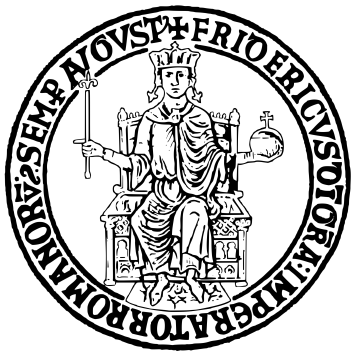


Progettazione e sviluppo di una base di dati relazionale per il tracciamento del SARS-CoV-2

Autore: Alessandro Palumbo (Matricola N86001990)
Docenti: Adriano Peron - Silvio Barra

March 22 2021

Indice

Capitolo 1

Descrizione del Progetto

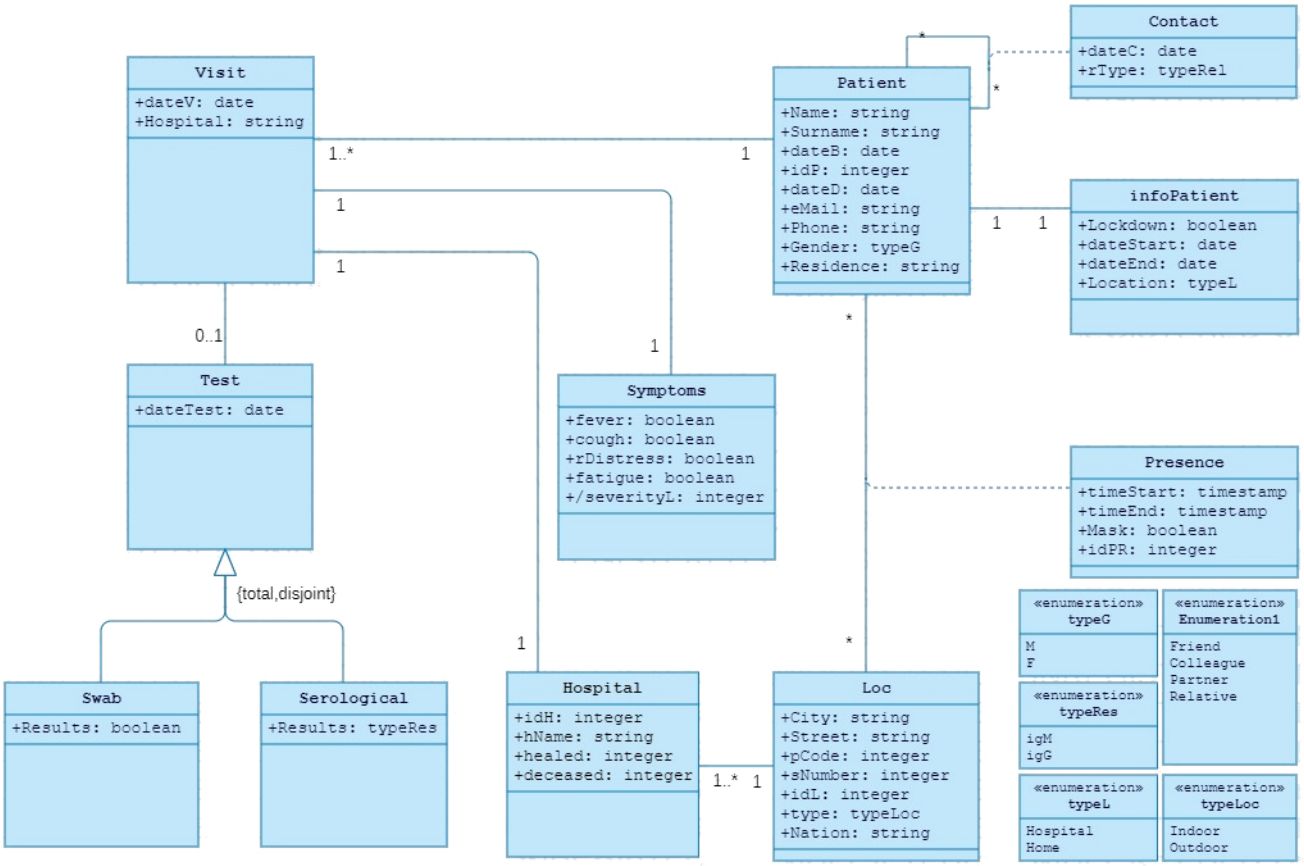
Il progetto qui descritto consiste nell'implementazione di una base di dati relazionale, con il fine di recuperare dati ed elaborarli per costituire cosí un sistema di **contact tracing** per il monitoraggio del virus **SARS-CoV-2**. Si suppone che i dati vengano memorizzati in un database distribuito gesitito da strutture ospedaliere e simili. Si possono dunque elaborare dati quali le occasioni di contatto tra persone (che verranno qui descritte unicamente come pazienti) e i luoghi nei quali tali contatti sono avvenuti, le possibili relazioni e legami interpersonali, i test svolti (che verranno suddivisi unicamente in test sierologico e tampone naso faringeo) e i conseguenti risultati, infine verranno memorizzate le informazioni relative ad ogni singolo paziente sullo stato della malattia e sull'eventuale e conseguente stato di isolamento forzato.

