UNIVERSITÀ DEGLI STUDI DI NAPOLI 'FEDERICO II'
Scuola Politecnica e delle Scienze di base
Dipartimento di Ingegneria Elettrica e Tecnologie
dell'Informazione

Corso di Laurea in Informatica
Progetto di Basi di Dati
AA. 2021/2022

Progetazzione e sviluppo di una base di dati relazionale per un sistema di gestione di una rubrica telefonica avanzata

Gianluigi Mercorio
N86003834

Michele Pio Loreto
N86003803

27-01-2022

Questa pagina è stata intenzionalmente lasciata in bianco.

Indice

Capitolo 1

Descrizione del progetto

1.1 Analisi del problema

Si provvederà alla progettazione ed allo sviluppo di una base di dati dedicata alla descrizione, memorizzazione e gestione di una rubrica telefonica avanzata.

La base di dati conterrà i dati relativi a diversi contatti quali le generalità, il numero di telefono fisso e il numero di telefono mobile, l'indirizzo di residenza ed eventuali indirizzi di posta elettronica ad esso associati.

Il sistema permetterà di organizzare tra loro contatti in uno o più gruppi, ciascuno dei quali verrà definito in termini di Categoria che essa rispecchia e di una breve descrizione che agevola la comprensione dell'utilizzo che ne verrà fatto.

Al contatto verrà associato un insieme, anche vuoto, di indirizzi di posta elettronica definiti in termini di indirizzo e dominio ed unici per ciascun contatto; ad essi possono essere associati in modo univoco account di messagging che verranno memorizzati in modo da tenere traccia del nome e del cognome della persona a cui fanno capo, della frase di benvenuto che esplica eventuali informazioni dichiarate dal proprietario dell'account ed identificati per Nickname, che saranno unici per ciascuna mail e fornitori associati.
Quest'ultimi verranno memorizzati ed identificati attraverso il nome della piattaforma di messagging, la propria casa produttrice e la categoria del servizio offerto.

Ciascun contatto può tenere traccia di eventuali indirizzi e numeri di telefono secondari ed il sistema consentirà di registrare eventuali reindirizzamenti telefonici tra un mobile ed un fisso.

1.2 Requisiti identificati

Sono state individuate le seguenti entità:

1. **CONTATTO** che identifica e tiene traccia dei dati di un contatto

2. **GRUPPO** che identifica e tiene traccia dei gruppi ai quali possono afferire uno o più contatti

3. **TELEFONO FISSO** che identifica e registra i numeri di telefono fisso che un contatto possiede

4. **TELEFONO MOBILE** che identifica e registra i numeri di telefono mobile posseduti da un contatto

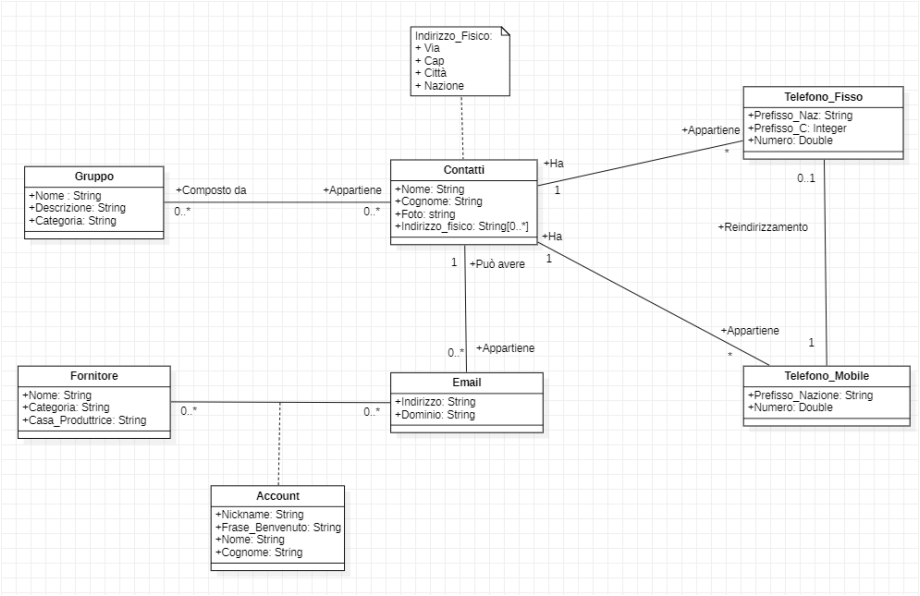
5. **EMAIL** che identifica e memorizza le mail che un contatto possiede

6. **FORNITORE** che identifica e registra le piattaforme di messagging eventualmente legate alle mail

Capitolo 2

Progettazione concettuale

2.1 Class Diagram



2.2 Analisi della ristrutturazione del Class Diagram

2.2.1 Analisi delle ridondanze

L'unica informazione ridondante presente è quella relativa agli attributi 'Nome' e 'Cognome' nella classe ACCOUNT, in quanto il valore di tali attributi può essere ricavato da una doppia join tra le classi CONTATTO - EMAIL - ACCOUNT.

2.2.2 Analisi degli identificativi

Per le classi CONTATTO, INDIRIZZO, EMAIL, TELEFONO_FISSO, TELEFONO_MOBILE si è scelto di utilizzare delle chiavi surrogate, in modo da evitare l'utilizzo di chiavi primarie composte;
Per convenzioni tali chiavi inizieranno con il prefisso 'Cod_'.

Per la classe ACCOUNT si è scelta come chiave primaria l'attributo 'Nickname'.

Per la classe GRUPPO si è scelta come chiave primaria l'attributo 'Nome'.

Per la classe FORNITORE si è scelta come chiave primaria l'attributo 'Nome'.

