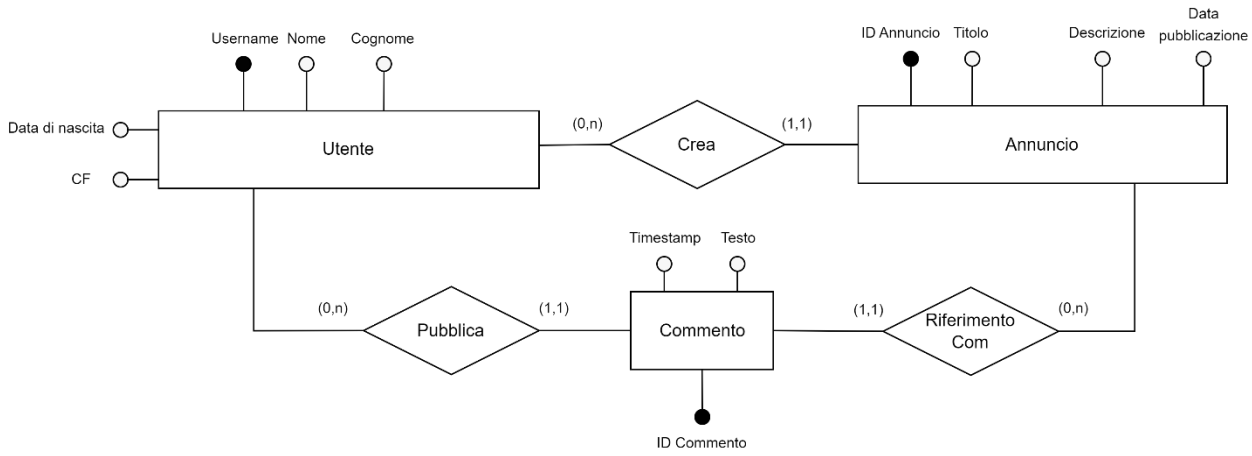
Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Si passa poi alla modellazione del concetto di *Messaggio privato*, che per semplicità viene rinominato *Messaggio*. Esso è identificato da un ID e possiede come attributi la stringa di testo scambiata e il timestamp in cui è stato inviato. L'entità *Messaggio* ha inoltre due associazioni con l'entità *Utente* per individuare mittente e destinatario e una con l'entità *Annuncio* per individuare l'annuncio al quale si riferisce. Si specifica anche qui che si è consapevoli che un messaggio non può esistere senza un mittente, un destinatario e un annuncio, e che quindi idealmente sarebbe stato opportuno modellarlo come entità debole, tuttavia l'alternativa all'ID sarebbe stata l'uso del timestamp, che non è però un identificatore affidabile in quanto è poco probabile che un mittente invii un messaggio, relativo allo stesso annuncio, allo stesso destinatario, nello stesso istante, ma non impossibile. Si aggiunge inoltre una regola aziendale per imporre che l'utente mittente sia diverso dall'utente destinatario. Il risultato è la seguente porzione di schema.



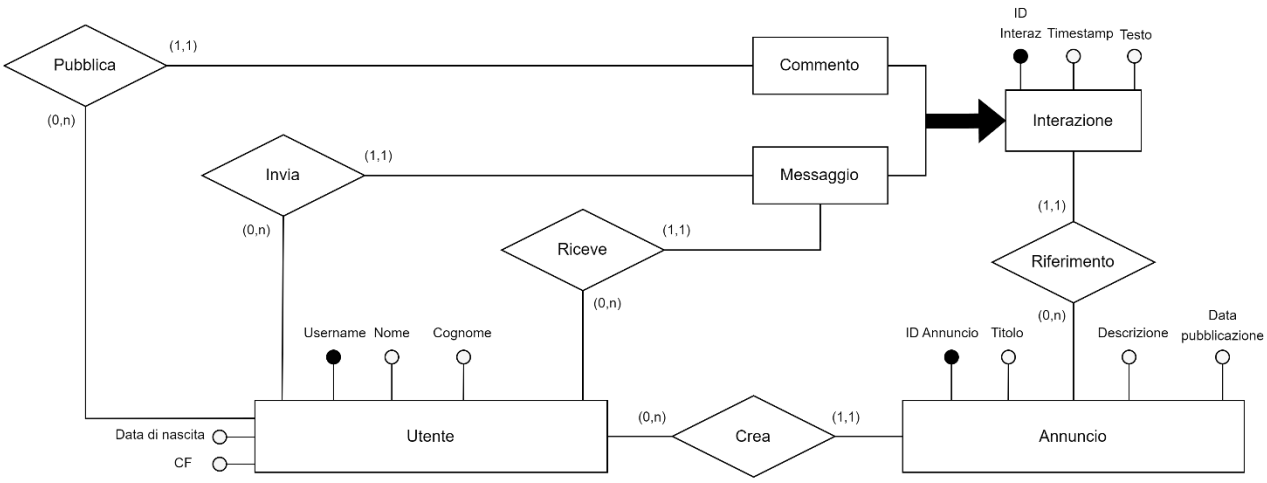
Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Si passa ora alla modellazione dell'entità *Commento pubblico* che per semplicità verrà chiamata *Commento* all'interno dello schema. Un commento viene pubblicato sulla pagina dell'annuncio ed è visibile a tutti gli utenti che visualizzano l'annuncio, perciò è necessaria solamente l'associazione tra l'utente che scrive il commento e l'annuncio.



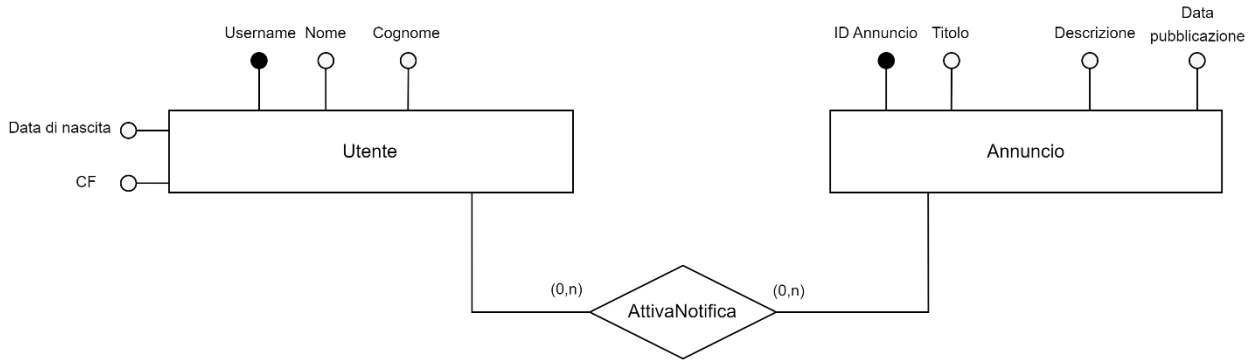
Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Ci si accorge quindi della somiglianza tra i concetti di *Commento* e *Messaggio* e si sceglie pertanto di utilizzare una generalizzazione totale, avente come entità padre l'entità *Interazione* e come entità figlie *Commento* e *Messaggio*. La porzione di schema risultante è la seguente.



Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

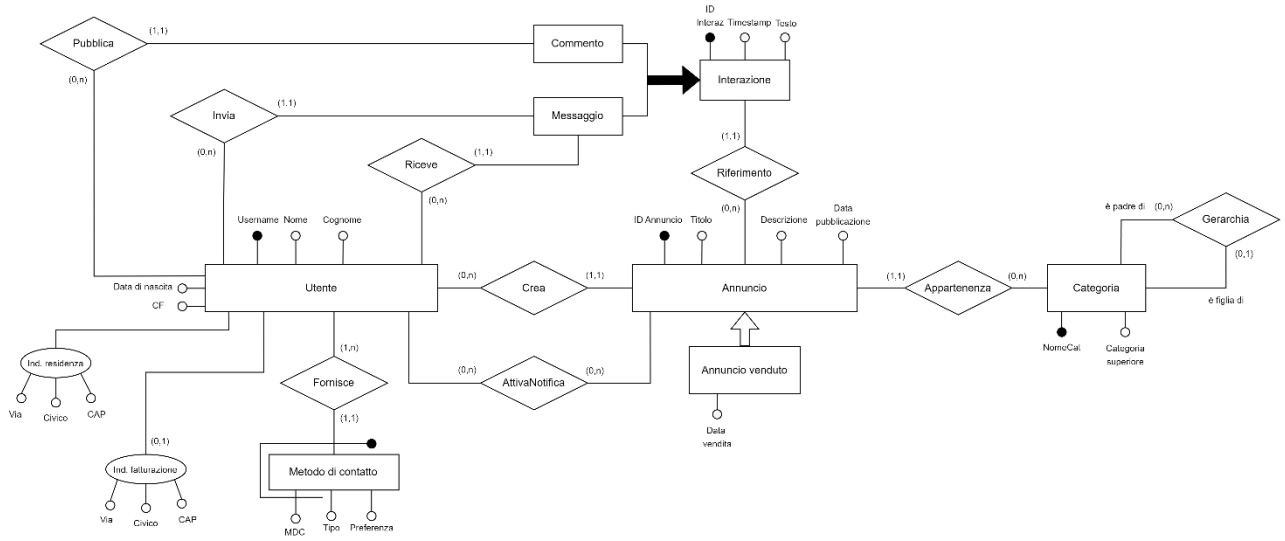
Si procede andando a catturare la possibilità per un utente di attivare le notifiche per un annuncio. Questo viene realizzato tramite una semplice associazione.



Gli attributi "Ind. fatturazione" e "Ind. residenza" sono stati omessi per rendere più chiaro lo schema.

Integrazione finale

Di seguito è presentata l'integrazione finale dello schema concettuale.



Regole aziendali

L'attributo *Data vendita* dell'entità *Annuncio venduto* deve essere maggiore o uguale all'attributo *Data pubblicazione*.

L'utente mittente di un messaggio non può coincidere con l'utente destinatario del messaggio stesso.

L'utente deve specificare esattamente un metodo di contatto come preferito.

Dizionario dei dati

Per le entità figlie sono riportati solo gli attributi e gli identificatori aggiuntivi rispetto all'entità padre.

Entità	Descrizione	Attributi	Identificatori
Utente	Utente del sistema.	Nome, Cognome, Indirizzo residenza, Indirizzo fatturazione, Data di nascita, CF	Username
Annuncio	Annuncio pubblicato in bacheca da un utente, permette la vendita di un oggetto.	Titolo, Descrizione, Data pubblicazione	ID Annuncio
Annuncio venduto	Specializza *Annuncio*. Definisce gli annunci contrassegnati come venduti.	Data vendita	
Interazione	Intesa in senso generico, è l'entità che definisce lo scambio di informazioni (pubblico o privato) tra utenti riguardante un annuncio.	Timestamp, Testo	ID Interaz
Commento	Specializza *Interazione*. Commento pubblico inseribile da un utente in un annuncio.		

Messaggio	Specializza *Interazione*. Messaggio inviato privatamente da un utente mittente a un utente destinatario.		
Categoria	Categoria di appartenenza di un annuncio.		NomeCat
Metodo di contatto	Metodo di contatto fornito dall'utente.		MDC, Fornisce

4. Progettazione logica

Volume dei dati

Si suppone di mantenere i dati degli annunci per non più di 10 anni, mentre i dati relativi alle interazioni vengono mantenuti per massimo 5 anni.

Ogni utente fornisce in media un solo metodo di contatto.

Solo il 10% degli utenti fornisce anche un indirizzo di fatturazione.

Si stima che ogni utente pubblichi in media 2 annunci al mese.

Ogni annuncio riceve in media 10 interazioni, di cui l'80% sono messaggi privati e il 20% commenti pubblici.

Per ogni annuncio vengono attivate le notifiche da 5 utenti in media e vengono disattivate (manualmente dagli utenti) subito dopo che l'annuncio viene contrassegnato come venduto.

Ogni giorno vengono creati circa 200 annunci e il tempo di vendita medio è di circa 20 giorni.

Concetto nello schema	Tipo[1]	Volume atteso
Utente	E	3.000
Metodo di contatto	E	3.000
Annuncio	E	730.000
Annuncio venduto	E	726.000
Interazione	E	3.650.000
Messaggio	E	2.920.000
Commento	E	730.000
Categoria	E	30
Crea	R	730.000
Pubblica	R	730.000
Invia	R	2.920.000
Riceve	R	2.920.000
Riferimento	R	3.650.000
Appartenenza	R	730.000
Gerarchia	R	30
AttivaNotifica	R	3.650.000

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
U1	Registrazione nuovo utente	1 / giorno
C1	Inserimento nuova categoria	3 / anno
A1	Creazione nuovo annuncio	200 / giorno
A2	Contrassegna annuncio come venduto	200 / giorno

[1] Indicare con E le entità, con R le relazioni

A3	Lista annunci pubblicati dall'utente	200 / mese
A4	Lista annunci attivi	300 / giorno
N1	Utente attiva notifiche per annuncio	1000 / giorno
N2	Lista annunci con notifiche attive	3000 / settimana
I1	Utente pubblica commento su annuncio	400 / giorno
I2	Utente scrive un messaggio privato	1600 / giorno
R1	Report percentuale annunci venduti per utente	2 / anno

Costo delle operazioni

Per calcolare il costo delle operazioni assumiamo che il costo di un accesso in lettura sia 1 e che quello di un accesso in scrittura sia 2.