Capitolo 2

Progettazione Concettuale

La fase di progettazione consiste in una descrizione dei requisiti e comprende descrizioni dettagliate dei tipi di entità, delle associazioni e dei vincoli. Tramite questo approccio, il progettista può concentrarsi sulle specifiche delle proprietà dei dati, senza doversi preoccupare dei dettagli implementativi. Di seguito viene usata la notazione **UML** (Unified Modeling Language) dei diagrammi delle classi.

2.1 Class Diagram



2.2 Ristrutturazione Class Diagram

Ai fini di rendere l'implementazione piú efficiente e facilitare la traduzione da class diagram a schemi relazionali, si procede con la ristruttuazione di quest'ultimo.

2.2.1 Analisi Delle Ridondanze

Nessuna ridondanza degna di nota.

2.2.2 Analisi Degli Identificativi

Vengono aggiunte alle classi delle "chiavi tecniche" che fungeranno da identificativi numerici ai fini di una piú facile discriminazione delle istanze.

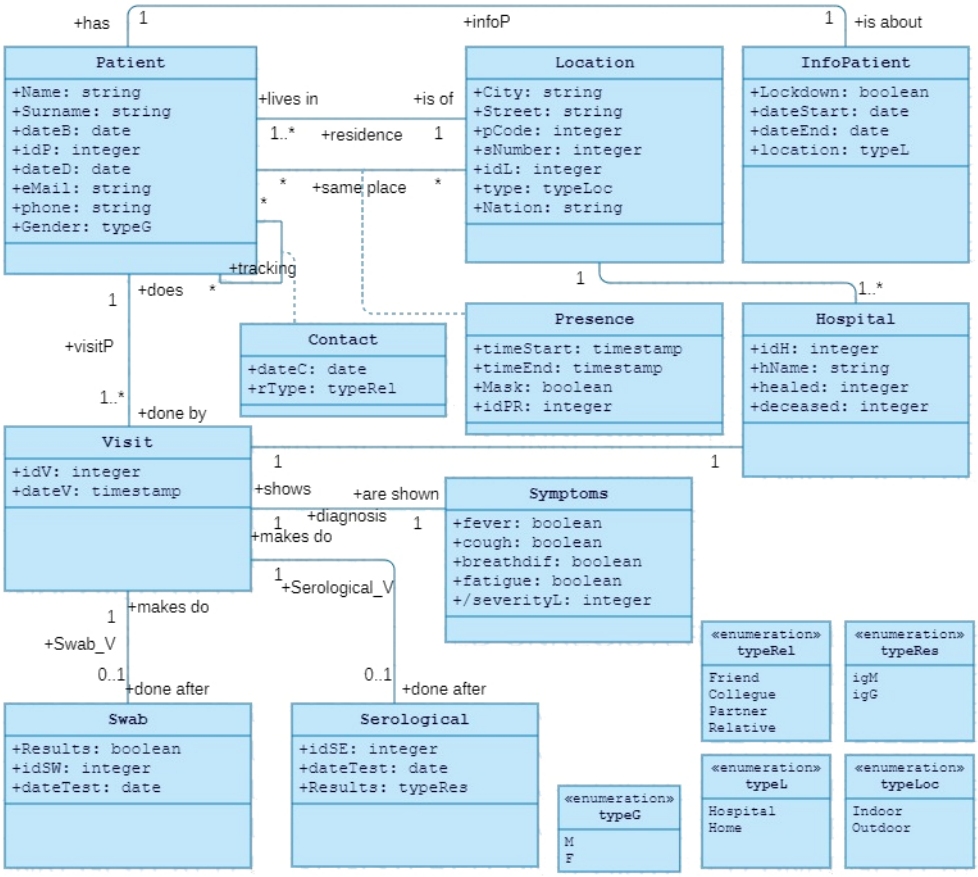
2.2.3 Rimozione Degli Attributi Strutturati/Multipli