2.2.3 Rimozione degli attributi multipli e composti

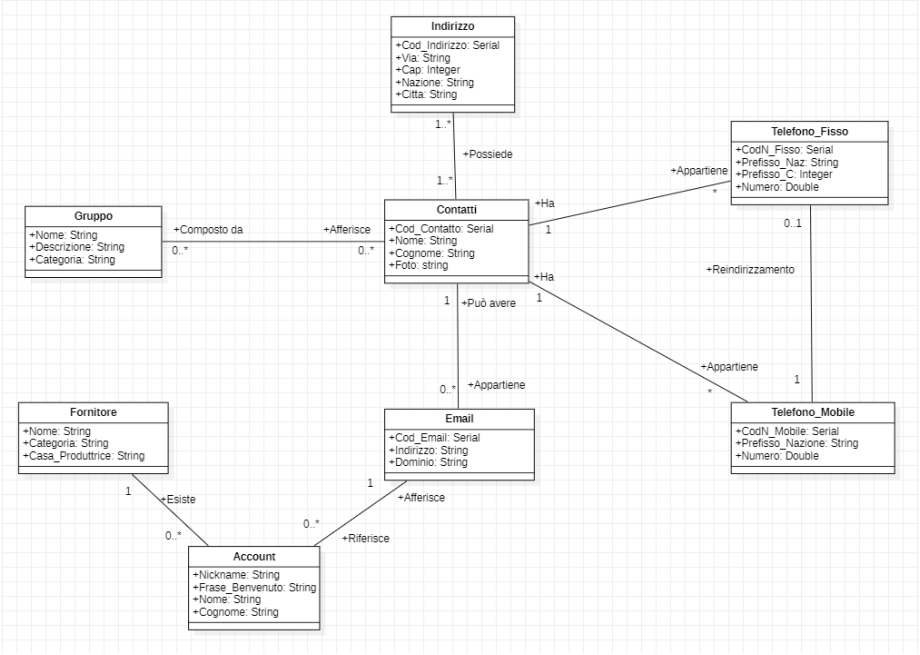
E' necessario gestire l'attributo composto 'Indirizzo' della classe CONTATTO trasformandolo in una classe INDIRIZZO a sé stante. Quest'ultima è messa in relazione con la classe CONTATTO, con grado di cardinalità molti a molti.

2.2.4 Modellazione delle classi di relazione

Si è presa la decisione di gestire la classe di relazione ACCOUNT che intercorre tra le classi EMAIL e FORNITORE come una classe a sé stante in relazione con quest'ultime:

- ACCOUNT-FORNITORE con cardinalità uno a molti;
- ACCOUNT-EMAIL con cardinalità uno a molti.

2.3 Class Diagram ristrutturato



2.4 Dizionario delle classi

2.4.1 Classe CONTATTO

DESCRIZIONE:
Descrittore di un contatto appartenente ad una rubrica

ATTRIBUTI:
• Cod_Contatto (SERIAL) = chiave primaria, identifica univocamente ogni istanza della classe CONTATTO
• Nome (STRING) = nome associato al contatto
• Cognome (STRING) = cognome associato al contatto
• Foto (STRING) = percorso della foto salvata in memoria, associata al contatto

2.4.2 Classe INDIRIZZO

DESCRIZIONE:
Descrittore di un indirizzo appartenente ad un contatto

ATTRIBUTI:
• Cod_Indirizzo (SERIAL) = chiave primaria, identifica univocamente ogni istanza della classe INDIRIZZO
• Via (STRING) = via associata ad un indirizzo
• Cap (INTEGER) = CAP associato ad un indirizzo
• Nazione (STRING) = nazione dell'indirizzo
• Nazione (STRING) = nazione dell'indirizzo

2.4.3 Classe GRUPPO

DESCRIZIONE:
Descrittore di un gruppo di contatti

ATTRIBUTI:
• Nome (STRING) = chiave primaria, identifica univocamente ogni istanza della classe GRUPPO
• Descrizione (STRING) = descrizione facoltativa del gruppo
• Categoria (STRING) = categoria associata ad un gruppo

2.4.4 Classe EMAIL

DESCRIZIONE:
Descrittore di un'email appartenente ad un contatto

ATTRIBUTI:
• Cod_Email (SERIAL) = chiave tecnica, identifica univocamente ogni istanza della classe EMAIL
• Indirizzo (STRING) = indirizzo associato all' Email
• Dominio (STRING) = dominio associato all' Email

2.4.5 Classe ACCOUNT

DESCRIZIONE:
Descrittore di un account appartenente ad un contatto

ATTRIBUTI:
• Nickname (STRING) = chiave primaria, identifica univocamente ogni istanza della classe ACCOUNT
• Frase_Benvenuto (STRING) = frase di benvenuto facoltativa dell'account
• Nome (STRING) = nome associato all'account
• Cognome (STRING) = cognome associato all'account

2.4.6 Classe FORNITORE

DESCRIZIONE:
Descrittore di un fornitore

ATTRIBUTI:
• Nome (STRING) = chiave primaria, identifica univocamente ogni istanza della classe FORNITORE
• Categoria (STRING) = categoria associata ad un fornitore
• Casa_Produttrice (STRING) = casa produttrice associata ad un fornitore

2.4.7 Classe TELEFONO_FISSO

DESCRIZIONE:
Descrittore di un telefono fisso appartenente ad un contatto

ATTRIBUTI:
• CodN_Fisso (SERIAL)= chiave primaria, identifica univocamente ogni istanza della classe TELEFONO_FISSO
• Prefisso_Naz (STRING) = prefisso nazionale associato al telefono fisso
• Prefisso_C(INTEGER) = prefisso di citta associato al telefono fisso
• Numero (DOUBLE) = numero del telefono fisso

2.4.8 Classe TELEFONO_MOBILE

DESCRIZIONE:
Descrittore di un telefono mobile appartenente ad un contatto

ATTRIBUTI:
• CodN_Mobile (SERIAL) = chiave primaria, identifica univocamente ogni istanza della classe TELEFONO_MOBILE
• Prefisso_Naz (STRING) = prefisso nazionale associato al telefono mobile
• Numero (INTEGER) = numero del telefono mobile

2.5 Dizionario delle associazioni

2.5.1 Associazione POSSIEDE

DESCRIZIONE:
Tiene traccia del possesso da parte di un contatto dei suoi indirizzi

CLASSI COINVOLTE:
• CONTATTO[1..*] ruolo (possiede) indica il contatto che possiede l'indirizzo
• INDIRIZZO[1..*] ruolo (posseduto) indica l'indirizzo posseduto dal contatto

2.5.2 Associazione AFFERISCE

DESCRIZIONE:
Tiene traccia della partecipazione di un contatto ad un gruppo

CLASSI COINVOLTE:
• CONTATTO[0..*] ruolo (afferisce) indica l'afferenza di un contatto ad un gruppo
• GRUPPO [0..*] ruolo (composto da) indica i contatti di cui è composto un gruppo

2.5.3 Associazione APPARTIENE

DESCRIZIONE:
Tiene traccia delle eventuali email che appartengono ad un contatto

CLASSI COINVOLTE:
• EMAIL[0..*] ruolo (appartiene) indica l'appartenenza dell'email ad un contatto
• CONTATTO[1] ruolo (può avere) indica che un contatto può o non può possedere un'email

2.5.4 Associazione HA [CONTATTO - TELEFONO_FISSO]

DESCRIZIONE:
Tiene traccia dei numeri di telefono fissi posseduti da un contatto

CLASSI COINVOLTE:
• CONTATTO [1] ruolo (ha) indica che un contatto possiede almeno un telefono fisso
• TELEFONO_FISSO [*] (appartiene) descrive l'appartenenza di un telefono fisso ad un contatto

