***Titolo***

Progettazione e descrizione di una base di dati relazionale per la gestione e memorizzazione di rubriche telefoniche avanzate.

**Introduzione**

Per espandere ulteriormente il dominio del problema abbiamo optato per l’inserimento dell’entità utente, in questo modo è permessa l’esistenza di più rubriche corrispondenti a diversi utenti. I contatti saranno infatti associati all’utente che li ha salvati, dando la possibilità di accedere alla propria rubrica tramite delle credenziali di accesso.

**Analisi del problema**

Si vuole progettare una base di dati che permetta all’utente di memorizzare e gestire una rubrica telefonica avanzata.L’utente potrà accedere alla propria rubrica oppure registrarsi per poterne creare una nuova, per effettuare l’accesso e la registrazione sarà necessario fornire e-mail e password valide.  
La rubrica sarà costituita da contatti, in cui verranno mantenute tutte le informazioni principali, e ad esso saranno associati indirizzi di posta elettronica, indirizzi fisici, numeri di telefono e account di messaging. I contatti della rubrica potranno essere organizzati in gruppi.

**Requisiti identificati**

La rubrica deve essere in grado di consentire la memorizzazione delle informazioni riguardanti i contatti, ovvero nome, cognome, una o più e-mail ed eventualmente una foto profilo. Un contatto può essere inserito tra i preferiti e può essere partecipante di uno o più gruppi. I gruppi avranno un nome, una data di creazione e una descrizione.

Ad ogni contatto può essere associato un indirizzo fisico principale e uno o più secondari, che specifichino via, città, zipcode e nazione. Un contatto può avere una sola e-mail principale ed un solo indirizzo fisico principale e tutti i contatti devono essere dotati di almeno un numero di telefono mobile ed un numero di telefono fisso, e per ogni telefono mobile può essere indicato un telefono fisso a cui verranno reindirizzate

eventuali chiamate senza risposta e, analogamente per ogni telefono fisso può essere indicato un telefono

mobile per il reindirizzamento.

Agli stessi indirizzi fisici e numeri di telefono fissi può corrispondere anche più di un contatto.

Inoltre, ad ogni contatto verranno associati gli account a sistemi di messaging. Per ognuno di questi account verrà memorizzato il fornitore, il nickname, la frase di benvenuto e l’indirizzo e-mail collegato che dovrà necessariamente esistere tra gli account di posta già salvati per il contatto.

**Progettazione concettuale**

*\*class diagram non ristrutturato\**

**Rimozione delle gerarchie**

Le gerarchie da eliminare sono le specializzazioni della relazione PhoneNumber. Si eliminano tali specializzazioni e si inserisce un attributo Type di tipo string all’interno della relazione che può assumere solamente i valori “MOBILE” oppure “LANDLINE”.

**Analisi delle ridondanze**

Si è deciso di aggiungere un’associazione tra la relazione User e la relazione Group per snellire le query che richiedessero un accesso a Group necessitando dell’User, operazione che potrebbe essere piuttosto frequente.

**Rimozione degli attributi composti**

Attributo composto in Address?

**Analisi degli indentificativi**

Analizzando la relazione Address ottenuta in seguito alla rimozione degli attributi composti, si nota che la chiave candidata per tale relazione è composta da ben quattro attributi (street, streetName, number, zipCode). Al fine di alleggerire gli accessi sulla relazione, si è deciso quindi di accorpare gli attributi street, streetName e number in un unico attributo di tipo string denominato street.

**Rimozione degli attributi multipli**

L’unico attributo multiplo presente è email nella relazione Contact, al suo posto è stata creata una relazione aggiuntiva nella quale è stato inserito l’attributo main di tipo boolean. Si è preferita questa soluzione per poter aggiungere un’associazione tra la relazione Social e la nuova relazione Email.

**Partizione/Accorpamento delle associazioni**

Si è preferito inoltre eliminare l’associazione tra Contact e Social. In questo modo implementando un vincolo di foreign key sull’attributo assEmail della tabella Social, si rispetta la condizione secondo la quale per ogni account social, la e-mail collegata dovrà necessariamente esistere tra gli account di posta già salvati per il contatto a cui si riferisce.

*\*class diagram ristrutturato\**

**Dizionario delle classi**

**User**: utente proprietario della rubrica

* E-mail (string): e-mail valida del proprietario della rubrica
* Password (string): password di accesso alla rubrica. Deve essere di almeno 8 caratteri
* Nickname (string): nome identificativo del proprietario della rubrica. Deve essere almeno di 3 caratteri

**Contact**: istanza di un contatto salvato nella rubrica

* Name: (string): nome del contatto
* Surname: (string): cognome del contatto
* Propic: (string): percorso sul disco del file contenente l’immagine profilo del contatto

**Address:** indirizzo fisico associato ad uno o più contatti

* Street: (string) stringa contenente nome della strada e numero civico dell’indirizzo
* ZipCode (string): codice postale formato da 5 caratteri (CAP) dell’indirizzo
* City (string): nome della città all’interno dell’indirizzo
* Province (string): identificativo di 2 caratteri della provincia relativa alla città
* Country (string): nazione dell’indirizzo
* Main (boolean): indica se è l’indirizzo principale del contatto. Può esistere solo un indirizzo principale per contatto