All'interno della classe *Patient* vi é un attributo **residence** di tipo *string* che indica la residenza di un paziente. Avendo giá a disposizione un set di attributi per identificare un luogo nella classe *Loc* si preferisce quindi eliminare l'attributo e aggiungere bensí una associazione tra le due classi.

2.2.4 Analisi Delle Gerarchie Di Specializzazione

La specializzazione che vede coinvolte la superclasse *Test* e le sottoclassi *Swab* e *Serological* é di tipo **total**,**disjoint**. L'eliminazione della specializzazione consisterá nello "schiacciare" la superclasse nelle sottoclassi.

2.2.5 Class Diagram Ristrutturato



2.3 Dizionario Dei Dati

Il dizionario dei dati è un sistema per immagazzinare informazioni di catalogo sugli schemi ed i vincoli, oltre ad informazioni aggiuntive quali decisioni progettuali e standard d'uso. Verrano qui presentati tre tipi di dizionario: delle classi, dei vincoli e delle associazioni.

2.3.1 Dizionario Delle Classi

Nome	Descrizione	Attributi
Loc	Descrittore di ciascun luogo presente nel database.	**idL**(*integer*): Identifica univocamente ogni istanza di Loc. **City**(*string*): Nome della città in questione. **p_Code**(*integer*): Codice Postale (CAP). **Street**(*string*): Riferimento urbano (via/piazza/viale..) della città. **s_Number**(*integer*): Numero civico della via. **Nation**(*string*): Nazione di riferimento. In questo caso si assume di stare esclusivamente in 'Italia'. **l_Type**(*string*): Tipo di luogo (all'aperto/al chiuso).
Patient	Descrittore di ciascun paziente presente nel database.	**Name**(*string*): Nome proprio del paziente. **Surname**(*string*): Cognome del paziente **dateB**(*date*): Data di nascita del paziente **dateD**(*date*): Data di morte del paziente (settata a NULL di default). **idP**(*integer*): Identifica univocamente ogni istanza di Patient **eMail**(*string*): Indirizzo e-Mail fornito dal paziente. **Phone**(*string*): Numero di cellulare fornito dal paziente. **Gender**(*string*): Sesso del paziente (Maschile/Femminile).