Operazione U1 – Registrazione nuovo utente

Per registrare un nuovo utente è necessario anche registrare i metodi di contatto che egli fornisce. Date le precedenti supposizioni sappiamo che un utente fornisce un solo metodo di contatto, avremo quindi bisogno di un solo accesso in scrittura su *Utente*, *Fornisce* e *Metodo di contatto*.

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	S
Fornisce	R	1	S
Metodo di contatto	E	1	S

Costo operazione: 3*2 = 6 accessi

Costo totale: 6 accessi / giorno

Operazione C1 – Inserimento nuova categoria

Per inserire una nuova categoria è necessario un accesso in scrittura sull'entità *Categoria* e uno sulla relazione *Gerarchia* per salvare anche la corrispondente tupla genitore-figlio.

Concetto	Costrutto	Accessi	Tipo
Categoria	E	1	S
Gerarchia	R	1	S

Costo operazione: 2*2 = 4 accessi

Costo totale: 12 accessi / anno

Operazione A1 – Creazione nuovo annuncio

Questa operazione va ad inserire un nuovo annuncio nella bacheca. Per fare ciò si deve accedere in scrittura alla relazione *Crea*, all'entità *Annuncio* e alla relazione *Appartenenza*. Si noti che la scelta della categoria non è stata inclusa nel costo di questa operazione, essendo un'operazione a parte che prevede l'interazione con l'utente.

Concetto	Costrutto	Accessi	Tipo
Pubblica	R	1	S
Annuncio	E	1	S
Appartenenza	R	1	S

Costo operazione: 3*2 = 6 accessi

Costo totale: 1200 accessi / giorno

Operazione A2 – Contrassegna annuncio come venduto

Questa operazione ha bisogno di un accesso in lettura sull'entità *Annuncio* per memorizzare l'annuncio in questione, un accesso in scrittura per rimuovere l'annuncio e infine un accesso in scrittura per scrivere l'annuncio in *Annuncio venduto*.

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	1	L
Annuncio	E	1	S
Annuncio venduto	E	1	S

Costo operazione: 1+2+2 = 5 accessi

Costo totale: 1000 accessi / giorno

Operazione A3 – Lista annunci pubblicati dall'utente

Questa operazione va a mostrare all'utente tutti gli annunci che egli ha pubblicato nel tempo sulla piattaforma. L'operazione ha dunque bisogno di un accesso in lettura sull'entità *Utente*, un accesso in lettura sulla relazione *Crea* per recuperare i suoi annunci ed infine esegue 240 accessi in lettura sull'entità *Annuncio* dato che in media un utente pubblica 2 annunci al mese (2*12*10=240 annunci in 10 anni).

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	L
Pubblica	R	1	L

Annuncio	E	240	L

Costo operazione: 1+1+240 = 242 accessi

Costo totale: 48400 accessi / mese

Operazione A4 – Lista annunci attivi

Questa operazione va a mostrare all'utente tutti gli annunci attivi disponibili sulla piattaforma. Per fare ciò si deve sottrarre, all'insieme contenente tutte le occorrenze dell'entità *Annuncio*, l'insieme contenente tutte le occorrenze dell'entità figlia *Annuncio venduto*, ottenendo in questo modo solo le occorrenze che fanno parte di *Annuncio* e che non fanno parte di *Annuncio venduto*. Questo, per le stime fatte in precedenza, equivale a circa 730.000 accessi in lettura sull'entità *Annuncio* e 726.000 accessi in lettura sull'entità *Annuncio venduto*.

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	730.000	L
Annuncio venduto	E	726.000	L

Costo operazione: 730.000 + 726.000 = 1.456.000 accessi

Costo totale: 1.456.000 * 300 = 436.800.000 accessi / giorno

Operazione N1 – Utente attiva notifiche per annuncio

In questo caso è necessario solo un accesso in scrittura sulla relazione *AttivaNotifica.*

Concetto	Costrutto	Accessi	Tipo
AttivaNotifica	R	1	S

Costo operazione: 2 accessi

Costo totale: 2000 accessi / giorno

Operazione N2 – Lista annunci con notifiche attive

Questa operazione va a mostrare all'utente tutti gli annunci per cui ha attivato le notifiche.

Si inizia accedendo in lettura all'entità *Utente*, poi si accede in lettura alla relazione *AttivaNotifica* e infine, supponendo che ogni utente in media abbia le notifiche attive per 8 annunci e che quando un annuncio viene venduto l'utente disabiliti le notifiche per quell'annuncio, si effettuano 8 accessi in lettura all'entità *Annuncio*.

Concetto	Costrutto	Accessi	Tipo

Utente	E	1	L
AttivaNotifica	R	1	L
Annuncio	E	8	L

Costo operazione: 1+1+8 = 10 accessi

Costo totale: 30000 accessi / settimana

Operazione I1 – Utente pubblica commento su annuncio

Vengono effettuati un accesso in scrittura sulla relazione *Pubblica,* uno sull'entità *Commento* e uno sulla relazione *Riferimento*.

Concetto	Costrutto	Accessi	Tipo
Scrive	R	1	S
Commento	E	1	S
Riferimento	R	1	S

Costo operazione: 3*2 = 6 accessi

Costo totale: 2400 accessi / giorno

Operazione I2 – Utente scrive un messaggio privato

Questa operazione accede in scrittura alle relazioni *Invia, Riceve, Riferimento* e all'entità *Messaggio,* ma ha bisogno anche di due accessi in lettura all'entità *Utente,* per controllare che mittente e destinatario siano due utenti diversi.

Concetto	Costrutto	Accessi	Tipo
Utente	E	2	L
Invia	R	1	S
Messaggio	E	1	S
Riceve	R	1	S
Riferimento	R	1	S

Costo operazione: 2*1+4*2 = 10 accessi

Costo totale: 16000 accessi / giorno

Operazione R1 – Report percentuale annunci venduti per utente

Per generare questo report è necessario innanzitutto accedere a tutte le occorrenze dell'entità *Utente,* poi a tutte le occorrenze dell'entità *Annuncio* e *Annuncio venduto* e alla relazione *Pubblica*. Si va

quindi, per ogni utente, a recuperare tutti gli annunci che egli ha pubblicato, andando ad accedere in particolare anche a quelli contrassegnati come venduti.

