

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

1	Introduzione	3
1.1	Analisi del problema	3
1.2	Requisiti identificati	3
2	Progettazione concettuale	5
2.1	Class Diagram	5
2.2	Analisi della ristrutturazione del Class Diagram	6
2.2.1	Analisi delle ridondanze	6
2.2.2	Analisi degli identificativi	6
2.2.3	Rimozione degli attributi multipli	6
2.2.4	Rimozione degli attributi composti	6
2.2.5	Partizione/Accorpamento delle associazioni	6
2.2.6	Rimozione delle gerarchie	7
2.3	Class Diagram ristrutturato	8
2.4	Dizionario delle classi	9
2.5	Dizionario delle associazioni	13
2.6	Dizionario dei vincoli	15
3	Schema logico	16
4	Progettazione Fisica	18
4.1	Definizione Tabelle	18
4.2	Definizione dei vincoli	21
4.3	Trigger e Automazioni	22

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:

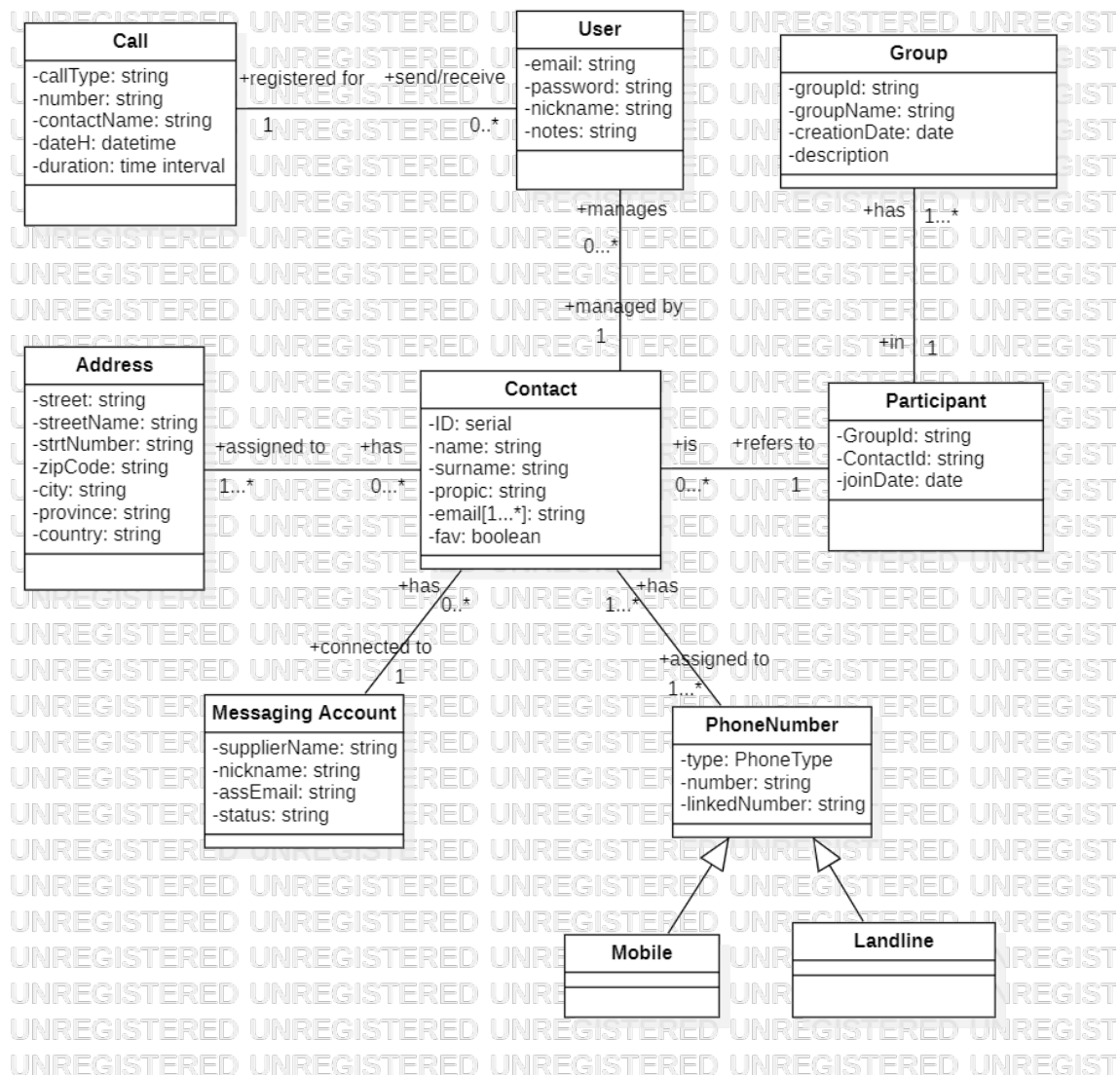
- I contatti, ovvero nome, cognome, una o più e-mail ed eventualmente una foto profilo. Un contatto può essere inserito tra i preferiti e può essere associato ad un indirizzo fisico principale e uno o più secondari, che specifichino via, città, zipcode (CAP) e nazione. Un contatto può avere una sola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