Nome	Descrizione	Attributi
Symptoms	Descrittore dei sintomi di un paziente dopo una determinata visita presente nel database.	**Fever**(*boolean*): Si attiva nel caso il paziente abbia la febbre. **Cough**(*boolean*): Si attiva nel caso il paziente abbia la tosse. **r_Distress**(*boolean*): Si attiva nel caso il paziente abbia difficoltà respiratorie. **Fatigue**(*boolean*): Si attiva nel caso il paziente si senta particolarmente affaticato. **severiytyL**(*integer*): L'attributo è la somma degli attributi precedenti, necessario per capire quanto lo stato del paziente sia grave.
infoPatient	Descrittore dello stato di quarantena attuale di un paziente presente nel database.	**Lockdown**(*boolean*): Si attiva quando il paziente è costretto alla quarantena. **dateStart**(*date*): Data di inizio quarantena. **dateEnd**(*date*): Data di fine quarantena. **typeL**(*string*): Luogo dove il paziente resterà in quarantena (ospedale o casa).
Contact	Descrittore del tracciamento degli eventuali contatti,in caso di positività al virus, tra due (o più) pazienti presenti nel database.	**dateC**(*date*): Data in cui è avvenuto il contatto. **rType**(*typeRel*): Tipo di relazione tra i due pazienti che hanno avuto il contatto.

Nome	Descrizione	Attributi
Visit	Descrittore di ciascuna visita fatta da un paziente presente nel database.	**idV**(*integer*): Identifica univocamente ogni istanza di Visit. **dateV**(*date*): Data della visita fatta da un paziente.
Swab	Descrittore di ciascun tampone fatto da un paziente dopo una determinata visita presente nel database.	**idSW**(*integer*): Identifica univocamente ogni istanza di Swab. **Results**(*boolean*): Attributo di carattere booleano che mostra se il paziente è risultato negativo o positivo al tampone.
Serological	Descrittore di ciascun test sierologico fatto da un paziente dopo una determinata visita presente nel database.	**idSE**(*integer*): Identifica univocamente ogni istanza di Serological. **Results**(*typeRes*): Attributo di tipo typeRel, i risultati possibili posso essere 'igm' o 'igg'.
Hospital	Descrittore di ciascuna struttura ospedaliera presente nel database.	**idH**(*integer*): Identifica univocamente ogni istanza di Hospital. **hName**(*string*): Nome dell'ospedale. **healed**(*integer*): Numero di pazienti guariti al seguito un test eseguito all'ospedale. **deceased**(*string*): Numero di pazienti deceduti a seguito di complicazioni.
Presence	Descrittore di presenze di un paziente in uno (o più) luoghi presenti nel database.	**timeStart**(*timestamp*): Data ed ora nelle quali il paziente si è trovato in un determinato luogo. **timeEnd**(*timestamp*): Data ed ora nelle quali il paziente ha lasciato il luogo precedente. **idPr**(*integer*): Identifica univocamente ogni istanza di Presence. **Mask**(*boolean*): *0* il paziente non indossa la mascherina, *1* il paziente indossa la mascherina.

2.3.2 Dizionario Dei Vincoli

Nome	Descrizione
Single Phone	Un paziente non puó avere lo stesso numero di cellulare di un altro paziente.
Single eMail	Un paziente non puó avere lo stesso indirizzo eMail di un altro paziente.
Check Visit Dead	Se un paziente é deceduto non puó effettuare una visita.
Check Visit	Si puo' effettuare una visita solo nel caso non si sia gia' guariti dal virus.
Check Death Date	Nel caso un paziente sia deceduto, la data di morte non puó essere minore della data di nascita.

2.3.3 Dizionario Delle Associazioni

Nome	Descrizione	Classi Coinvolte
Residence	Esprime la residenza di un paziente.	**Patient[1..*]** ruolo **lives in**: Indica il paziente che ha residenza in un determinato luogo. **Loc[1]** ruolo **is of**: Indica la residenza del paziente.
Same Place	Esprime la presenza di un paziente in un luogo in un determinato intervallo di tempo.	**Patient[*]** ruolo **is in**: Indica il paziente che si trova nel luogo ed intervallo scelto. **Loc[*]** ruolo **has**: Indica il luogo nel quale si trova un paziente.
infoP	Esprime l'eventuale stato di quarantena di un paziente.	**Patient[1]** ruolo **lives in**: Indica il paziente che si puó trovare o meno in quarantena. **infoPatient[1]** ruolo **is about**: Indica le informazioni di eventuale quarantena relative al paziente.
Tracking	Esprime l'avvenuto contatto tra due o piú pazienti risultati positivi al test.	**Patient[*]** ruolo **infects**: Indica il paziente che ha diffuso ad uno (o piú pazienti) il virus. **Patient[*]** ruolo **intefcted by**: Indica il paziente che é venuto in contatto con una persona portatrice del virus.
Diagnosis	Esprime la diagnosi effettuata in una visita da un paziente al manifestarsi di eventuali sintomi.	**Visit[1]** ruolo **shows**: Indica la visita che comporterá la conferma della diagnosi in base agli eventuali sintomi. **Symptoms[1]** ruolo **are shown**: Indica i sintomi che si sono evinti dalla visita.