**Email**: uno degli indirizzi di posta elettronica associati al contatto

* E-mail (string): stringa contenente l’indirizzo di posta elettronica
* Main (boolean): indica se è l’e-mail principale del contatto. Può esistere solo una e-mail principale per contatto

**Social:** eventualeaccount social o di messaging associato ad un contatto

* supplierName (string): nome del fornitore del social o dell’app di messaging
* nickname (string): nome identificativo dell’account del contatto
* assEmail (string): e-mail associata all’account del contatto. Deve esistere tra le e-mail già salvate per il contatto
* welcome (string): frase di benvenuto o stato personale dell’account

**Phone Number**: numero di telefono associato ad uno o più contatti

* Type (string): tipo di riferimento telefondico. Può essere mobile (MOBILE) oppure fisso (LANDLINE)
* Number (string): stringa contenente il numero di telefono
* linkedNumber: (boolean) indica se è un numero di reindirizzamento. Per ogni contatto deve esistere un numero di reindirizzamento tra i numeri fissi, e uno tra i numeri mobili

**Group:** gruppo a cui possono appartenere uno o più contatti all’interno di una rubrica

* ID (string): chiave tecnica. Codice identificativo del gruppo creato da un utente
* Groupname (string): nome del gruppo all’interno della rubrica
* CreationDate (date): data della creazione del gruppo

**Participants:** descrittore dell’associazione tra i partecipanti ad un gruppo ed il gruppo

* GroupID: (string) ID del gruppo a cui partecipa il contatto
* ContactID (string): ID del contatto partecipante
* joinDate (date): data di inizio partecipazione al gruppo

**Dizionario delle associazioni**

User-Contact  
Esprime l’appartenenza di un contatto ad una rubrica di un utente

User-Group  
Esprime l’appartenenza di un gruppo ad una rubrica di un utente

Group-Participant  
Indica a quale a quale gruppo si riferisce la partecipazione di un contatto

Contact-Participant  
Indica quale contatto è partecipante ad un gruppo

Contact-Address  
Indica a quali contatti si riferisce un indirizzo fisico

Contact-Email  
Indica a quale contatto si riferisce una e-mail

Social-Email  
Indica a quale e-mail è collegato un account social

Contact-PhoneNumber  
Indica a quali contatti si riferisce un numero di telefono

**Dizionario dei vincoli**

**emailFormat**: *n-upla*, una e-mail deve contenere in sequenza almeno un carattere, una @, almeno un carattere, un punto e almeno un carattere.

**passwordLen**: *n-upla*, una password deve contenere almeno 8 caratteri

**usernameLen**: *n-upla*, il nickname dell’user deve contenere almeno 3 caratteri

**uniqueMainEmail**: …, ad un contatto può essere associata una sola e-mail di tipo main

**uniqueMainAddress**: …, ad un contatto può essere associato un solo indirizzo fisico di tipo main

**uniqueLinkedNumber**: …, ad un contatto può essere associato un solo numero di telefono MOBILE di reindirizzamento, ed un solo numero LANDLINE di reindirizzamento

**checkNumberType**: *dominio*, la variabile Type di PhoneNumber può assumere solo i valori “MOBILE” e “LANDLINE”

**checkContactNumbers**: *interrelazionale*, ogni contatto deve avere almeno un telefono fisso e un telefono mobile. *(Traccia) –:* ***trigger*** *in inserimento/modifica di un contatto*

**distinctEmail**: …, non possono esistere due contatti associati alla stessa e-mail*. (Traccia) – intra relazionale (EMAIL: unicità coppia attributi (email, contactID))*

**FK\_socialEmail:** …, l’indirizzo e-mail di un social di un contatto deve esistere necessariamente tra gli account di posta già salvati per il contatto. *(Traccia) SOCIAL: assEmail foreign key*

**PK\_social**: *primary key*, può esistere un solo tipo di account social collegato ad una email *– (SOCIAL: unicità della coppia di attributi “supplierName, assEmail”)*

**timeConsistancygGroup**: *interrelazionale*, la joinDate di un partecipante in un gruppo deve essere successiva alla data di creazione del gruppo

**Schema Logico**

Grassetto: chiave primaria

Doppia sottolineatura: foreign key

USER (**email**, password, nickname, notes)

CONTACT (**ID**, name, surname, propic, favorite, user)

*user -> User.email*

GROUP (**ID**, name, creationDate, description, user)

*user -> User.email*

PARTICIPANT (contactID, groupID, joinDate)

*contactID -> Contact.ID, groupID -> Group.ID*

ADDRESS (**street, zipCode**, city, province, country, main)

EMAIL (**email**, main, contactID)

*contactID -> Contact.ID*

SOCIAL (**supplierName, assEmail**, nickname, welcome)

assEmail -> Email.email

PHONENUMBER (**number**, type, linkedNumber)

ASSIGNEDADDRESS (contactID, addressStr, addressZip)

*contactID -> Contact.ID, (addressStr, addressZip) -> (Address.street, address.zipCode)*

ASSIGNEDPHONE (contactID, number)

*contactID -> Contact.ID, number -> PhoneNumber.number*

**Trigger:**

1. Alla rimozione di un partecipante, se il gruppo ha 0 partecipanti viene automaticamente eliminato

**Procedure:**

**…**