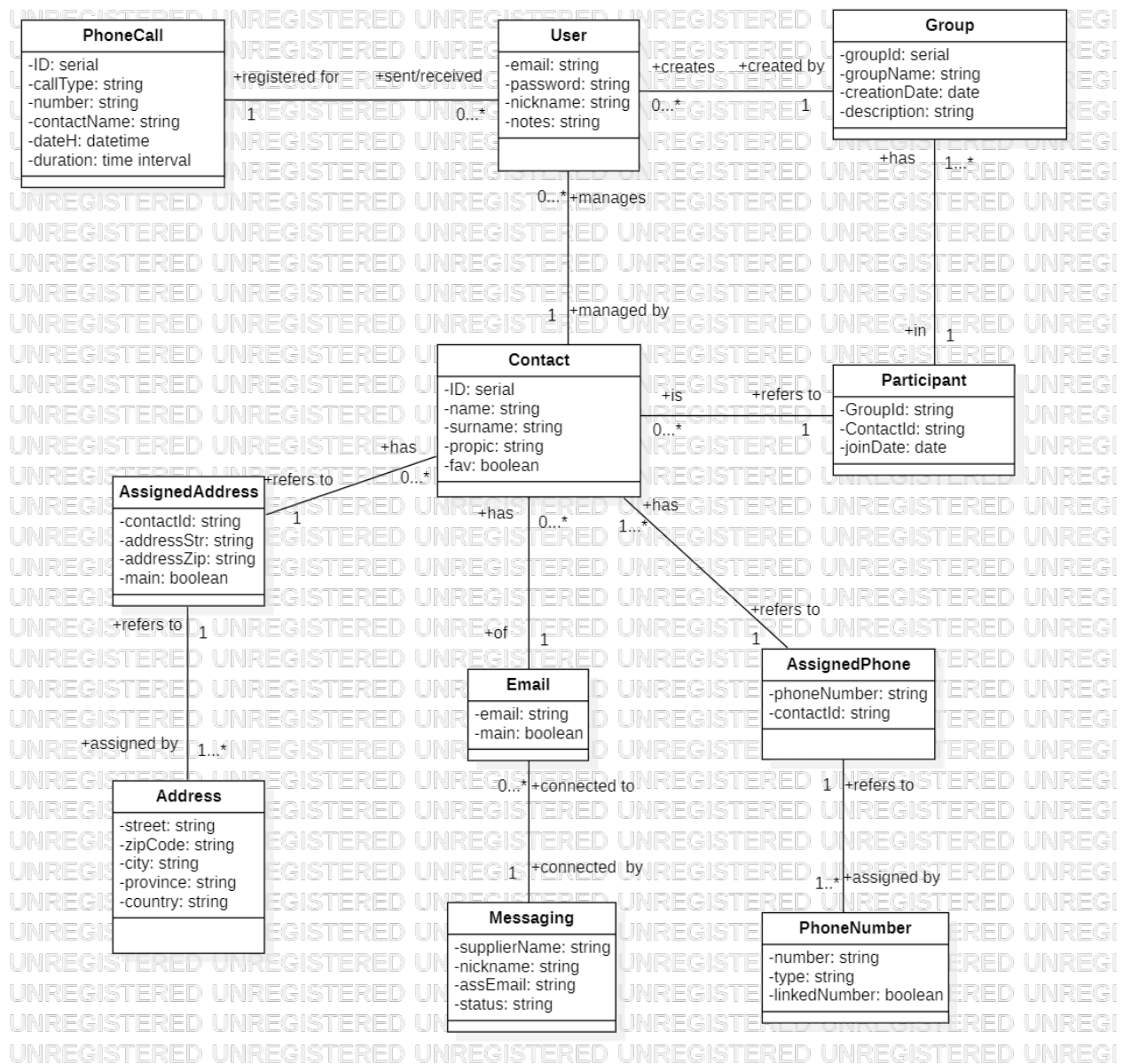
Si è preferito inoltre eliminare l'associazione tra Contact e Social. In questo modo implementando un vincolo di foreign key sull'attributo assEmail della tabella Messaging, si rispetta la condizione secondo la quale per ogni account di messaging, la e-mail collegata dovrà necessariamente esistere tra 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 contatti ed il gruppo a cui appartengono	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	<p>Messaging ruolo connected by [1]: un account messaging è collegato tramite una e-mail.</p> <p>Email ruolo connected to [0...*]: una e-mail può essere collegato a più account di messaging.</p>
Contact-AssignedPhone	Indica quali numeri di telefono assegnare ad un contatto	<p>Contact ruolo has [1...*]: un contatto ha uno o più numeri di telefono assegnati.</p> <p>AssignedPhone ruolo refers to [1]: un numero di telefono assegnato fa riferimento ad un contatto esistente.</p>
AssignedPhone-PhoneNumber	Indica quali contatti assegnare ad un numero di telefono	<p>AssignedPhone ruolo assigned by [1...*]: un numero di telefono è assegnato da una o più istanze di AssignedPhone.</p> <p>PhoneNumber ruolo refers to [1]: un numero di telefono assegnato fa riferimento ad un numero di telefono esistente.</p>
PhoneCall-User	Indica per quale utente è stata registrata una chiamata	<p>PhoneCall ruolo registered for [1]: una chiamata viene registrata per un utente.</p> <p>User ruolo sent or received [0...*]: un utente può effettuare o ricevere più chiamate.</p>