Nome	Descrizione	Classi Coinvolte
Visit_P	Esprime l'occorrenza di una o piú visite fatte da un paziente.	**Visit[1..*]** ruolo **done by**: Indica la visita fatta da un paziente. **Patient[1]** ruolo **is of**: Indica il paziente che fa la visita.
Swab_V	Esprime risultati di un tampone naso faringeo al seguito di una visita.	**Visit[1]** ruolo **makes do**: Indica la visita per la quale si passa prima di fare il test. **Swab[0..1]** ruolo **done after**: Indica il tampone naso faringeo fatto dopo la visita.
Serological_V	Esprime risultati di un test sierologico al seguito di una visita.	**Visit[1]** ruolo **makes do**: Indica la visita per la quale si passa prima di fare il test. **Serological[0..1]** ruolo **done after**: Indica il test sierologico fatto dopo la visita.
Hospital_V	Esprime le occorrenze di una visita effettuate in un dato ospedale.	**Visit[1]** ruolo **makes do**: Indica la visita compiuta in un dato ospedale. **Hospital[1]** ruolo **is done**: Indica l'ospedale dove verranno effettuate le visite.
Hospital_L	Esprime l'indirizzo dell'ospedale dove avranno luogo le visite.	**Loc[1]** ruolo **is in**: Indica il luogo dove si trova l'ospedale. **Hospital[1..*]** ruolo **is at**: Indica un ospedale che si trova in un determinato luogo.

Capitolo 3

Progettazione Logica

In questo capitolo si passa alla fase di **progettazione logica**. Le classi create nel Class Diagram (opportunamente ristrutturato) saranno qui presentate sotto forma di **Schemi Relazionali**. Lo schema descrive il Database come una collezione di predicati e di vincoli per mettere in chiaro le eventuali associazioni. La singola sottolineatura indicherà una chiave primaria, una doppia sottolineatura indicherà invece una chiave esterna.

3.1 Schemi Relazionali

LOC (idL,City,p_Code,Street,s_Number,Nation,l_Type)

PRIMARY KEY: $\underline{idL}$

PATIENT (p_Name,p_Surname,idP,b_Date,d_Date,Gender,Phone, eMail,id_PL))

PRIMARY KEY: $\underline{idP}$
FOREIGN KEY: $\underline{\underline{id\_PL}} \longrightarrow Loc(idL)$

HOSPITAL (idH,id_HL,h_Name,healed,deceased)

PRIMARY KEY: $\underline{idH}$
FOREIGN KEY: $\underline{\underline{id\_HL}} \longrightarrow Loc(idL)$

VISIT (idV,date_V,v_Hospital,id_PV)

PRIMARY KEY: $\underline{idV}$
FOREIGN KEY: $\underline{\underline{id\_PV}} \longrightarrow Patient(idP)$ **FOREIGN KEY:** $\underline{\underline{v\_Hospital}} \longrightarrow Hospital(idH)$

SWAB (id_SW,id_VSW,Results)

PRIMARY KEY: id_SW
FOREIGN KEY: $id\_VSW \longrightarrow Visit(idV)$

SEROLOGICAL (id_SE,id_VSE,Results)

PRIMARY KEY: id_SE
FOREIGN KEY: $id\_VSE \longrightarrow Visit(idV)$

SYMPTOMS (Fever,Cough,r_Distress,Fatigue,severity_L,id_SV)