2.5.5 Associazione HA [CONTATTO - TELEFONO_MOBILE]

DESCRIZIONE:
Tiene traccia dei numeri di telefono mobile posseduti da un contatto

CLASSI COINVOLTE:
• CONTATTO[1] ruolo (ha) indica che un contatto possiede almeno un telefono mobile
• TELEFONO_MOBILE[*] ruolo (appartiene) indica l'appartenenza di un telefono mobile ad un contatto

2.5.6 Associazione REINDIRIZZAMENTO

DESCRIZIONE:
Tiene traccia dei reindirizzamenti telefonici fra telefono fissi e telefoni mobili

CLASSI COINVOLTE:
• TELEFONO_FISSO [0..1] ruolo (reindirizzato) indica il reindirizzamento di un telefono fisso verso un telefono mobile
• TELEFONO_MOBILE [1] ruolo (reindirizzato) indica l'eventuale reindirizzamento di un telefono mobile verso un telefono fisso

2.5.7 Associazione RIFERISCE

DESCRIZIONE:
Tiene traccia delle email che riferiscono ad un account

CLASSI COINVOLTE:
• ACCOUNT[0..*] ruolo (riferisce) indica l'account che fa riferimento ad un'email
• EMAIL[1] ruolo (afferisce) indica l'email che appartiene all'account

2.5.8 Associazione ESISTE

DESCRIZIONE:
Tiene traccia degli account esistenti per un fornitore

CLASSI COINVOLTE:
• ACCOUNT[0..*] ruolo(esiste) indica la possibile esistenza di un account per un fornitore
• FORNITORE[1] ruolo(esiste) indica l'esistenza di un fornitore

2.6 Dizionario dei vincoli

2.6.1 Vincolo PK_CONTATTO

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella CONTATTO

2.6.2 Vincolo PK_INDIRIZZO

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella INDIRIZZO

2.6.3 Vincolo PK_GRUPPO

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella GRUPPO

2.6.4 Vincolo PK_EMAIL

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella EMAIL

2.6.5 Vincolo PK_FORNITORE

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella FORNITORE

2.6.6 Vincolo PK_ACCOUNT

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella ACCOUNT

2.6.7 Vincolo PK_TELEFONOM

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella TELEFONO_MOBILE

2.6.8 Vincolo PK_TELEFONOF

TIPO:
Intrarelazionale

DESCRIZIONE:
Definizione chiave primaria della tabella TELEFONO_FISSO

2.6.9 Vincolo FK_POSSIEDE_INDIRIZZO

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Indirizzo) della tabella POSSIEDE

2.6.10 Vincolo FK_POSSIEDE_CONTATTO

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Contatto) della tabella POSSIEDE

2.6.11 Vincolo FK_AFFERENZA_CONTATTO

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Contatto) della tabella AFFERENZA

2.6.12 Vincolo FK_AFFERENZA_GRUPPO

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Nome_Gruppo) della tabella AFFERENZA

2.6.13 Vincolo FK_EMAIL_CONTATTO

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Contatto) della tabella EMAIL

2.6.14 Vincolo FK_ACCOUNT_NOME_F

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Nome_Fornitore) della tabella ACCOUNT

2.6.15 Vincolo FK_ACCOUNT_EMAIL

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Email) della tabella ACCOUNT

2.6.16 Vincolo FK_TELEFONOF

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Contatto) della tabella TELEFONO_FISSO

2.6.17 Vincolo FK_TELEFONOM

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(Cod_Contatto) della tabella TELEFONO_MOBILE

2.6.18 Vincolo FK_REINDIRIZZAMENTO_TF

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(CodN_Fisso) della tabella REINDIRIZZAMENTO

2.6.19 Vincolo FK_REINDIRIZZAMENTO_TM

TIPO:
Interrelazionale

DESCRIZIONE:
Definizione chiave esterna(CodN_Mobile) della tabella REINDIRIZZAMENTO

2.6.20 Vincolo U_EMAIL

TIPO:
Intrarelazionale

DESCRIZIONE:
Un' email deve far riferimento ad un unico contatto

2.6.21 Vincolo U_ACCOUNT

TIPO:
Intrarelazionale

DESCRIZIONE:
Ad un account deve far riferimento un'unica email ed un unico fornitore

2.6.22 Vincolo DOMINIO_CORRETTO

TIPO:
Intrarelazionale

DESCRIZIONE:
Formattazione corretta del dominio dell'email

2.6.23 Vincolo INDIRIZZO_CORRETTO

TIPO:
Intrarelazionale

DESCRIZIONE:
Formattazione corretta dell'indirizzo dell'email

2.6.24 Vincolo PREFISSO_NAZ_CORRETTO_TM

TIPO:
Intrarelazionale

DESCRIZIONE:
Formattazione corretta del prefisso nazionale del telefono mobile

2.6.25 Vincolo PREFISSO_NAZ_CORRETTO_TF

TIPO:
Intrarelazionale

DESCRIZIONE:
Formattazione corretta del prefisso nazionale del telefono fisso

Capitolo 3

Schema logico

In questo capitolo viene documentata la traduzione dello schema concettuale (ristrutturato) in uno schema logico. Negli schemi relazionali che seguiranno le chiavi primarie sono indicate con una singola sottolineatura mentre le chiavi esterne con una doppia sottolineatura.

3.1 Traduzione

CONTATTO (Cod_Contatto, Nome, Cognome, Foto)

INDIRIZZO (Cod_Indirizzo, Via, CAP, Nazione, Città)

POSSIEDE (Cod_Indirizzo, Cod_Contatto)

Cod_Indirizzo → INIDIRIZZO.Cod_Indirizzo
Cod_Contatto → CONTATTO.Cod_Contatto

EMAIL (Cod_Email, Indirizzo, Dominio, Cod_Contatto)

Cod_Contatto → CONTATTO.Cod_Contatto

FORNITORE (Nome, Categoria, Casa_Produttrice)

ACCOUNT (Nickname, Frase_Benvenuto, Nome, Cognome, Nome_Fornitore, Cod_Email)

Nome_Fornitore → FORNITORE.Nome
Cod_Email → EMAIL.Cod_Email

GRUPPO (Nome, Descrizione, Categoria)

AFFERENZA (Cod_Contatto, Nome_Gruppo)

Cod_Contatto → CONTATTO.Cod_Contatto
Nome_Gruppo → GRUPPO.Nome

TELEFONO_FISSO (CodN_Fisso, Numero, Prefisso_Naz, Prefisso_C, Cod_Contatto)

Cod_Contatto → CONTATTO.Cod_Contatto

TELEFONO_MOBILE (CodN_Mobile, Numero, Prefisso_Naz, Cod_Contatto)

Cod_Contatto → CONTATTO.Cod_Contatto

REINDIRIZZAMENTO (CodN_Fisso, CodN_Mobile)

CodN_Fisso → TELEFONO_FISSO.CodN_Fisso
CodN_Mobile → TELEFONO_MOBILE.CodN_Mobile

Capitolo 4

Progettazione fisica

In questo capitolo vengono documentate le definizioni SQL delle tabelle indicate dallo schema logico, inoltre verranno anche fornite le implementazioni dei vincoli, delle procedure e dei trigger.
Si fa presente che le seguenti definizioni fanno capo alla sintassi di PostgreSQL.