Concetto	Costrutto	Accessi	Tipo
Utente	E	3.000	L
Pubblica	R	730.000	L
Annuncio	E	730.000	L
Annuncio venduto	E	726.000	L

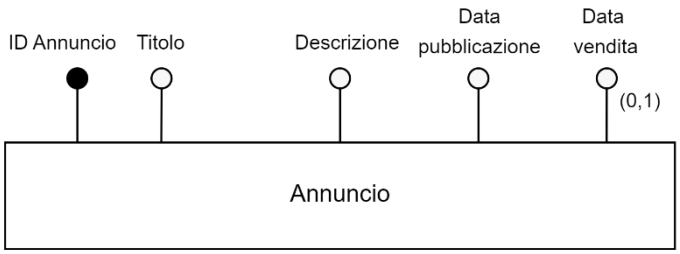
Costo operazione: 3000 + 730000 + 730000 + 726000 = 2.189.000 accessi

Costo totale: 4.378.000 accessi / anno

Ristrutturazione dello schema E-R

Si inizia col rimuovere le generalizzazioni.

La prima generalizzazione presa in esame è quella di *Annuncio* con *Annuncio venduto*. Si sceglie di rimuovere questa generalizzazione accorpando l'entità figlia nel genitore, andando ad aggiungere un attributo opzionale Data vendita in *Annuncio*. Questa scelta permette di ridurre il costo delle operazioni A2, A4 ed R1 mantenendo valido lo schema.



Il costo aggiornato delle operazioni A2 ed R1 è il seguente:

Operazione A2:

Si mantiene l'accesso in lettura per assicurarsi che la data di vendita sia successiva o uguale alla data di pubblicazione, si procede poi con un accesso in scrittura sull'entità *Annuncio* per aggiornare il suo attributo Data vendita.

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	1	L
Annuncio	E	1	S

Costo operazione: 1+2 = 3 accessi

Costo totale: 600 accessi / giorno

Operazione A4:

Non è più necessario ora accedere in lettura alle occorrenze dell'entità *Annuncio venduto*. Il costo di questa operazione rimane molto alto, tuttavia non sono stati individuati altri modi per ridurlo ulteriormente, essendo caratterizzato solamente da letture su un'unica entità.

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	730.000	L

Costo operazione: 730.000 accessi

Costo totale: 730.000 * 300 = 219.000.000 accessi / giorno

Operazione R1:

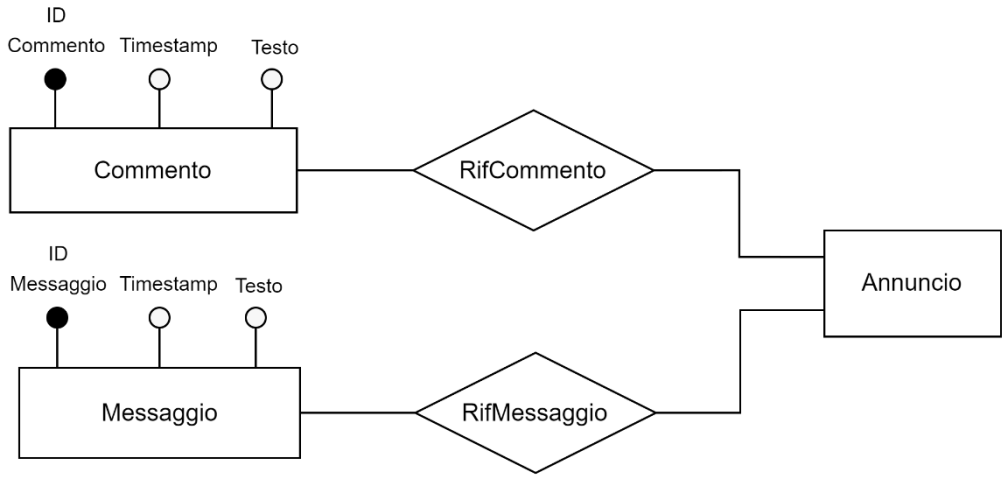
Si risparmiano tutti gli accessi in lettura che erano relativi all'entità *Annuncio venduto* in quanto ora si trova tutto incorporato nell'entità *Annuncio*.

Concetto	Costrutto	Accessi	Tipo
Utente	E	3.000	L
Pubblica	R	730.000	L
Annuncio	E	730.000	L

Costo operazione: 3000 + 730000 + 730000 = 1.463.000 accessi

Costo totale: 2.926.000 accessi / anno

Si procede col rimuovere la generalizzazione dell'entità *Interazione*. In questo caso, dato che tutte le operazioni si interfacciano con le entità figlie *Messaggio* e *Commento*, si decide di accorpare l'entità genitore nelle entità figlie producendo la seguente porzione di schema. Non si evidenziano differenze prestazionali rispetto allo schema concettuale precedente.



L'entità Annuncio è rappresentata in versione semplificata per motivi di leggibilità.

Si prosegue rimuovendo l'associazione *Gerarchia* e andando ad aggiungere l'attributo *Categoria superiore* all'entità *Categoria*. Si noti che questo attributo è opzionale in quanto le categorie ai vertici della gerarchia non hanno una categoria superiore. Questo comporta una riduzione del costo dell'operazione di inserimento di una categoria.

Operazione C1:

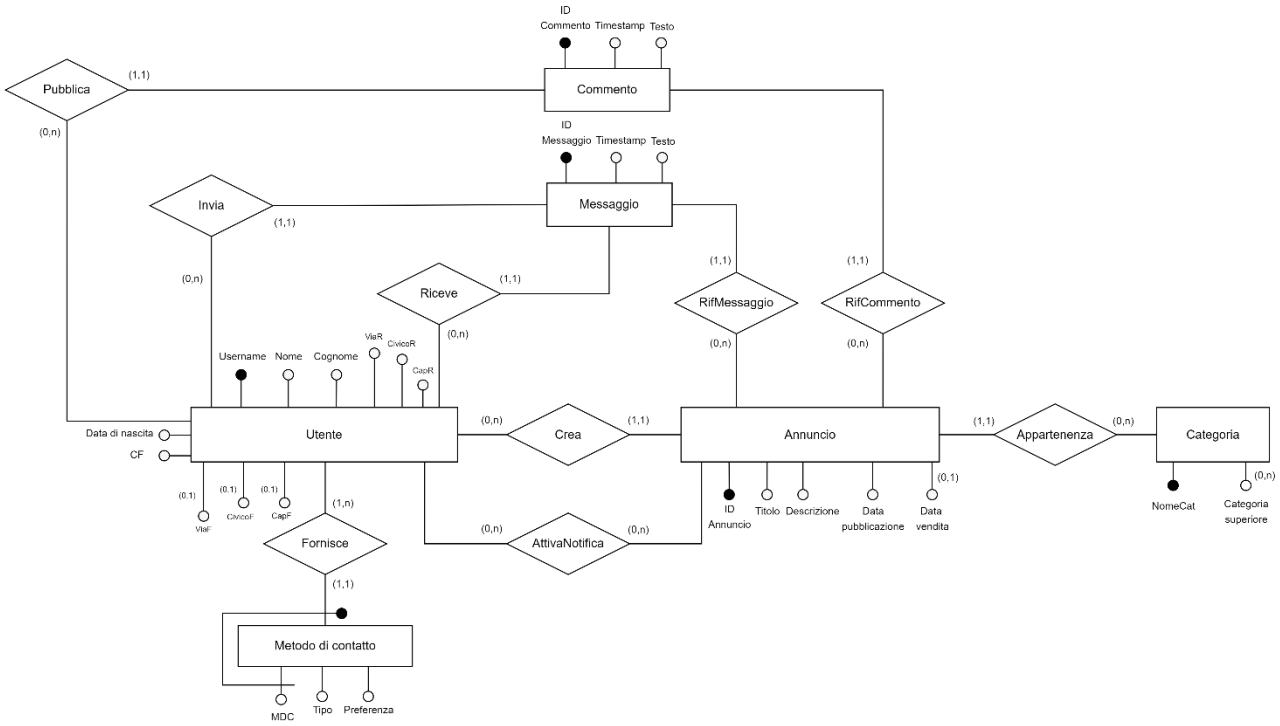
Concetto	Costrutto	Accessi	Tipo
Categoria	E	1	S