FOREIGN KEY: $id\_SV \longrightarrow Visit(idV)$

INFOPATIENT (Lockdown,date_Start,date_End,p_Location,id_IP)

FOREIGN KEY: $id\_IP \longrightarrow Patient(idP)$

CONTACT (id_PR1,id_PR2,date_C,r_Type,id_PRC1,id_PRC1,)

FOREIGN KEY: $id\_PRC1 \longrightarrow Presence(id\_Pr)$
FOREIGN KEY: $id\_PRC2 \longrightarrow Presence(id\_Pr)$
FOREIGN KEY: $id\_PR1 \longrightarrow Patient(idP)$
FOREIGN KEY: $id\_PR2 \longrightarrow Patient(idP)$

PRESENCE (time_Start,time_End,p_Mask,id_Pr,id_PP,id_PL)

PRIMARY KEY: id_Pr
FOREIGN KEY: $id\_PP \longrightarrow Patient(idP)$
FOREIGN KEY: $id\_PL \longrightarrow Loc(idL)$

Capitolo 4

Progettazione Fisica

L'ultima fase consiste nella vera e propria implementazione del database con il DBMS scelto (*Oracle DB XE 11g*). La traduzione dal modello logico a quello fisico consiste nella creazione delle tabelle contenenti i campi elencati tra le parentesi (Vedi Schemi Relazionali).

4.1 Note Implementative

IL DBMS scelto per l'implementazione è **ORACLE Database 11g Express Edition**. Alcuni identificativi verranno modificati opportunamente per non incappare in errori di compilazione. Gli attributi di tipo *boolean* verranno simulati con un semplice attributo di tipo **NUMBER** o di tipo **CHAR** come negli esempi:

```
    .
    .
    att1 NUMBER(1) CHECK (att1 IN(0,1));
    .
    .
```

Listing 4.1: Esempio con NUMBER

```
    .
    .
    att2 CHAR(1) CHECK (att2 IN('N','P'));
    .
    .
```

Listing 4.2: Esempio con CHAR

4.2 Definizione Delle Tabelle

```
CREATE TABLE Loc (
        idL      INTEGER,
        City     VARCHAR2(15) NOT NULL,
        Street   VARCHAR2(35) NOT NULL,
        s_Number INTEGER NOT NULL,
        l_Type   CHAR(7) CHECK (l_type IN('Outdoor','Indoor')),
        p_Code   CHAR(5) NOT NULL,
        Nation   CHAR(6) NOT NULL,
```

```
9
10          CONSTRAINT PK_idL PRIMARY KEY(idL)
11 );
12
13 CREATE SEQUENCE seq_loc_pk
14 START WITH 1
15 INCREMENT BY 1;
16
17 CREATE OR REPLACE TRIGGER auto_pk_loc
18 BEFORE INSERT ON Loc
19 FOR EACH ROW
20 BEGIN
21     :NEW.idL := seq_loc_pk.NEXTVAL;
22 END;
23 /
```

Listing 4.3: Classe Loc (createDatabase.sql)

```
1  CREATE TABLE Patient(
2          p_Name   VARCHAR2(20) NOT NULL,
3          p_Surname VARCHAR2(20) NOT NULL,
4          idP      INTEGER,
5          b_Date   DATE NOT NULL,
6          d_Date   DATE DEFAULT NULL,
7          Gender   CHAR(1) CHECK(Gender IN('M','F')),
8          Phone    VARCHAR2(10) NOT NULL CHECK (REGEXP_LIKE(Phone,'
    ^((00|\+)39[\. ]??)??3\d{2}[\. ]??\d{6,7}$')),
9          eMail    varchar(320) NOT NULL CHECK (REGEXP_LIKE(eMail,'^[A
    -Za-z]+[A-Za-z0-9.]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$')),
10         id_PL     INTEGER,
11
12         CONSTRAINT PK_idP PRIMARY KEY(idP),
13         CONSTRAINT UQ_eMail UNIQUE(eMail),
14         CONSTRAINT UQ_Phone UNIQUE(Phone),
15         CONSTRAINT FK_idPL FOREIGN KEY(id_PL) REFERENCES Loc(idL)
16        ON DELETE SET NULL
17 );
18
19 CREATE SEQUENCE seq_patient_pk
20 START WITH 1
21 INCREMENT BY 1;
22
23 CREATE OR REPLACE TRIGGER auto_pk_patient
24 BEFORE INSERT ON Patient
25 FOR EACH ROW
26 BEGIN
27     :NEW.idP := seq_patient_pk.NEXTVAL;
28 END;
29 /
```

