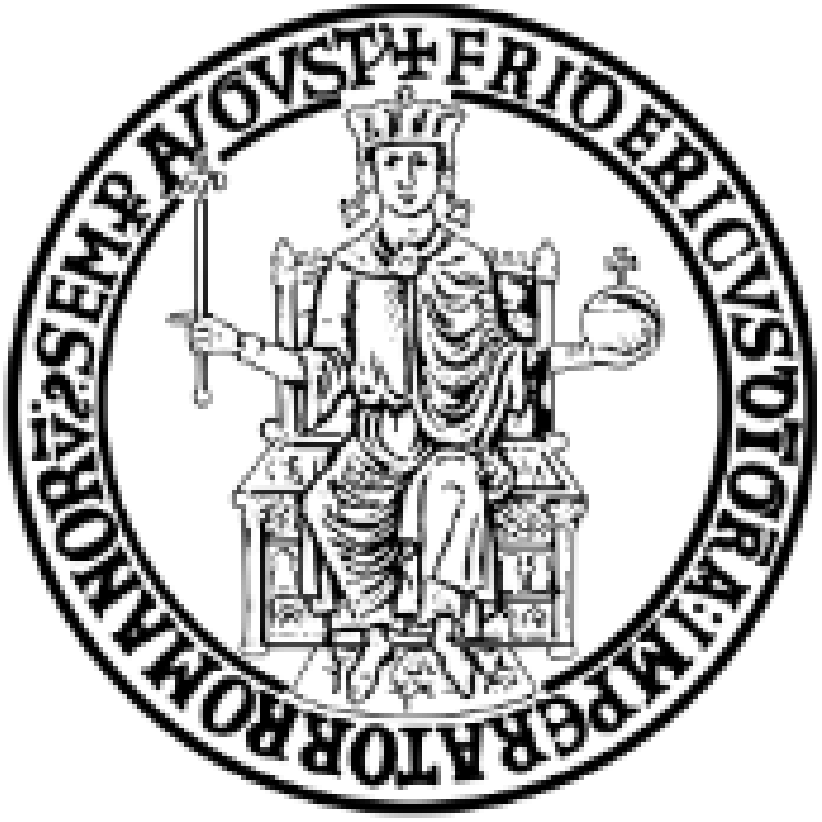


Progettazione e descrizione di una base di dati relazionale per la gestione e memorizzazione di rubriche telefoniche avanzate.

Inserire il nome
Inserire la matricola
Inserire la data

Indice

Capitolo 1

Introduzione

1.1 Analisi del problema

Si vuole progettare una base di dati che permetta all'utente di memorizzare e gestire una rubrica telefonica avanzata. 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.
L'utente potrà quindi 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. Nella rubrica sarà inoltre salvato il log delle chiamate che potranno essere effettuate, ricevute o perse.

1.2 Requisiti identificati

Una rubrica è un insieme di contatti gestiti da un utente, a cui vi potrà accedere tramite e-mail e password. Inoltre, all'utente sarà data la possibilità di scrivere delle note o eventuali memo. La rubrica deve essere in grado di consentire la memorizzazione delle informazioni riguardanti:

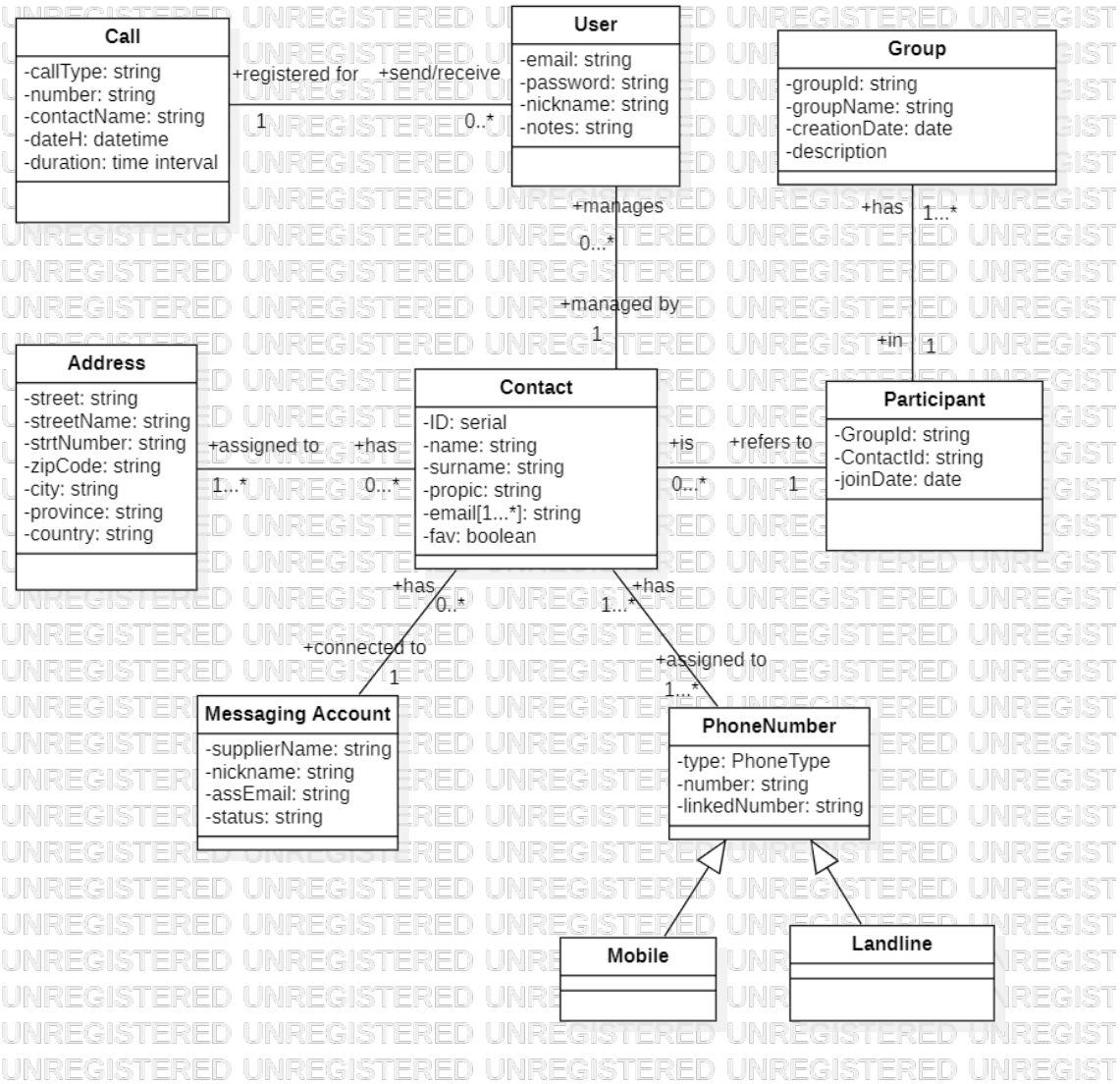
- I contatti, ovvero nome, cognome, una o più e-mail ed eventualmente una foto profilo. Un contatto può essere inserito tra i preferiti e può essere associato ad un indirizzo fisico principale e uno o più secondari, che specifichino via, città, zipcode (CAP) e nazione. Un contatto può avere una sola 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 può corrispondere anche più di un contatto.

- La cronologia delle chiamate effettuate, ricevute e perse.
- I gruppi, ovvero un sottoinsieme dei contatti. Un contatto può essere partecipante a uno o più gruppi, e un gruppo ha uno o più partecipanti. I gruppi avranno un nome, una data di creazione e una descrizione.
- Gli account associati a sistemi di messaging. Per ognuno di questi account verrà memorizzato il fornitore, il nickname, la frase di benvenuto (status) e l'indirizzo e-mail collegato che dovrà necessariamente esistere tra gli account di posta già salvati per il contatto.

Capitolo 2

Progettazione concettuale

2.1 Class Diagram



2.2 Analisi della ristrutturazione del Class Diagram

2.2.1 Analisi delle ridondanze

Nell'analisi non sono evidenti particolari ridondanze, tuttavia 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.