Costo operazione: 2 accessi

Costo totale: 6 accessi / anno

Infine si rimuovono gli attributi composti relativi agli indirizzi dell'utente. In questo caso, essendo gli indirizzi poco significativi nel dominio dell'applicazione si sceglie di rappresentarli come attributi dell'entità utente, così da non aumentare il costo dell'operazione U1, che altrimenti avrebbe dovuto prevedere ulteriori scritture su un'eventuale entità *Indirizzo*.

Lo schema ristrutturato è il seguente:



Si noti che gli attributi *ViaR, CivicoR, CapR* si riferiscono all'indirizzo di residenza dell'utente, mentre gli attributi *ViaF, CivicoF, CapF* si riferiscono all'indirizzo di fatturazione.

Trasformazione di attributi e identificatori

Non sono presenti attributi ripetuti o identificatori esterni da modificare.

Traduzione di entità e associazioni

UTENTE(<u>username</u>, cf, nome, cognome, data_ nascita, r_via, r_civico, r_cap, f_via*, f_civico*, f_cap*)

ANNUNCIO(<u>ID_annuncio</u>, utente, titolo, descrizione, categoria, data_pubblicazione, data_vendita*)

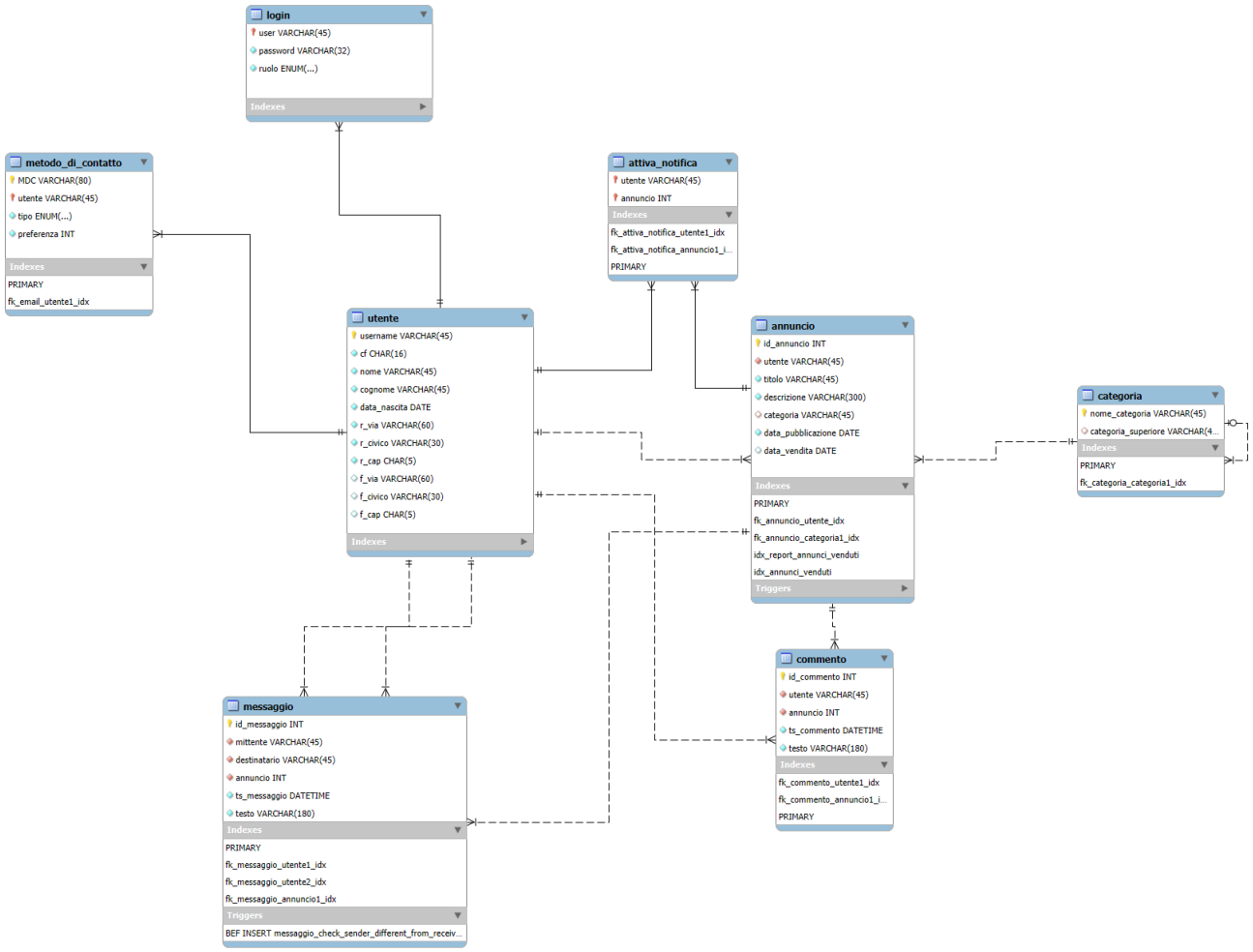
CATEGORIA(<u>nome_categoria</u>, categoria_superiore*)

MESSAGGIO(<u>ID_messaggio</u>, mittente, destinatario, annuncio, ts_messaggio, testo)

COMMENTO(<u>ID_commento</u>, utente, annuncio, ts_commento, testo)

METODO DI CONTATTO(<u>MDC, utente</u>, tipo, preferenza)

ATTIVA NOTIFICA(<u>utente, annuncio</u>)



Normalizzazione del modello relazionale

Dal punto di vista concettuale potrebbe essere opportuno avere due entità separate *Indirizzo residenza* e *Indirizzo fatturazione* associate all'entità *Utente*. Tuttavia, per quanto detto in precedenza, si è scelto di integrare questi due indirizzi come attributi nell'entità *Utente*. Questo non causa problemi riguardo violazioni delle forme normali in quanto lo schema presentato è in BCNF e permette anche migliori prestazioni nell'operazione di registrazione di un nuovo utente evitando scritture su delle ipotetiche entità *Indirizzo*.