Listing 4.4: Classe Patient (createDatabase.sql)

```
1     CREATE TABLE Hospital(
2         idH INTEGER,
3         id_HL INTEGER,
4         h_Name VARCHAR2(30) NOT NULL, --mettere unique
5         healed INTEGER DEFAULT 0,
6         deceased INTEGER DEFAULT 0,
7
```

```
        CONSTRAINT PK_H PRIMARY KEY(IdH),
        CONSTRAINT FK_HL FOREIGN KEY(id_HL) REFERENCES Loc(idL)
        ON DELETE SET NULL
);

CREATE SEQUENCE seq_hospital_pk
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER auto_pk_hospital
BEFORE INSERT ON Hospital
FOR EACH ROW
BEGIN
    :NEW.idH := seq_hospital_pk.NEXTVAL;
END;
/
```

Listing 4.5: Classe Hospital (createDatabase.sql)

```
    CREATE TABLE Visit (
        idV   INTEGER,
        date_V  DATE NOT NULL,
        v_Hospital INTEGER NOT NULL,
        id_PV   INTEGER,

        CONSTRAINT PK_idV PRIMARY KEY(idV),
        CONSTRAINT FK_idPV FOREIGN KEY(id_PV) REFERENCES Patient(idp)
        ON DELETE CASCADE,
        CONSTRAINT FK_idHV FOREIGN KEY(v_Hospital) REFERENCES Hospital(idH)
        ON DELETE CASCADE
);

CREATE SEQUENCE seq_visit_pk
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER auto_pk_visit
BEFORE INSERT ON Visit
FOR EACH ROW
BEGIN
    :NEW.idV := seq_visit_pk.NEXTVAL;
END;
/
```

Listing 4.6: Classe Visit (createDatabase.sql)

```
   CREATE TABLE Swab(
        id_SW   INTEGER,
        id_VSW  INTEGER,
        Results CHAR(1) DEFAULT 'n' NOT NULL CHECK(Results IN('n','p')),

        CONSTRAINT PK_idSW PRIMARY KEY(id_SW),
        CONSTRAINT FK_idVSW FOREIGN KEY(id_VSW) REFERENCES Visit(idv)
        ON DELETE CASCADE
);
```

```
10
11 CREATE SEQUENCE seq_swab_pk
12 START WITH 1
13 INCREMENT BY 1;
14
15 CREATE OR REPLACE TRIGGER auto_pk_swab
16 BEFORE INSERT ON Swab
17 FOR EACH ROW
18 BEGIN
19     :NEW.id_SW := seq_swab_pk.NEXTVAL;
20 END;
21 /
```

Listing 4.7: Classe Swab (createDatabase.sql)

```
1    CREATE TABLE Symptoms(
2         Fever    NUMBER(1) NOT NULL CHECK (Fever IN (0,1)),
3         Cough    NUMBER(1) NOT NULL CHECK (Cough IN (0,1)),
4         r_Distress NUMBER(1) NOT NULL CHECK (r_Distress IN (0,1)),
5         Fatigue NUMBER(1) NOT NULL CHECK (Fatigue IN (0,1)),
6         Severity_l INTEGER AS (Fever+Cough+r_Distress+Fatigue),
7         id_SV    INTEGER,
8
9         CONSTRAINT FK_idSV FOREIGN KEY(id_SV) REFERENCES Visit(idV)
10        ON DELETE CASCADE
11 );
```