Le operazioni che chiamano in causa il nome dello schema sono effettuate presupponendo che il nome dello schema sia 'public'.

4.1 Definizione Tabelle

4.1.1 Definizione Tabella CONTATTO

```
CREATE TABLE CONTATTO(
 Cod_Contatto  SERIAL        NOT NULL,
 Nome          VARCHAR(100)  NOT NULL,
 Cognome       VARCHAR(100)  NOT NULL,
 Foto          VARCHAR(1000) NOT NULL );

--vincolo chiave primaria tabella CONTATTO
ALTER TABLE CONTATTO
 ADD CONSTRAINT PK_CONTATTO PRIMARY KEY(Cod_Contatto);

--SEQUENZA SERIAL Cod_Contatto
ALTER SEQUENCE public.contatto_cod_contatto_seq
 RESTART WITH 1;
```

4.1.2 Definizione Tabella INDIRIZZO

```
CREATE TABLE INDIRIZZO(
 Cod_Indirizzo SERIAL       NOT NULL,
 Via           VARCHAR(100) NOT NULL,
 Cap           INTEGER      NOT NULL,
 Nazione       VARCHAR(100) NOT NULL,
 Citta         VARCHAR(100) NOT NULL );

--vincolo chiave primaria tabella INDIRIZZO
ALTER TABLE INDIRIZZO
 ADD CONSTRAINT PK_INDIRIZZO
  PRIMARY KEY(Cod_Indirizzo);

--SEQUENZA SERIAL Cod_Indirizzo
ALTER SEQUENCE public.indirizzo_cod_indirizzo_seq
 RESTART WITH 1;
```

4.1.3 Definizione Tabella POSSIEDE

```
CREATE TABLE POSSIEDE(
 Cod_Indirizzo  SERIAL NOT NULL,
 Cod_Contatto   SERIAL NOT NULL );

--vincoli chiave esterna tabella POSSIEDE
ALTER TABLE POSSIEDE
 ADD CONSTRAINT FK_POSSIEDE_INDIRIZZO
  FOREIGN KEY(Cod_Indirizzo)
  REFERENCES INDIRIZZO(Cod_Indirizzo) ON DELETE CASCADE;

ALTER TABLE POSSIEDE
 ADD CONSTRAINT FK_POSSIEDE_CONTATTO
  FOREIGN KEY(Cod_Contatto)
  REFERENCES CONTATTO(Cod_Contatto) ON DELETE CASCADE;
```

4.1.4 Definizione Tabella GRUPPO

```
CREATE TABLE GRUPPO(
 Nome           VARCHAR(100)  NOT NULL,
 Descrizione    VARCHAR(1000) NOT NULL,
 Categoria      VARCHAR(100)  NOT NULL );

--vincolo chiave primaria tabella GRUPPO
ALTER TABLE GRUPPO
 ADD CONSTRAINT PK_GRUPPO
  PRIMARY KEY(Nome);
```

4.1.5 Definizione Tabella AFFERENZA

```
CREATE TABLE AFFERENZA(
 Cod_Contatto     SERIAL          NOT NULL,
 Nome_Gruppo      VARCHAR(100)    NOT NULL );

--vincoli chiave esterna tabella AFFERENZA
ALTER TABLE AFFERENZA
 ADD CONSTRAINT FK_AFFERENZA_CONTATTO
  FOREIGN KEY(Cod_Contatto)
  REFERENCES CONTATTO(Cod_Contatto) ON DELETE CASCADE;

ALTER TABLE AFFERENZA
 ADD CONSTRAINT FK_AFFERENZA_GRUPPO
  FOREIGN KEY(Nome_Gruppo)
  REFERENCES GRUPPO(Nome) ON DELETE CASCADE;
```

4.1.6 Definizione Tabella EMAIL

```
CREATE TABLE EMAIL(
 Cod_Email        SERIAL          NOT NULL,
 Indirizzo        VARCHAR(100)    NOT NULL,
 Dominio          VARCHAR(100)    NOT NULL,
 Cod_Contatto     SERIAL          NOT NULL );

--vincolo chiave primaria tabella EMAIL
ALTER TABLE EMAIL
 ADD CONSTRAINT PK_EMAIL
  PRIMARY KEY(Cod_Email);

--vincolo chiave esterna tabella EMAIL
ALTER TABLE EMAIL
 ADD CONSTRAINT FK_EMAIL_CONTATTO
  FOREIGN KEY(Cod_Contatto)
  REFERENCES CONTATTO(Cod_Contatto) ON DELETE CASCADE;

--vincolo unique tabella EMAIL
ALTER TABLE EMAIL
 ADD CONSTRAINT U_EMAIL
  UNIQUE(Cod_Email,Cod_Contatto);

--vincolo dominio corretto email
ALTER TABLE EMAIL
 ADD CONSTRAINT DOMINIO_CORRETTO
  CHECK (Dominio LIKE '@_%._%');

--vincolo inidirizzo corretto email
ALTER TABLE EMAIL
 ADD CONSTRAINT INDIRIZZO_CORRETTO
  CHECK (Indirizzo LIKE '_____%');

--SEQUENZA SERIAL Cod_Email
ALTER SEQUENCE public.email_cod_email_seq
 RESTART WITH 1;
```

4.1.7 Definizione Tabella FORNITORE

```
CREATE TABLE FORNITORE(
 Nome              VARCHAR(100)   NOT NULL,
 Categoria         VARCHAR(100)   NOT NULL,
 Casa_Produttrice  VARCHAR(100)   NOT NULL );

--vincolo chiave primaria tabella FORNITORE
ALTER TABLE FORNITORE
 ADD CONSTRAINT PK_FORNITORE
  PRIMARY KEY(Nome);
```