5. Progettazione fisica

Utenti e privilegi

Si decide di non permettere l'accesso al sistema ad utenti non autenticati, viene quindi implementata una tabella Login in cui vengono memorizzate le credenziali di login dei vari utenti correlate con il loro ruolo. Dalle specifiche si evince che sono necessari due tipi di utente:

- Utente (generico, da qui in poi semplicemente "utente")
- Amministratore

Gli utenti sono i veri attori della piattaforma, ovvero coloro che possono scambiarsi messaggi, pubblicare annunci, attivare notifiche ecc. Inoltre gli utenti possono registrarsi autonomamente.
Gli amministratori invece si occupano di gestire la gerarchia delle categorie e generare report.
I relativi utenti nella base di dati sono quindi:

- ba_login
- ba_user
- ba_admin

Dopo aver effettuato il login con successo verrà assegnato all'utente il profilo che gli spetta.

I permessi assegnati sono i seguenti:

```
GRANT EXECUTE ON procedure `bacheca_online`.`login` TO 'ba_login';
GRANT EXECUTE ON procedure `bacheca_online`.`registraUtente` TO 'ba_login';
GRANT EXECUTE ON procedure `bacheca_online`.`inserisciMetodiDiContatto` TO 'ba_login';

GRANT EXECUTE ON procedure `bacheca_online`.`attivaNotifichePerAnnuncio` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`contrassegnaAnnuncioVenduto` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`creaAnnuncio` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`listaAnnunciAttivi` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`listaAnnunciNotificheAttive` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`listaAnnunciPubblicatiUtente` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`scriviMessaggio` TO 'ba_user';
GRANT EXECUTE ON procedure `bacheca_online`.`pubblicaCommento` TO 'ba_user';

GRANT EXECUTE ON procedure `bacheca_online`.`reportPercentualeAnnunciVendutiUtente` TO
'ba_admin';
```

GRANT EXECUTE ON procedure `bacheca_online`.`aggiungiCategoria` TO 'ba_admin';

Strutture di memorizzazione

Tabella login		
Colonna	**Tipo di dato**	**Attributi**[2]
user	Varchar(45)	PK, NN
password	Varchar(32)	NN
ruolo	Enum(‘user’,’admin’)	NN

Tabella utente		
Colonna	**Tipo di dato**	**Attributi**[3]
username	Varchar(45)	PK, NN
cf	Char(16)	NN
nome	Varchar(45)	NN
cognome	Varchar(45)	NN
data_nascita	Date	NN
r_via	Varchar(60)	NN
r_civico	Varchar(15)	NN
r_cap	Char(5)	NN
f_via	Varchar(60)	
f_civico	Varchar(15)	
f_cap	Char(5)	

Tabella annuncio		
Colonna	**Tipo di dato**	**Attributi**[4]
id_annuncio	Int	PK, NN, AI
utente	Varchar(45)	NN
titolo	Varchar(45)	NN
descrizione	Varchar(300)	NN
categoria	Varchar(45)	NN
data_pubblicazione	Date	NN
data_vendita	Date	

Tabella categoria

[2] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Colonna	Tipo di dato	Attributi[5]
nome_categoria	Varchar(45)	PK, NN
categoria_superiore	Varchar(45)	

Tabella messaggio		
Colonna	**Tipo di dato**	**Attributi[6]**
id_messaggio	Int	PK, NN, AI
mittente	Varchar(45)	NN
destinatario	Varchar(45)	NN
annuncio	Int	NN
ts_messaggio	DateTime	NN
testo	Varchar(180)	NN

Tabella commento		
Colonna	**Tipo di dato**	**Attributi[7]**
id_commento	Int	PK, NN, AI
utente	Varchar(45)	NN
annuncio	Int	NN
ts_commento	DateTime	NN
testo	Varchar(180)	NN

Tabella attiva_notifica		
Colonna	**Tipo di dato**	**Attributi[8]**
utente	Varchar(45)	PK, NN
annuncio	Int	PK, NN

Tabella metodo_di_contatto		
Colonna	**Tipo di dato**	**Attributi[9]**
MDC	Varchar(80)	PK, NN
utente	Varchar(45)	PK, NN
tipo	Enum('Email','Telefono','Cellulare')	NN
preferenza	Int	NN

Indici

Si sceglie di implementare questo indice per migliorare le prestazioni dell'operazione R1, infatti con questo indice si vanno a velocizzare le query di raggruppamento dell'utente e di controllo sulla data di vendita degli annunci.

Tabella annuncio	
Indice report_annunci_venduti	**Tipo[10]:**
utente, data_vendita	IDX

Si sceglie inoltre di creare un ulteriore indice solo sull'attributo *data_vendita* per ottimizzare al meglio l'operazione A4, che è l'operazione più frequente in assoluto nel sistema.

Tabella annuncio	
Indice annunci_venduti	**Tipo[11]:**
data_vendita	IDX

Trigger

Utilizziamo i trigger principalmente per andare ad implementare i controlli richiesti dalle regole aziendali.

Tabella messaggio: BEFORE INSERT:

Con questo trigger si va a controllore che l'utente mittente del messaggio sia diverso dall'utente destinatario.

```
CREATE TRIGGER `bacheca_online`.`messaggio_check_sender_different_from_receiver` BEFORE INSERT ON `messaggio` FOR EACH ROW
BEGIN
    IF New.mittente = New.destinatario THEN
        SIGNAL SQLSTATE '45000' SET message_text = "Il mittente del messaggio non può coincidere con il destinatario";
    END IF;
END
```

[11] IDX = index, UQ = unique, FT = full text, PR = primary.

Tabella annuncio: BEFORE INSERT:

Questo trigger ha l'obiettivo di assicurarsi che un annuncio non possa essere inserito da subito con una data di vendita già impostata.

```
CREATE TRIGGER `bacheca_online`.`annuncio_check_data_vendita_null` BEFORE INSERT ON `annuncio` FOR EACH ROW
BEGIN
	IF New.data_vendita <> null THEN
		SIGNAL SQLSTATE '45000' SET message_text = "La data di vendita non può essere impostata al momento dell'inserimento";
	END IF;
END
```

Tabella annuncio: BEFORE UPDATE:

Con questo trigger ci si assicura che la data di vendita impostata sia uguale o successiva alla data di pubblicazione dell'annuncio.

```
CREATE TRIGGER `bacheca_online`.`annuncio_check_vendita_dopo_pubblicazione` BEFORE UPDATE ON `annuncio` FOR EACH ROW
BEGIN
	IF New.data_vendita < Old.data_pubblicazione THEN
		SIGNAL SQLSTATE '45000' SET message_text = "La data di vendita deve corrispondere o essere successiva alla data di pubblicazione";
	END IF;
END
```

Eventi

Gli eventi implementati sono eventi di cleanup, istanziati in fase di configurazione del sistema, e servono a rimuovere i dati relativi alle interazioni (messaggi e commenti) dopo 5 anni e i dati relativi agli annunci dopo 10 anni. Si è scelto di eseguire il controllo ogni 3 mesi in modo arbitrario.