2.6 Dizionario dei vincoli

Nome	Tipo	Descrizione
emailFormat	n-upla	Il formato di un e-mail deve essere così composto: 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.
checkNumberType	Dominio	La variabile Type di PhoneNumber può assumere solo i valori "MOBILE" e "LANDLINE".
checkCallType	Dominio	La variabile Type di Call può assumere solo i valori "SENT", "ENTERED" o "MISSED".
PK_messaging	Intra-relazionale (primary key)	Può esistere un solo tipo di account social collegato ad una email.
distinctEmail	Intra-relazionale	Non possono esistere due contatti associati alla stessa e-mail.
FK_messagingEmail	Inter-relazionale (foreign key)	L'indirizzo e-mail di un account di messaging di un contatto deve esistere necessariamente tra gli account di posta già salvati per il contatto.
uniqueMainEmail	Inter-relazionale	Ad un contatto può essere associata una sola e-mail di tipo main.
uniqueMainAddress	Inter-relazionale	Ad un contatto può essere associato un solo indirizzo fisico di tipo main.
uniqueLinked Number	Inter-relazionale	Ad un contatto può essere associato un solo numero di telefono MOBILE di reindirizzamento, ed un solo numero LANDLINE di reindirizzamento.
checkContact Numbers	Inter-relazionale	Ogni contatto deve avere almeno un telefono fisso e un telefono mobile.
timeConsistency Group	Inter-relazionale	La joinDate di un partecipante in un gruppo deve essere successiva alla data di creazione del gruppo.

Capitolo 3

Schema logico

*alcuni nomi sono stati cambiati poiché parole chiave di postgresql

R_USER	(<u>email</u> , passW, nickName, notes)
CONTACT	(<u>contactID</u> , contactName, surname, propic, favorite, <u>r_user</u>) $r_user \rightarrow R_User.email$
R_GROUP	(<u>groupID</u> , groupName, creationDate, Description, <u>r_user</u>) $r_user \rightarrow R_User.email$
PARTICIPANT	(<u>contactID</u> , <u>groupID</u> , joinDate) $contactID \rightarrow Contact.contactID$ $groupID \rightarrow Group.groupID$
ADDRESS	(<u>street</u> , <u>zipCode</u> , city, province, country)
EMAIL	(<u>email</u> , main, <u>contactID</u>) $contactID \rightarrow Contact.contactID$
MESSAGING	(<u>supplierName</u> , <u>assEmail</u> , nickName, status) $assEmail \rightarrow Email.email$
PHONENUMBER	(<u>phoneNumber</u> , phoneType, linkedNumber)
PHONECALL	(<u>callID</u> , callType, phoneNumber, contactName, dateH, du- ration, <u>r_user</u>) $r_user \rightarrow R_User.email$
ASSIGNEDADDRESS	(<u>contactID</u> , <u>addressStr</u> , <u>addressZip</u> , main) $contactID \rightarrow Contact.contactID$ $(addressStr, addressZip) \rightarrow (Address.street, Ad-dress.zipCode)$