4.1.8 Definizione Tabella ACCOUNT

```
CREATE TABLE ACCOUNT(
 Nickname          VARCHAR(100)   NOT NULL,
 Frase_benvenuto   VARCHAR(100),
 Nome              VARCHAR(100)   NOT NULL,
 Cognome           VARCHAR(100)   NOT NULL,
 Nome_Fornitore    VARCHAR(100)   NOT NULL,
 Cod_Email         SERIAL         NOT NULL );

--vincoli chiave primaria tabella ACCOUNT
ALTER TABLE ACCOUNT
 ADD CONSTRAINT PK_ACCOUNT
  PRIMARY KEY(Nickname);

--vincoli chiave esterna tabella ACCOUNT
ALTER TABLE ACCOUNT
 ADD CONSTRAINT FK_ACCOUNT_NOME_F
  FOREIGN KEY(Nome_Fornitore)
  REFERENCES FORNITORE(Nome) ON DELETE CASCADE;

ALTER TABLE ACCOUNT
 ADD CONSTRAINT FK_ACCOUNT_EMAIL
  FOREIGN KEY(Cod_Email)
  REFERENCES EMAIL(Cod_Email) ON DELETE CASCADE;

--vincolo unique tabella ACCOUNT
ALTER TABLE ACCOUNT
 ADD CONSTRAINT U_ACCOUNT
  UNIQUE(Nome_Fornitore, Cod_Email);
```

4.1.9 Definizione Tabella TELEFONO_MOBILE

```
CREATE TABLE TELEFONO_MOBILE(
 CodN_Mobile        SERIAL             NOT NULL,
 Prefisso_Naz       VARCHAR(5)         NOT NULL,
 Numero             DOUBLE PRECISION   NOT NULL,
 Cod_Contatto       SERIAL             NOT NULL );

--vincolo chiave primaria tabella TELEFONO_MOBILE
ALTER TABLE TELEFONO_MOBILE
 ADD CONSTRAINT PK_TELEFONOM
  PRIMARY KEY(CodN_Mobile);

--vincolo chiave esterna tabella TELEFONO_MOBILE
ALTER TABLE TELEFONO_MOBILE
 ADD CONSTRAINT FK_TELEFONOM
  FOREIGN KEY(Cod_Contatto)
  REFERENCES CONTATTO(Cod_Contatto) ON DELETE CASCADE;

--vincolo prefisso nazionale corretto telefono_mobile
ALTER TABLE TELEFONO_MOBILE
 ADD CONSTRAINT PREFISSO_NAZ_CORRETTO_TM
  CHECK (Prefisso_Naz LIKE '+%');

--SEQUENZA SERIAL CodN_mobile
ALTER SEQUENCE public.telefono_mobile_codn_mobile_seq
 RESTART WITH 1;
```

4.1.10 Definizione Tabella TELEFONO_FISSO

```
CREATE TABLE TELEFONO_FISSO(
 CodN_Fisso         SERIAL             NOT NULL,
 Prefisso_Naz       VARCHAR(5)         NOT NULL,
 Prefisso_C         INTEGER            NOT NULL,
 Numero             DOUBLE PRECISION   NOT NULL,
 Cod_Contatto       SERIAL             NOT NULL );

--vincolo chiave primaria tabella TELEFONO_FISSO
ALTER TABLE TELEFONO_FISSO
 ADD CONSTRAINT PK_TELEFONOF
  PRIMARY KEY(CodN_Fisso);

--vincolo chiave esterna TELEFONO_FISSO
ALTER TABLE TELEFONO_FISSO
 ADD CONSTRAINT FK_TELEFONOF
  FOREIGN KEY(Cod_Contatto)
  REFERENCES CONTATTO(Cod_Contatto) ON DELETE CASCADE;

--vincolo prefisso nazionale corretto telefono_fisso
ALTER TABLE TELEFONO_FISSO
 ADD CONSTRAINT PREFISSO_NAZ_CORRETTO_TF
  CHECK (Prefisso_Naz LIKE '+%');

--SEQUENZA SERIAL CodN_fisso
ALTER SEQUENCE public.telefono_fisso_codn_fisso_seq
 RESTART WITH 1;
```

4.1.11 Definizione Tabella REINDIRIZZAMENTO

```
CREATE TABLE REINDIRIZZAMENTO(
 CodN_Fisso       SERIAL       NOT NULL,
 CodN_Mobile      SERIAL       NOT NULL );

--vincoli chiave esterna tabella REINDIRIZZAMENTO
ALTER TABLE REINDIRIZZAMENTO
 ADD CONSTRAINT FK_REINDIRIZZAMENTO_TF
  FOREIGN KEY(CodN_Fisso)
  REFERENCES TELEFONO_FISSO(CodN_Fisso) ON DELETE CASCADE;

ALTER TABLE REINDIRIZZAMENTO
 ADD CONSTRAINT FK_REINDIRIZZAMENTO_TM
  FOREIGN KEY(CodN_Mobile)
  REFERENCES TELEFONO_MOBILE(CodN_Mobile) ON DELETE CASCADE;
```

4.2 Definizione Trigger

4.2.1 Trigger INDIRIZZO_PRINCIPALE_OBBLIGATORIO

```
create or replace TRIGGER INDIRIZZO_PRINCIPALE_OBBLIGATORIO
BEFORE DELETE ON INDIRIZZO
FOR EACH ROW
EXECUTE PROCEDURE trigger_function_eliminaindirizzo();

--DEFINIZIONE trigger_function_eliminaindirizzo()
create or replace function trigger_function_EliminaIndirizzo()
    returns trigger
language PLPGSQL
AS $$
BEGIN
    IF NOT EXISTS (SELECT * FROM POSSIEDE WHERE Cod_Indirizzo <> OLD.Cod_Indirizzo
                AND Cod_Contatto = OLD.Cod_Contatto) THEN
        RAISE EXCEPTION E'L\'eliminazione dell\'indirizzo viola il vincolo relazionale
            tra CONTATTO e INDIRIZZO che prevede almeno un indirizzo per ogni contatto'
        USING HINT = 'Assegna prima un indirizzo alternativo allo stesso contatto';
    END IF;

    RETURN OLD;
END;
$$
```

4.2.2 Trigger TELEFONO_MOBILE_OBBLIGATORIO

```
create or replace TRIGGER TELEFONO_MOBILE_OBBLIGATORIO
BEFORE DELETE ON TELEFONO_MOBILE
FOR EACH ROW
EXECUTE PROCEDURE trigger_function_EliminaMobile();

--DEFINIZIONE trigger_function_EliminaMobile()
create or replace function trigger_function_EliminaMobile()
    returns trigger
language PLPGSQL
AS $$
BEGIN
    IF NOT EXISTS (SELECT * FROM TELEFONO_MOBILE WHERE CodN_Mobile <> OLD.CodN_Mobile
                AND Cod_Contatto = OLD.Cod_Contatto) THEN
        RAISE EXCEPTION E'L\'eliminazione del telefono mobile viola il vincolo relazionale
            tra CONTATTO e TELEFONO_MOBILE che prevede almeno un telefono mobile per ogni contatto'
        USING HINT = 'Assegna prima un telefono mobile alternativo allo stesso contatto';
    END IF;

    RETURN OLD;
END;
$$
```