2.2.2 Analisi degli identificativi

Analizzando la relazione Address, 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. Inoltre, per gli stessi motivi si aggiunge una chiave tecnica all'interno della relazione call.

2.2.3 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 Messaging e la nuova relazione Email.

2.2.4 Rimozione degli attributi composti

Nella progettazione concettuale non si è reso necessario l'utilizzo di attributi composti.

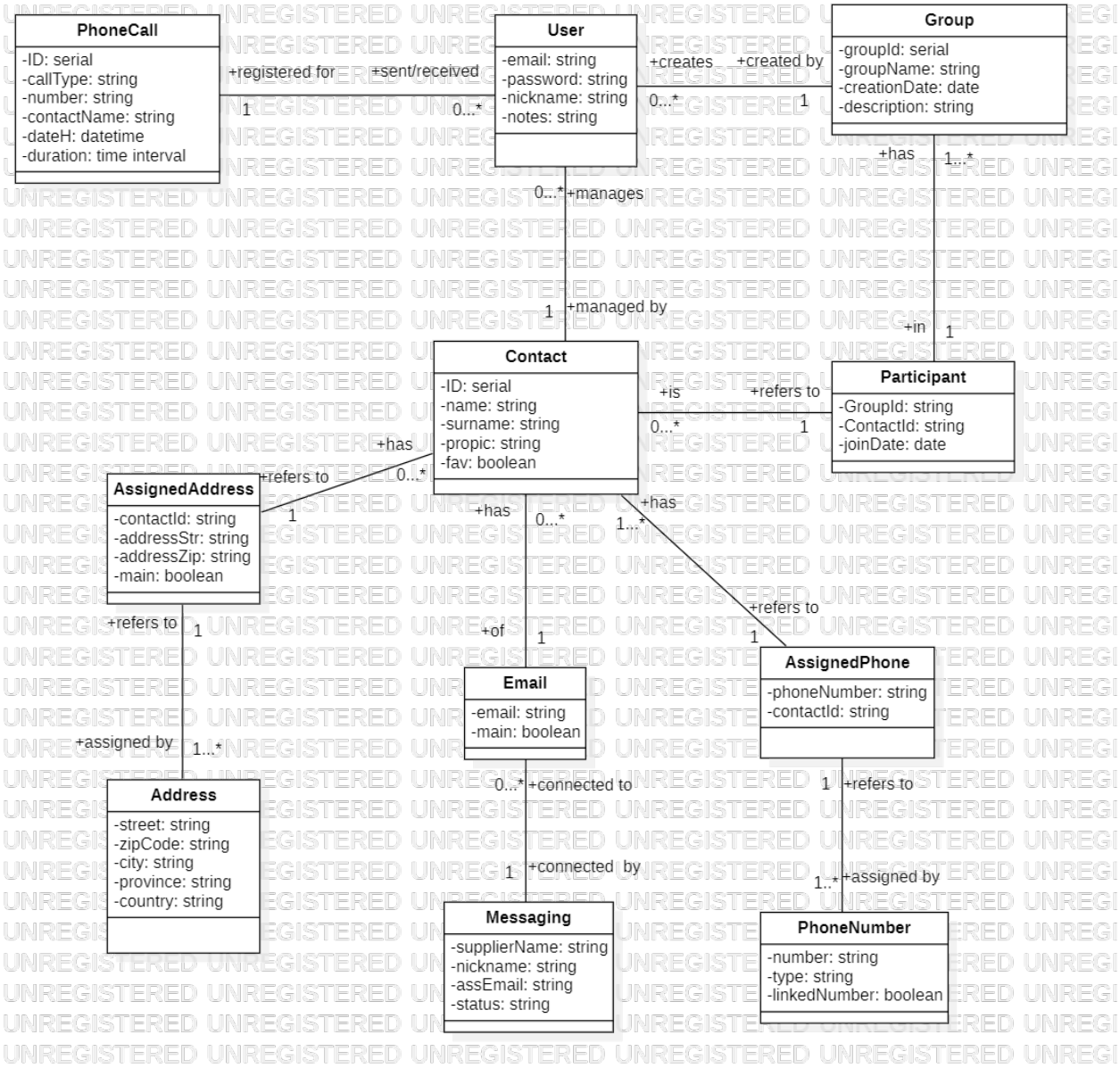
2.2.5 Partizione/Accorpamento delle associazioni