ASSIGNEDPHONE	$(\underline{\underline{\text{contactID}}}, \underline{\underline{\text{phoneNumber}}})$ $\text{contactID} \rightarrow \text{Contact.contactID}$ $\text{phoneNumber} \rightarrow \text{PhoneNumber.phoneNumber}$
---------------	--

Capitolo 4

Progettazione Fisica

4.1 Definizione Tabelle

```
1 CREATE TABLE R_User(  
2   email VARCHAR(64) NOT NULL PRIMARY KEY,  
3   passW VARCHAR(64) NOT NULL,  
4   nickname VARCHAR(64),  
5   notes VARCHAR(1024)  
6 );  
7  
8 CREATE TABLE Contact(  
9   contactID SERIAL NOT NULL PRIMARY KEY,  
10  contactName VARCHAR(32) NOT NULL,  
11  surname VARCHAR(32) NOT NULL,  
12  propic VARCHAR(1024) DEFAULT 'C:\Users\Velios\Desktop\Uni\cd\  
    defaultpic',  
13  favorite BOOLEAN NOT NULL DEFAULT 'false',  
14  r_user VARCHAR(64) NOT NULL,  
15  CONSTRAINT FK_userContact FOREIGN KEY (r_user) REFERENCES R_User(  
    email) ON DELETE CASCADE  
16 );  
17  
18 CREATE TABLE R_Group(  
19  groupID SERIAL NOT NULL PRIMARY KEY,  
20  groupName VARCHAR(32) NOT NULL,  
21  creationDate DATE NOT NULL,  
22  description VARCHAR(64),  
23  r_user VARCHAR(64) NOT NULL,  
24  CONSTRAINT FK_userGroup FOREIGN KEY (r_user) REFERENCES R_User(  
    email) ON DELETE CASCADE  
25 );  
26  
27 CREATE TABLE Participant(  
28  groupID INTEGER NOT NULL,  
29  contactID INTEGER NOT NULL,  
30  joinDate DATE NOT NULL DEFAULT CURRENT_DATE  
31  CONSTRAINT FK_groupIDParticipant FOREIGN KEY (groupID) REFERENCES  
    R_Group(groupID) ON DELETE CASCADE,  
32  CONSTRAINT FK_contactIDParticipant FOREIGN KEY (contactID)  
    REFERENCES Contact(contactID) ON DELETE CASCADE
```

```

33 );
34
35 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
49         Contact(contactID) ON DELETE SET NULL
50 );
51
52 CREATE TABLE Messaging(
53     supplierName VARCHAR(64) NOT NULL,
54     nickname VARCHAR(64) NOT NULL,
55     assEmail VARCHAR(1024) NOT NULL,
56     status VARCHAR(1024),
57     -- Implementazione del vincolo PK_messaging
58     CONSTRAINT PK_Messaging PRIMARY KEY (supplierName, assEmail),
59     -- Implementazione del vincolo FK_messagingEmail
60     CONSTRAINT FK_AssEmail FOREIGN KEY (assEmail) REFERENCES Email(
61         email) ON DELETE CASCADE
62 );
63
64 CREATE TABLE PhoneNumber(
65     phoneType VARCHAR(8) NOT NULL,
66     phoneNumber CHAR(10) NOT NULL PRIMARY KEY,
67     linkedNumber BOOLEAN NOT NULL DEFAULT 'false'
68 );
69
70 CREATE TABLE PhoneCall(
71     callID SERIAL NOT NULL PRIMARY KEY,
72     callType VARCHAR(32) NOT NULL,
73     dateH TIMESTAMP NOT NULL,
74     phoneNumber CHAR(10) NOT NULL,
75     duration INTERVAL,
76     contactName VARCHAR(64) DEFAULT 'UNKNOWN',
77     r_user VARCHAR(64) NOT NULL,
78     CONSTRAINT FK_userCall FOREIGN KEY (r_user) REFERENCES R_User(
79         email) ON DELETE CASCADE
80 );
81
82 CREATE TABLE AssignedAddress(
83     contactID INTEGER NOT NULL,
84     addressStr VARCHAR(126) NOT NULL,
85     addressZip VARCHAR(5) NOT NULL,
86     main BOOLEAN DEFAULT 'false',
87     CONSTRAINT FK_ADContactID FOREIGN KEY (contactID) REFERENCES
88         Contact(contactID) ON DELETE CASCADE,