4.2.3 Trigger TELEFONO_FISSO_OBBLIGATORIO

```
create or replace TRIGGER INDIRIZZO_PRINCIPALE_OBBLIGATORIO
BEFORE DELETE ON INDIRIZZO
FOR EACH ROW
EXECUTE PROCEDURE trigger_function_eliminaindirizzo();

--DEFINIZIONE trigger_function_eliminaindirizzo()
create or replace function trigger_function_EliminaIndirizzo()
    returns trigger
language PLPGSQL
AS $$
BEGIN
    IF NOT EXISTS (SELECT * FROM POSSIEDE WHERE Cod_Indirizzo <> OLD.Cod_Indirizzo
                AND Cod_Contatto = OLD.Cod_Contatto) THEN
        RAISE EXCEPTION E'L\'eliminazione dell\'indirizzo viola il vincolo relazionale
            tra CONTATTO e INDIRIZZO che prevede almeno un indirizzo per ogni contatto'
        USING HINT = 'Assegna prima un indirizzo alternativo allo stesso contatto';
    END IF;

    RETURN OLD;
END;
$$
```

4.3 Definizione PROCEDURE

4.3.1 Procedura CREA CONTATTO

```
--procedura creazione CONTATTO
create or replace procedure creaContatto
( Nome_c VARCHAR(100), Cognome_c VARCHAR(100), Foto_c VARCHAR(1000),
Ind_email VARCHAR(100), Dom_email VARCHAR(100),
Via_c VARCHAR(100), CAP_c INTEGER, Nazione_c VARCHAR(100), Citta_c VARCHAR(100),
Pre_naz_fisso VARCHAR(5),Pre_C_fisso INTEGER, Num_fisso DOUBLE PRECISION,
Pre_naz_mobile VARCHAR(5), Numero_mobile DOUBLE PRECISION )
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
DECLARE c_contatto INTEGER;
DECLARE c_indirizzo INTEGER;
begin
    api := E'\'';
    sep := E'\',\'';
    comando := E'INSERT INTO CONTATTO(Nome, Cognome, Foto) VALUES(' || api || Nome_c || sep
            || Cognome_c || sep || Foto_c || api || ')';
        EXECUTE comando;
    SELECT MAX(Cod_Contatto) INTO c_contatto FROM CONTATTO
        WHERE Nome = Nome_c AND Cognome = Cognome_c AND Foto = Foto_c;
    comando := E'INSERT INTO EMAIL(Indirizzo, Dominio, Cod_Contatto) VALUES(' || api || Ind_email || sep
            || Dom_email || sep || c_contatto || api || ')';
        EXECUTE comando;
    comando := E'INSERT INTO TELEFONO_FISSO(Prefisso_Naz, Prefisso_C, Numero, Cod_Contatto) VALUES(' || api
            || Pre_naz_fisso || sep || Pre_C_fisso || sep || Num_fisso || sep || c_contatto || api || ')';
        EXECUTE comando;
    comando := E'INSERT INTO TELEFONO_MOBILE(Prefisso_Naz, Numero, Cod_Contatto) VALUES(' || api || Pre_naz_mobile
            || sep || Numero_mobile || sep || c_contatto || api || ')';
        EXECUTE comando;
    comando := E'INSERT INTO INDIRIZZO(Via, Cap, Nazione, Citta) VALUES(' || api || Via_c || sep || CAP_c || sep
            || Nazione_c || sep || Citta_c || api || ')';
        EXECUTE comando;
    SELECT MAX(Cod_Indirizzo) INTO c_indirizzo FROM INDIRIZZO
        WHERE Via = Via_c AND CAP = CAP_c AND Nazione = Nazione_c AND Citta = Citta_C;
    comando := E'INSERT INTO POSSIEDE(Cod_Indirizzo, Cod_Contatto) VALUES(' || api || c_indirizzo || sep
            || c_contatto || api || ')';
        EXECUTE comando;
end;
$$
```

4.3.2 Procedura INSERISCI EMAIL

```
--procedura inserimento dati tabella EMAIL
create or replace procedure aggiungiEmail
(Ind_email VARCHAR(100), Dom_email VARCHAR(100), c_contatto INTEGER)
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
begin
    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO EMAIL(Indirizzo, Dominio, Cod_Contatto) VALUES(' || api
            || Ind_email || sep || Dom_email || sep || c_contatto || api || ')';
    EXECUTE comando;

end;
$$
```

4.3.3 Procedura INSERISCI ACCOUNT

```
--procedura inserimento dati tabella ACCOUNT
create or replace procedure aggiungiAccount (
Nickname VARCHAR(100), Frase VARCHAR(100), Nome VARCHAR(100),
Cognome VARCHAR(100), Nome_Fornitore VARCHAR(100), c_email INTEGER)
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
begin

    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO ACCOUNT(Nickname, Frase_Benvenuto, Nome, Cognome, Nome_Fornitore, Cod_Email)
            VALUES(' || api || Nickname || sep || Frase || sep || Nome || sep || Cognome || sep
            ||Nome_Fornitore || sep || c_email ||  api || ')';
    EXECUTE comando;

end;
$$
```

4.3.4 Procedura INSERISCI CONTATTO A GRUPPO

```
--procedura inserimento dati contatto in un gruppo
create or replace procedure aggiungiGruppoContatto
(c_contatto INTEGER, Nome_g VARCHAR(100))
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
begin

    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO AFFERENZA(Cod_Contatto, Nome_Gruppo)
    VALUES(' || api || c_contatto || sep || Nome_g || api || ')';
    EXECUTE comando;

end;
$$
```

4.3.5 Procedura INSERISCI INDIRIZZO

```
--procedura inserimento dati tabella INDIRIZZO
create or replace procedure aggiungiIndirizzo (Via_c VARCHAR(100),
CAP_c INTEGER, Nazione_c VARCHAR(100), Citta_c VARCHAR(100), c_Contatto INTEGER)
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
DECLARE c_indirizzo INTEGER;
begin

    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO INDIRIZZO(Via, CAP, Nazione, Citta) VALUES(' || api
            || Via_c || sep || CAP_c || sep || Nazione_c || sep || Citta_c || api || ')';
    EXECUTE comando;

    SELECT MAX(Cod_Indirizzo) INTO c_indirizzo FROM INDIRIZZO
    WHERE Via = Via_c AND CAP = CAP_c AND Nazione = Nazione_c AND Citta = Citta_C;

    comando := E'INSERT INTO POSSIEDE(Cod_Indirizzo, Cod_Contatto) VALUES(' || api
            || c_indirizzo || sep || c_contatto || api || ')';
    EXECUTE comando;

end;
$$
```