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 Messaging, si rispetta la condizione secondo la quale per ogni account di messaging, la e-mail collegata dovrà necessariamente esistere tra gli account di posta già salvati per il contatto a cui si riferisce.
Vengono aggiunte per completezza le relazioni necessarie ad esprimere le associazioni di tipo molti a molti, ovvero quella tra Contact e Address, e quella tra Contact e PhoneNumber. Le associazioni prenderanno rispettivamente il nome di AssignedAddress e AssignedPhone. Inoltre, in AssignedAddress viene inserita la variabile main per identificare quale indirizzo associato è principale per quel contatto.

2.2.6 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".

2.3 Class Diagram ristrutturato



2.4 Dizionario delle classi

Classe	Descrizione	Attributi
User	Utente proprietario della rubrica	**Email** (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. **Notes** (string): note o memo scritte dall'utente.
Contact	Istanza di un contatto salvato nella rubrica	**ContactID** (serial): chiave tecnica. Codice identificativo di un contatto gestito da un utente. **Name** (string): nome del contatto. **Surname** (string): cognome del contatto. **Propic** (string): percorso sul disco del file contenente l'immagine profilo del contatto. **Favorite** (boolean): indica se il contatto è inserito tra i preferiti.
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.

Email	Indica uno degli indirizzi di posta elettronica associati al contatto.	**Email** (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.
Messaging	Descrive un eventuale account di messaging collegato ad un contatto	**SupplierName** (string): nome del fornitore 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. **Status** (string): frase di benvenuto o stato personale dell'account.
Phone Number	Numero di telefono associato ad uno o più contatti	**PhoneType** (string): tipo di riferimento telefonico. Può essere mobile (MOBILE) oppure fisso (LANDLINE). **PhoneNumber** (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	**GroupID** (serial): 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, formato GG/MM/AAAA. **Description** (string): breve descrizione del gruppo.

Participant	Descrittore dell'associazione tra i partecipanti ad un gruppo ed il gruppo	**GroupID** (integer): ID del gruppo a cui partecipa il contatto. **ContactID** (integer): ID del contatto partecipante. **JoinDate** (date): data di inizio partecipazione al gruppo, formato GG/MM/AAAA.
PhoneCall	Istanza di una chiamata effettuata, ricevuta o persa	**CallID** (serial): chiave tecnica. Codice identificativo di una chiamata archiviata per un utente. **CallType** (string): tipo della chiamata. Può essere effettuata (SENT), ricevuta (ENTERED) o persa (MISSED). **CallNumber** (string): stringa contenente il numero di telefono relativo alla chiamata. **ContactName** (string): nome e cognome del contatto o dei contatti associati al numero di telefono. Vale sconosciuto (UNKNOWN) se il numero non è presente in rubrica. **DateH** (datetime): data e ora della chiamata. **Duration** (time interval): durata della chiamata. Vale 0 se la chiamata è persa.
Assigned Address	Esprime l'assegnazione di un indirizzo fisico ad un contatto e viceversa	**ContactID** (integer): ID del contatto a cui è assegnato l'indirizzo. **AddressStr** (string): nome della strada e numero civico dell'indirizzo da assegnare al contatto. **AddressZip** (string): codice postale dell'indirizzo da assegnare al contatto. **Main** (boolean): indica se è l'indirizzo principale del contatto. Può esistere solo un indirizzo principale per contatto.

Assigned Phone	Esprime l'assegnazione di un numero di telefono ad un contatto e viceversa	**PhoneNumber** (string): numero di telefono da associare al contatto. **ContactID** (integer): ID del contatto da associare al numero di telefono.