Cleanup annunci dopo 10 anni:

```
CREATE EVENT IF NOT EXISTS `bacheca_online`.`clean_annunci_10_anni`
ON SCHEDULE
  EVERY 3 MONTH
ON COMPLETION PRESERVE
```

```
DO BEGIN
  DELETE FROM `annuncio` WHERE DATEDIFF(CURDATE(),'data_pubblicazione')>=3650;
END
```

Cleanup interazioni dopo 5 anni:

```
CREATE EVENT IF NOT EXISTS `bacheca_online`.`clean_interazioni_5_anni`
ON SCHEDULE
  EVERY 3 MONTH
ON COMPLETION PRESERVE
DO BEGIN
  DELETE FROM `messaggio` WHERE DATEDIFF(NOW(),'ts_messaggio')>=1825;
  DELETE FROM `commento` WHERE DATEDIFF(NOW(),'ts_commento')>=1825;
END
```

Viste

Non è stato ritenuto opportuno l'utilizzo di viste.

Stored Procedures e transazioni

PROCEDURA Login:

```
CREATE PROCEDURE `login`(in var_user VARCHAR(80), in var_password VARCHAR(80), out var_ruolo ENUM('user', 'admin'))
BEGIN
    DECLARE var_check_username VARCHAR(80);

    SELECT user, ruolo INTO var_check_username, var_ruolo
    FROM Login
    WHERE user = var_user and password = MD5(var_password);

    IF var_check_username IS NULL THEN
    SIGNAL SQLSTATE '45000' set message_text = 'Credenziali errate';
    END IF;
END
```

OPERAZIONE U1 – registraUtente

L'operazione registraUtente prevede l'inserimento dei dati di un nuovo utente nelle tabelle *Login*, *Utente* e *Metodo_di_contatto*. Per quanto riguarda l'inserimento dei metodi di contatto ci si assicura anche che la regola aziendale venga rispettata (#metodi di contatto >= 1 e #preferiti = 1) tramite due controlli nell'operazione registraUtente. I metodi di contatto vengono passati come parametro alla procedura e sono nella forma "tipo:contatto:preferenza; …". Questa lista di metodi di contatto va pertanto tokenizzata (utilizzando come delimitatore il carattere ';') e poi divisa nelle sue componenti (tipo, contatto e preferenza) utilizzando stavolta il delimitatore ':'.
Si sceglie come livello di isolamento Read Uncommitted perché l'operazione effettua solamente inserimenti, che quindi non necessitano di controllare la presenza di lock sulle tuple.

```
CREATE PROCEDURE `registraUtente` (in var_username varchar(45), in var_password varchar(32), in var_cf char(16), in var_nome varchar(45), in var_cognome varchar(45), in var_data_nascita date, in var_r_via varchar(60), in var_r_civico varchar(15), in var_r_cap char(5), in var_f_via varchar(60), in var_f_civico varchar(15), in var_f_cap char(5), in var_metodi_di_contatto TEXT)
BEGIN

  DECLARE var_num_contatti_inseriti INT;
  DECLARE var_num_preferiti INT;
      DECLARE exit handler for sqlexception

  begin
            rollback;
    resignal;
      end;

  set transaction isolation level read uncommitted;
  start transaction;

  INSERT INTO utente (username, cf, nome, cognome, data_nascita, r_via, r_civico, r_cap, f_via, f_civico, f_cap) VALUES (var_username, var_cf, var_nome, var_cognome, var_data_nascita, var_r_via, var_r_civico, var_r_cap, var_f_via, var_f_civico, var_f_cap);
  INSERT INTO login (user, password, ruolo) VALUES (var_username, MD5(var_password), 'user');
```

```
   call inserisciMetodiDiContatto(var_username, var_metodi_di_contatto, var_num_contatti_inseriti, var_num_preferiti);
   if var_num_contatti_inseriti <= 0 then
      signal sqlstate '45000' set message_text = "L'utente deve fornire almeno un recapito";
   end if;

   -- Eseguo il controllo qui e non in un trigger apposito in modo da effettuarlo una volta sola, invece che per ogni inserimento.
   IF var_num_preferiti <> 1 THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "L'utente deve selezionare esattamente un metodo di contatto come preferito";
        END IF;

   commit;
END
```

La procedura inserisciMetodiDiContatto è la seguente:

```
CREATE PROCEDURE `inserisciMetodiDiContatto` (in var_username VARCHAR(45), in var_metodi_di_contatto TEXT, out var_num_contatti_inseriti INT, out var_num_preferiti INT)
BEGIN

        DECLARE var_temp TEXT;
   DECLARE var_i INT;
   DECLARE var_tipo VARCHAR(50);
   DECLARE var_contatto VARCHAR(80);
   DECLARE var_preferenza INT;
   SET var_num_contatti_inseriti = 0;
   SET var_num_preferiti = 0;
        -- separo metodi di contatto col ; (for loop)
   -- per ogni metodo di contatto estraggo le parti che mi interessano
   SET var_i = 1;

   insert_metodi_di_contatto: loop
```

```
    SET var_temp = tokenizzaMetodiDiContatto(var_metodi_di_contatto, var_i);
    IF var_temp = ''
    THEN
      LEAVE insert_metodi_di_contatto;
    END IF;

    -- Estrarre i valori separati da ':'
            SELECT
    SUBSTRING_INDEX(var_temp, ':', 1),
    SUBSTRING_INDEX(SUBSTRING_INDEX(var_temp, ':', 2), ':', -1),
    CAST(SUBSTRING_INDEX(var_temp, ':', -1) AS UNSIGNED)
            INTO var_tipo, var_contatto, var_preferenza;

    INSERT INTO metodo_di_contatto(MDC, utente, tipo, preferenza) VALUES (var_contatto, var_username, var_tipo, var_preferenza);
    SET var_num_contatti_inseriti = var_num_contatti_inseriti+1;

    IF var_preferenza = 1
    THEN SET var_num_preferiti = var_num_preferiti+1;
    END IF;

    SET var_i = var_i+1;
  END LOOP;
END
```