4.3.6 Procedura INSERISCI TELEFONO FISSO

```
--procedura inserimento dati tabella TELEFONO_FISSO
create or replace procedure aggiungiFisso
(Pre_naz_fisso VARCHAR(5),Pre_C_fisso INTEGER,
 Num_fisso DOUBLE PRECISION, c_contatto INTEGER)
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
begin

    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO TELEFONO_FISSO(Prefisso_Naz, Prefisso_C, Numero, Cod_Contatto)
            VALUES(' || api || Pre_naz_fisso || sep || Pre_C_fisso || sep || Num_fisso
            || sep || c_contatto || api || ')';
    EXECUTE comando;

end;
$$
```

4.3.7 Procedura INSERISCI TELEFONO MOBILE

```
--procedura inserimento dati tabella TELEFONO_MOBILE
create or replace procedure aggiungiMobile
(Pre_naz_mobile VARCHAR(5), Numero_mobile DOUBLE PRECISION, c_contatto INTEGER)
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
begin

    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO TELEFONO_MOBILE(Prefisso_Naz, Numero, Cod_Contatto)
            VALUES(' || api || Pre_naz_mobile || sep || Numero_mobile || sep
            || c_contatto || api || ')';
    EXECUTE comando;

end;
$$
```

4.3.8 Procedura INSERISCI REINDIRIZZAMENTO

```
--procedura inserimento dati tabella reindirizzamento
create or replace procedure aggiungiReindirizzamento
(c_fisso INTEGER, c_mobile INTEGER)
language PLPGSQL
as $$
DECLARE comando text;
DECLARE api text;
DECLARE sep text;
begin

    api := E'\'';
    sep := E'\',\'';

    comando := E'INSERT INTO REINDIRIZZAMENTO(CodN_Fisso, CodN_Mobile)
            VALUES(' || api || c_fisso || sep || c_mobile || api || ')';
    EXECUTE comando;

end;
$$
```

4.3.9 Procedura MODIFICA CONTATTO

```
--procedura modifica contatto
create or replace procedure modificaContatto
(c_contatto INTEGER, nome varchar(100), cognome varchar(100))
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update CONTATTO set Nome = \'' || nome
                    | E'\', Cognome = \'' || cognome
                    || E'\' where Cod_Contatto =\'' || c_contatto || E'\'';
    EXECUTE comando_modifica;
end;
$$
```

4.3.10 Procedura MODIFICA FOTO CONTATTO

```
--procedura modifica foto_contatto
create or replace procedure modificaFotoContatto
(c_contatto integer, foto_path varchar(100))
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update CONTATTO set Foto =\''
                || foto_path || E'\' where Cod_Contatto =\''
                || c_contatto || E'\'';

    EXECUTE comando_modifica;
end;
$$
```

4.3.11 Procedura MODIFICA EMAIL

```
--procedura modifica email
create or replace procedure modificaEmail
(c_email integer, Indirizzo varchar(100), Dominio varchar(100))
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update Email set Indirizzo = \'' || indirizzo
                || E'\', Dominio = \'' || dominio
                || E'\' where Cod_Email = \'' ||c_email || E'\'';

    EXECUTE comando_modifica;
end;
$$
```

4.3.12 Procedura MODIFICA ACCOUNT

```
--procedura modifica account
create or replace procedure modificaAccount
(Nickname varchar(100), frase_ben VARCHAR(300))
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update ACCOUNT set Frase_Benvenuto =\''
                    || frase_ben || E'\' where Nickname = \''
                    || Nickname || E'\'';
    EXECUTE comando_modifica;
end;
$$
```

4.3.13 Procedura MODIFICA INDIRIZZO

```
--procedura modifica indirizzo
create or replace procedure modificaIndirizzo
(c_indirizzo integer, via varchar(100), cap integer, nazione varchar(100), citta varchar(100))
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update INDIRIZZO set Via = \'' || via || E'\', CAP = \'' || cap
                        || E'\', Nazione = \'' || nazione || E'\', Citta = \'' || citta
                        || E'\' where Cod_Indirizzo =\'' || c_indirizzo || E'\'';

    EXECUTE comando_modifica;
end;
$$
```

4.3.14 Procedura MODIFICA TELEFONO MOBILE

```
--procedura modifica telefono_mobile
create or replace procedure modificaTelefonoMobile
(c_mobile integer, prefissoN varchar(5), numero double precision)
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update TELEFONO_MOBILE set Prefisso_Naz = \''
                    || prefissoN || E'\', Numero = \'' || numero
                    || E'\' where CodN_Mobile = \'' || c_mobile || E'\'';

    EXECUTE comando_modifica;
end;
$$
```

4.3.15 Procedura MODIFICA TELEFONO FISSO

```
--procedura modifica telefono_fisso
create or replace procedure modificaTelefonoFisso
(c_fisso integer, prefissoN varchar(5),
 prefissoC integer, numero double precision)
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'update TELEFONO_FISSO set Prefisso_Naz = \''
                    || prefissoN || E'\', Prefisso_C = \'' || prefissoC
                    || E'\', Numero =\'' || numero || E'\' where CodN_Fisso = \''
                    || c_fisso || E'\'';

    EXECUTE comando_modifica;
end;
$$
```

4.3.16 Procedura ELIMINA CONTATTO

```
--procedura elimina contatto
create or replace procedure eliminaContatto (c_contatto INTEGER)
language PLPGSQL
as $$
DECLARE comando_eliminazione text;
DECLARE comando text;
DECLARE Codici_ind_cursor refcursor;
DECLARE curr_ind INTEGER;
begin

    ALTER TABLE INDIRIZZO DISABLE TRIGGER INDIRIZZO_PRINCIPALE_OBBLIGATORIO;
    ALTER TABLE TELEFONO_FISSO DISABLE TRIGGER TELEFONO_FISSO_OBBLIGATORIO;
    ALTER TABLE TELEFONO_MOBILE DISABLE TRIGGER TELEFONO_MOBILE_OBBLIGATORIO;

    comando:= E'SELECT Cod_Indirizzo FROM POSSIEDE WHERE Cod_Contatto = \'' || c_contatto || E'\'';
    open Codici_ind_cursor FOR EXECUTE comando;
    LOOP
        FETCH Codici_ind_cursor INTO curr_ind;
        EXIT WHEN NOT FOUND;
        comando_eliminazione := E'delete from INDIRIZZO where Cod_Indirizzo = \'' || curr_ind || E'\'';
        EXECUTE comando_eliminazione;
    END LOOP;

    comando_eliminazione := E'delete from CONTATTO where Cod_Contatto = \'' || c_contatto || E'\'';
    EXECUTE comando_eliminazione;

    ALTER TABLE INDIRIZZO ENABLE TRIGGER INDIRIZZO_PRINCIPALE_OBBLIGATORIO;
    ALTER TABLE TELEFONO_FISSO ENABLE TRIGGER TELEFONO_FISSO_OBBLIGATORIO;
    ALTER TABLE TELEFONO_MOBILE ENABLE TRIGGER TELEFONO_MOBILE_OBBLIGATORIO;
end;
$$
```