2.5 Dizionario delle associazioni

Associazione	Descrizione	Tabelle Coinvolte
User-Contact	Esprime l'appartenenza di un contatto ad una rubrica gestita da un utente	**User**: ruolo manage [0...*] un utente può gestire più contatti all'interno della rubrica. **Contact**: ruolo managed by [1] il contatto è gestito da un utente.
User-Group	Esprime l'appartenenza ad una rubrica di un gruppo creato da un utente	**User**: ruolo creates [0...*] un utente può creare più gruppi all'interno della rubrica. **Group**: ruolo created by [1] il gruppo viene creato dell'utente.
Group-Participant	Indica a quale a quale gruppo si riferisce la partecipazione di un contatto	**Group**: ruolo has [1...*] un gruppo ha uno o più partecipanti. **Participant**: ruolo in [1] un partecipante è presente all'interno di un gruppo.
Contact-Participant	Indica quale contatto è partecipante ad un gruppo	**Contact**: ruolo is [0...*] un contatto può essere partecipante a più gruppi. **Participant**: ruolo refers to [1] un partecipante si riferisce ad un contatto esistente.
Contact-AssignedAddress	Indica quali indirizzi fisici assegnare ad un contatto	**Contact**: ruolo has [0...*] un contatto può avere più indirizzi assegnati. **AssignedAddress**: ruolo refers to [1] l'indirizzo assegnato fa riferimento ad un contatto esistente.

Address-AssignedAddress	Indica quali contatti assegnare ad un indirizzo fisico	**Address**: ruolo assigned by [1...*] un indirizzo è assegnato da una o più istanze di AssignedAddress. **AssignedAddress**: ruolo refers to [1] un indirizzo assegnato fa riferimento ad un indirizzo esistente.
Contact-Email	Indica a quale contatto si riferisce una e-mail	**Contact**: ruolo has [0...*] un contatto può avere più e-mail AssignedAddress. **Email**: ruolo of [1] una e-mail può essere di un solo contatto.
Messaging-Email	Indica a quale e-mail è collegato un account di messaging	**Messaging**: ruolo connected by [1] un account messaging è collegato tramite una e-mail. **Email**: ruolo connected to [0...*] una e-mail può essere collegato a più account di messaging.
Contact-AssignedPhone	Indica quali numeri di telefono assegnare ad un contatto	**Contact**: ruolo has [1...*] un contatto ha uno o più numeri di telefono assegnati. **AssignedPhone**: ruolo refers to [1] un numero di telefono assegnato fa riferimento ad un contatto esistente.
AssignedPhone-PhoneNumber	Indica quali contatti assegnare ad un numero di telefono	**AssignedPhone**: ruolo assigned by [1...*] un numero di telefono è assegnato da una o più istanze di AssignedPhone. **PhoneNumber**: ruolo refers to [1] un numero di telefono assegnato fa riferimento ad un numero di telefono esistente.
PhoneCall-User	Indica per quale utente è stata registrata una chiamata	**PhoneCall**: ruolo registered for [1] una chiamata viene registrata per un utente. **User**: ruolo sent or received [0...*] un utente può effettuare o ricevere più chiamate.