```

```

85     CONSTRAINT FK_ADAddress FOREIGN KEY (addressStr, addressZip)
      REFERENCES Address(street, zipCode) ON DELETE CASCADE
86 );
87
88 CREATE TABLE AssignedPhone(
89     phoneNumber CHAR(10) NOT NULL,
90     contactID INTEGER,
91     CONSTRAINT FK_APContactID FOREIGN KEY (ContactId) REFERENCES
      Contact(contactID) ON DELETE SET NULL,
92     CONSTRAINT FK_phoneNumber FOREIGN KEY(phoneNumber) REFERENCES
      PhoneNumber(phoneNumber) ON DELETE CASCADE
93 );

```

Listing 4.1: PostgreSQL

4.2 Definizione dei vincoli

4.3 Trigger e Automazioni

```
1  -- Alla rimozione di un contatto da un Gruppo, se quel Gruppo non
   ha pi partecipanti viene eliminato
2
3  CREATE FUNCTION AutoDeleteGroup()
4  RETURNS TRIGGER AS $AutoDeleteGroup$
5  BEGIN
6      IF NOT EXISTS (SELECT * FROM Participant as Par WHERE OLD.
   groupID = Par.groupID)
7      THEN
8          DELETE FROM R_Group as rg
9          WHERE OLD.groupID = rg.groupID;
10     END IF;
11     RETURN NEW;
12     END;
13 $AutoDeleteGroup$ LANGUAGE plpgsql;
14
15 CREATE TRIGGER AutoDeleteGroup
16 AFTER DELETE ON Participant
17 FOR EACH ROW
18 EXECUTE PROCEDURE AutoDeleteGroup();
19
20
21
22
23
24 -- La seguente procedura prende in input un numero telefonico e se
   non ci sono contatti associati ad esso lo elimina
25
26 CREATE PROCEDURE AutoDeletePhone(PNumber char(10))
27 LANGUAGE plpgsql AS $AutoDeletePhone$
28 BEGIN
29     IF NOT EXISTS (SELECT * FROM AssignedPhone as AP WHERE PNumber
   = AP.phoneNumber and AP.contactID <> NULL)
30     THEN
31         DELETE FROM PhoneNumber as PN
32         WHERE PN.phoneNumber = PNumber;
33     END IF;
34     END;
35 $AutoDeletePhone$;
36
37
38 -- il seguente trigger chiama la procedura AutoDeletePhone alla
   rimozione di un contatto dalla tabella AssignedPhone
39
40 CREATE FUNCTION CallDeletePhone()
41 RETURNS TRIGGER AS $CallDeletePhone$
42 BEGIN
43     CALL AutoDeletePhone(OLD.phonenumber);
44     RETURN NEW;
45     END;
46 $CallDeletePhone$ LANGUAGE plpgsql;
47
48 CREATE TRIGGER CallDeletePhone
```

```

49 AFTER UPDATE ON AssignedPhone
50 FOR EACH ROW
51 WHEN (NEW.contactID = NULL)
52 EXECUTE PROCEDURE CallDeletePhone();
53 -- NOTA: Quando un contatto viene eliminato, la foreign key
    a l l interno della tabella AssignedPhone viene impostata a NULL
    (si veda tabella di definizione) per evitare conflitti con i
    trigger che implementano i vincoli
54
55
56
57
58
59 -- Alla rimozione di un assegnazione di un indirizzo ad un contatto
    , se q u e l l indirizzo non ha altre assegnazioni viene eliminato
60
61 CREATE FUNCTION AutoDeleteAddress()
62 RETURNS TRIGGER AS $AutoDeleteAddress$
63 BEGIN
64     IF NOT EXISTS (SELECT * FROM AssignedAddress as AA WHERE OLD.
        addressStr = AA.addressStr and OLD.addressZip = AA.addressZip)
65     THEN
66         DELETE FROM Address
67         WHERE Address.street = OLD.addressStr and Address.zipcode =
            OLD.addressZip;
68     END IF;
69     RETURN NEW;
70 END;
71 $AutoDeleteAddress$ LANGUAGE plpgsql;
72
73 CREATE TRIGGER AutoDeleteAddress
74 AFTER DELETE ON AssignedAddress
75 FOR EACH ROW
76 EXECUTE PROCEDURE AutoDeleteAddress();
77
78
79
80
81
82 -- Alla creazione di un Gruppo, la data di creazione viene
    impostata a quella corrente
83
84 CREATE FUNCTION SetGroupDate()
85 RETURNS TRIGGER AS $SetGroupDate$
86 BEGIN
87     UPDATE R_Group as RG
88     SET NEW.creationDate = CURRENT_DATE
89     WHERE NEW.groupID = RG.groupID;
90     RETURN NEW;
91 END;
92 $SetGroupDate$ LANGUAGE plpgsql;
93
94 CREATE TRIGGER SetGroupDate
95 AFTER INSERT ON R_Group
96 FOR EACH ROW
97 EXECUTE PROCEDURE SetGroupDate();