4.3.17 Procedura ELIMINA EMAIL

```
--procedura elimina email
create or replace procedure eliminaEmail (c_email INTEGER)
language PLPGSQL
as $$
DECLARE comando_eliminazione text;
begin

    comando_eliminazione := E'delete from email where Cod_Email = \''
                        || c_email || E'\'';

    EXECUTE comando_eliminazione;
end;
$$
```

4.3.18 Procedura ELIMINA ACCOUNT

```
--procedura elimina account
create or replace procedure eliminaAccount (Nickname varchar(100))
language PLPGSQL
as $$
DECLARE comando_eliminazione text;
begin

    comando_eliminazione := E'delete from ACCOUNT where Nickname= \''
                        || Nickname  || E'\'';

    EXECUTE comando_eliminazione;
end;
$$
```

4.3.19 Procedura ELIMINA INDIRIZZO

```
--procedura elimina indirizzo
create or replace procedure eliminaIndirizzo (c_indirizzo INTEGER)
language PLPGSQL
as $$
DECLARE comando_eliminazione text;
begin

    comando_eliminazione := E'delete from INDIRIZZO where Cod_Indirizzo = \''
                        || c_indirizzo || E'\'';

    EXECUTE comando_eliminazione;
end;
$$
```

4.3.20 Procedura ELIMINA AFFERENZA

```
--procedura elimina afferenza
create or replace procedure eliminaAfferenza
(c_contatto INTEGER, nome_gruppo VARCHAR(100))
language PLPGSQL
as $$
DECLARE comando_modifica text;
begin

    comando_modifica := E'delete from AFFERENZA where Cod_Contatto = \''
                    || c_contatto  || E'\' and Nome_Gruppo =\''
                    || nome_gruppo || E'\'';

    EXECUTE comando_modifica;
end;
$$
```

4.3.21 Procedura ELIMINA TELEFONO MOBILE

```
--procedura elimina telefono_mobile
create or replace procedure eliminaTelefonoMobile(c_mobile INTEGER)
language PLPGSQL
as $$
DECLARE comando_eliminazione text;
begin

    comando_eliminazione := E'delete from TELEFONO_MOBILE where CodN_Mobile=\''
                        || c_mobile || E'\'';

    EXECUTE comando_eliminazione;
end;
$$
```

4.3.22 Procedura ELIMINA TELEFONO FISSO

```
--procedura elimina telefono_fisso
create or replace procedure eliminaTelefonoFisso(c_fisso INTEGER)
language PLPGSQL
as $$
DECLARE comando_eliminazione text;
begin

    comando_eliminazione := E'delete from TELEFONO_FISSO where CodN_Fisso=\''
                        || c_fisso || E'\'';

    EXECUTE comando_eliminazione;
end;
$$
```

4.3.23 Procedura RICERCA PER NOME

```
--procedura ricerca contatto per NOME
create or replace procedure ricercaNome (nomePersona varchar(100))
language PLPGSQL
as $$
DECLARE comando_ricerca text;
begin

    comando_ricerca := E'create or replace view vista_ricerca as select *
                    from CONTATTO
                    where nome =\'' || nomePersona || E'\'';

    EXECUTE comando_ricerca;
end;
$$
```

4.3.24 Procedura RICERCA PER EMAIL

```
--procedura ricerca contatto per EMAIL
create or replace procedure ricercaEmail (indirizzo varchar(100), dominio varchar(100))
language PLPGSQL
as $$
DECLARE comando_ricerca text;
begin

    comando_ricerca := E'create or replace view vista_ricerca_email as
            select c.nome, c.cognome, c.foto
                    from email as e join contatto as c on e.cod_contatto=c.cod_contatto
                    where e.Indirizzo = \'' || indirizzo|| E'\'
                        and e.Dominio = \'' || dominio|| E'\'';

    EXECUTE comando_ricerca;
end;
$$
```

4.3.25 Procedura RICERCA PER ACCOUNT

```
--procedura ricerca contatto per ACCOUNT
create or replace procedure RicercaPerAccount (Nickname varchar(100))
language PLPGSQL
as $$
DECLARE comando_ricerca text;
begin

    comando_ricerca := E'create or replace view vista_ricerca_per_account as
                    select c.nome,c.cognome,c.foto
                    from ACCOUNT as a join EMAIL as e on a.cod_email=e.cod_email
                    join contatto as c on e.cod_contatto=c.cod_contatto
                    where a.Nickname= \'' || Nickname  || E'\'';

    EXECUTE comando_ricerca;
end;
$$
```

4.3.26 Procedura RICERCA PER TELEFONO MOBILE

```
--procedura ricerca contatto per telefono_mobile
create or replace procedure ricercaPerTelefonoMobile(prefissoN varchar(5), numero double precision)
language PLPGSQL
as $$
DECLARE comando_ricerca text;
begin

    comando_ricerca := E'create or replace view vista_ricerca_per_mobile as
                    select c.nome,c.cognome,c.foto
                    from TELEFONO_MOBILE as t join CONTATTO as c on t.Cod_contatto=c.Cod_contatto
                    where t.Prefisso_Naz=\'' || prefissoN|| E'\'and t.Numero=\'' || numero|| E'\'';

    EXECUTE comando_ricerca;
end;
$$
```

4.3.27 Procedura RICERCA PER TELEFONO FISSO

```
--procedura ricerca contatto per telefono_fisso
create or replace procedure ricercaPerTelefonoFisso
(prefissoN varchar(5),prefissoC integer, numero double precision)
language PLPGSQL
as $$
DECLARE comando_ricerca text;
begin

    comando_ricerca := E'create or replace view vista_ricerca_per_fisso as
                    select c.nome,c.cognome,c.foto
                    from TELEFONO_FISSO as t join CONTATTO as c on t.Cod_contatto=c.Cod_contatto
                    where t.Prefisso_Naz=\'' || prefissoN
                    || E'\'and t.Prefisso_C=\'' || prefissoC
                    || E'\'and t.Numero=\'' || numero|| E'\'';

    EXECUTE comando_ricerca;
end;
$$
```