2.6 Dizionario dei vincoli

Capitolo 3

Schema logico

Capitolo 4

Progettazione Fisica

```
CREATE TABLE R_User(
  email VARCHAR(64) NOT NULL PRIMARY KEY,
  passW VARCHAR(64) NOT NULL,
  nickname VARCHAR(64),
  notes VARCHAR(1024)
);

CREATE TABLE Contact(
  contactID SERIAL NOT NULL PRIMARY KEY,
  contactName VARCHAR(32) NOT NULL,
  surname VARCHAR(32) NOT NULL,
  propic VARCHAR(1024) DEFAULT 'C:\Users\Velios\Desktop\Uni\cd\
    defaultpic',
  favorite BOOLEAN NOT NULL DEFAULT 'false',
  r_user VARCHAR(64) NOT NULL,
  CONSTRAINT FK_userContact FOREIGN KEY (r_user) REFERENCES R_User(
    email) ON DELETE CASCADE
);

CREATE TABLE R_Group(
  groupID SERIAL NOT NULL PRIMARY KEY,
  groupName VARCHAR(32) NOT NULL,
  creationDate DATE NOT NULL,
  description VARCHAR(64),
  r_user VARCHAR(64) NOT NULL,
  CONSTRAINT FK_userGroup FOREIGN KEY (r_user) REFERENCES R_User(
    email) ON DELETE CASCADE
);

CREATE TABLE Participant(
  groupID INTEGER NOT NULL,
  contactID INTEGER NOT NULL,
  joinDate DATE NOT NULL,
  CONSTRAINT FK_groupIDParticipant FOREIGN KEY (groupID) REFERENCES
    R_Group(groupID) ON DELETE CASCADE,
  CONSTRAINT FK_contactIDParticipant FOREIGN KEY (contactID)
    REFERENCES Contact(contactID) ON DELETE CASCADE
);

CREATE TABLE Address(
```

```
36    street VARCHAR(1024) NOT NULL,
37    zipCode CHAR(5) NOT NULL,
38    city VARCHAR(32),
39    province CHAR(2),
40    country VARCHAR(32),
41    CONSTRAINT PK_Address PRIMARY KEY (street, zipcode)
42  );
43
44  CREATE TABLE Email(
45    email VARCHAR(1024) NOT NULL PRIMARY KEY,
46    main BOOLEAN NOT NULL DEFAULT 'false',
47    contactID INTEGER,
48    CONSTRAINT FK_contactEmail FOREIGN KEY (contactID) REFERENCES
        Contact(contactID) ON DELETE SET NULL
49  );
50
51  CREATE TABLE Messaging(
52    supplierName VARCHAR(64) NOT NULL,
53    nickname VARCHAR(64) NOT NULL,
54    assEmail VARCHAR(1024) NOT NULL,
55    status VARCHAR(1024),
56    CONSTRAINT PK_Messaging PRIMARY KEY (supplierName, assEmail),
57    CONSTRAINT FK_AssEmail FOREIGN KEY (assEmail) REFERENCES Email(
        email) ON DELETE CASCADE
58  );
59
60  CREATE TABLE PhoneNumber(
61    phoneType VARCHAR(8) NOT NULL,
62    phoneNumber CHAR(10) NOT NULL PRIMARY KEY,
63    linkedNumber BOOLEAN NOT NULL DEFAULT 'false'
64  );
65
66  CREATE TABLE PhoneCall(
67    callID SERIAL NOT NULL PRIMARY KEY,
68    callType VARCHAR(32) NOT NULL,
69    dateH TIMESTAMP NOT NULL,
70    phoneNumber CHAR(10) NOT NULL,
71    duration INTERVAL,
72    contactName VARCHAR(64) DEFAULT 'UNKNOWN',
73    r_user VARCHAR(64) NOT NULL,
74    CONSTRAINT FK_userCall FOREIGN KEY (r_user) REFERENCES R_User(
        email) ON DELETE CASCADE
75  );
76
77  CREATE TABLE AssignedAddress(
78    contactID INTEGER NOT NULL,
79    addressStr VARCHAR(126) NOT NULL,
80    addressZip VARCHAR(5) NOT NULL,
81    main BOOLEAN DEFAULT 'false',
82    CONSTRAINT FK_ADContactID FOREIGN KEY (contactID) REFERENCES
        Contact(contactID) ON DELETE CASCADE,
83    CONSTRAINT FK_ADAddress FOREIGN KEY (addressStr, addressZip)
        REFERENCES Address(street, zipCode) ON DELETE CASCADE
84  );
85
86  CREATE TABLE AssignedPhone(
```

```
87   phoneNumber CHAR(10) NOT NULL,
88   contactID INTEGER NOT NULL,
89   CONSTRAINT FK_APContactID FOREIGN KEY (ContactId) REFERENCES
      Contact(contactID) ON DELETE CASCADE,
90   CONSTRAINT FK_phoneNumber FOREIGN KEY(phoneNumber) REFERENCES
      PhoneNumber(phoneNumber) ON DELETE CASCADE
91 );
```

Listing 4.1: Python example