```

```

98
99
100
101
102
103 -- La seguente procedura riceve in ingresso un numero telefonico e
    un utente, cerca il nome e il cognome dei contatti associati ad
    essi, e li scrive nella colonna contactName di PhoneCall dove il
    numero telefonico corrisponde. Se non ci sono contatti
    associati al numero telefonico, scrive      Unknown
104
105 CREATE PROCEDURE UpdCallContactName(PNumber char(10), usr R_User.
    email%type)
106 LANGUAGE plpgsql AS $UpdCallContactName$
107 DECLARE
108     resultString varchar(1024) := '';
109     namesur CURSOR FOR (SELECT contactName, surname FROM PhoneNumber
        as PN, AssignedPhone as AP, Contact as Con WHERE PN.phoneNumber
        = AP.phoneNumber and AP.contactID = Con.contactID and PN.
        phoneNumber = PNumber and Con.r_user = usr);
110 BEGIN
111     FOR nome IN namesur LOOP
112         resultString := resultString || nome.contactName || ' ' || nome
            .surname || ', ';
113     END LOOP;
114     resultString := rtrim(resultString, ' ');
115     IF resultString = '' THEN
116         resultString := 'Unknown';
117     END IF;
118     UPDATE PhoneCall as PC
119     SET contactName = resultString
120     WHERE PC.phoneNumber = PNumber;
121 END;
122 $UpdCallContactName$
123
124
125 -- Chiamata della procedura UpdCallContactName a l l inserimento di
    una nuova riga nella tabella PhoneCall
126
127 CREATE FUNCTION UpdPhoneCall()
128 RETURNS TRIGGER AS $UpdPhoneCall$
129 BEGIN
130     CALL UpdCallContactName(NEW.phoneNumber, NEW.r_user);
131     RETURN NEW;
132 END;
133 $UpdPhoneCall$ LANGUAGE plpgsql;
134
135 CREATE TRIGGER UpdPhoneCall
136 AFTER INSERT ON PhoneCall
137 FOR EACH ROW
138 EXECUTE PROCEDURE UpdPhoneCall();
139
140
141 -- Chiamata della procedura UpdCallContactName a l l assegnazione
    di un numero di telefono ad un contatto
142

```



```

143 CREATE FUNCTION UpdCallFromAP()
144 RETURNS TRIGGER AS $UpdCallFromAP$
145 DECLARE
146     currentUser varchar(10);
147 BEGIN
148     SELECT Con.r_user into currentUser
149     FROM Contact as Con
150     WHERE NEW.contactID = Con.contactID;
151     CALL UpdCallContactName(NEW.phoneNumber, currentUser);
152     RETURN NEW;
153 END;
154 $UpdCallFromAP$ LANGUAGE plpgsql;
155
156 CREATE TRIGGER UpdCallFromAP
157 AFTER INSERT or DELETE ON AssignedPhone
158 FOR EACH ROW
159 EXECUTE PROCEDURE UpdCallFromAP();

```

Listing 4.2: PostgreSQL