Ed infine questa è la funzione tokenizzaMetodiDiContatto:

```
CREATE FUNCTION `tokenizzaMetodiDiContatto` (var_lista_metodi TEXT, var_posizione INT)
RETURNS TEXT
DETERMINISTIC
BEGIN
        DECLARE var_num_token INT;

  -- Conta quanti token ci sono (numero di ';' + 1)
```

```
   SET var_num_token = (LENGTH(var_lista_metodi) - LENGTH(REPLACE(var_lista_metodi, ';', ''))) + 1;

   -- Se la posizione richiesta è maggiore del numero di token, restituisce stringa vuota
   IF var_posizione > var_num_token THEN
      RETURN '';
   END IF;

   return substring_index(substring_index(var_lista_metodi, ';', var_posizione), ';', -1);
END
```

OPERAZIONE C1 – aggiungiCategoria

Quest'operazione va ad effettuare un semplice inserimento di una nuova categoria all'interno della relativa tabella.

```
CREATE PROCEDURE `aggiungiCategoria` (in var_nomecat VARCHAR(45), in var_cat_sup VARCHAR(45))
BEGIN
        INSERT INTO categoria (nome_categoria, categoria_superiore) VALUES (var_nomecat, var_cat_sup);
END
```

OPERAZIONE A1 – creaAnnuncio

L'operazione creaAnnuncio va ad inserire un nuovo annuncio nella tabella *Annuncio*.

```
CREATE PROCEDURE `creaAnnuncio` (in var_utente VARCHAR(45), in var_titolo VARCHAR(45), in var_descrizione VARCHAR(300), in var_categoria VARCHAR(45) , in data_pubblicazione DATE)

BEGIN

        INSERT INTO annuncio(utente, titolo, descrizione, categoria, data_pubblicazione) VALUES (var_utente, var_titolo, var_descrizione, var_categoria, curdate());
```

```
END
```

OPERAZIONE A2 – contrassegnaAnnuncioVenduto

Questa operazione va a modificare l'attributo data_vendita di *Annuncio*. Dopo lo statement di Update si esegue un controllo su ROW_COUNT per verificare il numero di righe modificate. Se questo numero è 0 viene sollevata un'eccezione, in modo da fornire all'utente un feedback chiaro se l'aggiornamento fallisce.

```
CREATE PROCEDURE `contrassegnaAnnuncioVenduto` (in var_annuncio INT, in var_data_vendita
DATE)
BEGIN
        UPDATE annuncio
   SET data_vendita = var_data_vendita
   WHERE id_annuncio = var_annuncio AND data_vendita IS NULL;

   IF ROW_COUNT() = 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'L''annuncio risulta già contrassegnato come venduto o non esiste';
   END IF;
END
```

OPERAZIONE A3 – listaAnnunciPubblicatiUtente

Questa operazione va a mostrare all'utente tutti gli annunci che egli ha pubblicato nel tempo, sia attivi che venduti.

```
CREATE PROCEDURE `listaAnnunciPubblicatiUtente` (in var_utente VARCHAR(45))
BEGIN
   SELECT * FROM annuncio WHERE utente = var_utente;
END
```

OPERAZIONE A4 – listaAnnunciAttivi

Quest'operazione va a mostrare all'utente tutti gli annunci attivi sulla piattaforma in quel determinato momento.

```
CREATE PROCEDURE `listaAnnunciAttivi` ()
BEGIN

    SELECT id_annuncio, utente, titolo, descrizione, categoria, data_pubblicazione
  FROM annuncio
  WHERE data_vendita IS NULL
  ORDER BY data_pubblicazione DESC;

END
```

OPERAZIONE N1 – attivaNotifichePerAnnuncio

L'operazione va ad eseguire una semplice INSERT nella tabella *attiva_notifica*, al fine di memorizzare il fatto che l'utente ha attivato le notifiche per un annuncio.

```
CREATE PROCEDURE `attivaNotifichePerAnnuncio` (in var_utente VARCHAR(45), in var_annuncio int1)
BEGIN
    INSERT INTO attiva_notifica (utente, annuncio) VALUES (var_utente, var_annuncio);
END
```

OPERAZIONE N2 – listaAnnunciNotificheAttive

Questa operazione va a mostrare all'utente tutti gli annunci per cui ha attivato le notifiche.

```
CREATE PROCEDURE `listaAnnunciNotificheAttive` (in var_utente VARCHAR(45))
BEGIN
    SELECT A.id_annuncio, A.utente, A.titolo, A.descrizione, A.categoria, A.data_pubblicazione
  FROM attiva_notifica as AN
  JOIN annuncio as A ON AN.annuncio = A.id_annuncio
  WHERE AN.utente = var_utente
  ORDER BY A.data_pubblicazione DESC;
END
```

OPERAZIONE I1 – pubblicaCommento

L'operazione pubblicaCommento va ad inserire nella tabella *Commento* le informazioni del relativo commento scritto dall'utente su un annuncio.

```
CREATE PROCEDURE `pubblicaCommento` (in var_utente VARCHAR(45), in var_annuncio INT, in var_testo VARCHAR(180))
BEGIN
  INSERT INTO commento (utente, annuncio, ts_commento, testo) VALUES (var_utente, var_annuncio, now(), var_testo);
END
```

OPERAZIONE I2 – scriviMessaggio

Similmente all'operazione precedente questa operazione va ad effettuare la scrittura nel DB delle informazioni relative al messaggio scritto dall'utente.

```
CREATE PROCEDURE `scriviMessaggio` (in var_mittente VARCHAR(45), in var_destinatario VARCHAR(45), in var_annuncio INT, in var_testo VARCHAR(180))
BEGIN
  INSERT INTO messaggio (mittente, destinatario, annuncio, ts_messaggio, testo) VALUES (var_mittente, var_destinatario, var_annuncio, now(), var_testo);
END
```

OPERAZIONE R1 – reportPercentualeAnnunciVendutiUtente

Questa operazione va ad effettuare il report riguardante la percentuale di annunci venduti per ogni utente.

```
CREATE PROCEDURE `reportPercentualeAnnunciVendutiUtente` ()
BEGIN
    SELECT
    A.utente,
    COUNT(*) AS totale_annunci,
    COUNT(CASE WHEN A.data_vendita IS NOT NULL THEN 1 END) AS annunci_venduti,
    ROUND((COUNT(CASE WHEN a.data_vendita IS NOT NULL THEN 1 END) * 100.0) / COUNT(*), 2) AS percentuale_annunci_venduti
    FROM annuncio AS A
```

```
    GROUP BY A.utente
    ORDER BY percentuale_annunci_venduti DESC;
END
```

Oltre a queste procedure principali sono state introdotte, per motivi di utilizzo applicativo, anche le seguenti procedure non esplicitate precedentemente.

Per l'utente ba_admin:
-getCategories

Per l'utente ba_user:
-getCategories
-disattiveNotifichePerAnnuncio
-checkNotificheOn
-getCommentiAnnuncio
-getChatUtente
-getMessaggiChat