Listing 4.8: Classe Symptoms (createDatabase.sql)

```
1     CREATE TABLE Serological(
2         id_SE    INTEGER,
3         id_VSE   INTEGER,
4         Results CHAR(8) NOT NULL CHECK(Results IN('negative','igm',
    'igg')),
5
6         CONSTRAINT PK_idSE PRIMARY KEY(id_SE),
7         CONSTRAINT FK_idVSE FOREIGN KEY(id_VSE) REFERENCES Visit(
    idV)
8         ON DELETE CASCADE
9
10 CREATE SEQUENCE seq_serological_pk
11 START WITH 1
12 INCREMENT BY 1;
13
14 CREATE OR REPLACE TRIGGER auto_pk_serological
15 BEFORE INSERT ON Serological
16 FOR EACH ROW
17 BEGIN
18     :NEW.id_SE := seq_serological_pk.NEXTVAL;
19 END;
20 /
21
22 );
```

Listing 4.9: Classe Serological (createDatabase.sql)

```
1    CREATE TABLE infoPatient (
2         Lockdown NUMBER(1) NOT NULL CHECK (Lockdown IN (0,1)),
3         date_Start DATE,
```

```
        date_End DATE,
        p_Location CHAR(8) NOT NULL CHECK (p_Location IN ('Ospedale
    ','Casa')),
        id_IP   INTEGER,

        CONSTRAINT FK_idIP FOREIGN KEY (id_IP) REFERENCES Patient(
    idP)
        ON DELETE CASCADE
);
```

Listing 4.10: Classe infoPatient (createDatabase.sql)

```
   CREATE TABLE Contact (
        id_PR1  INTEGER,
        id_PR2  INTEGER,
        data_C DATE,
        r_Type  CHAR(7) DEFAULT NULL CHECK (r_Type IN ('Amico','
    Collega','Partner','Parente')),
        Gravity VARCHAR2(5),
        id_PRC1 INTEGER,
        id_PRC2 INTEGER,

        CONSTRAINT FK_IRPRC1 FOREIGN KEY(id_PRC1) REFERENCES
    Presence(id_PR)
        ON DELETE CASCADE,
        CONSTRAINT FK_IRPRC2 FOREIGN KEY(id_PRC2) REFERENCES
    Presence(id_PR)
        ON DELETE CASCADE,
        CONSTRAINT FK_idPR1 FOREIGN KEY(id_PR1) REFERENCES Patient(
    idP)
        ON DELETE CASCADE,
        CONSTRAINT FK_IDPR2 FOREIGN KEY(id_PR2) REFERENCES Patient(
    idP)
        ON DELETE CASCADE

);
```

Listing 4.11: Classe Contact (createDatabase.sql)

```
CREATE TABLE Presence (
        time_Start TIMESTAMP,
        time_End TIMESTAMP,
        p_Mask  NUMBER(1) CHECK (p_Mask IN(0,1)),
        id_PRP  INTEGER NOT NULL,
        id_PRL  INTEGER NOT NULL,
        id_PR INTEGER,

        CONSTRAINT PK_IDPR PRIMARY KEY(id_PR),
        CONSTRAINT FK_idPP FOREIGN KEY(id_PRP) REFERENCES Patient(
    idP)
        ON DELETE CASCADE,
        CONSTRAINT FK_idPPL FOREIGN KEY(id_PRL) REFERENCES Loc(idL)
        ON DELETE CASCADE
 );

CREATE SEQUENCE seq_presence_pk
START WITH 1
INCREMENT BY 1;

```

```
CREATE OR REPLACE TRIGGER auto_pk_presence
BEFORE INSERT ON Presence
FOR EACH ROW
BEGIN
    :NEW.id_Pr := seq_presence_pk.NEXTVAL;
END;
/
```

Listing 4.12: Classe Presence (createDatabase.sql)

4.3 Implementazione dei Vincoli

```
-- Nel caso di positivita' di un paziente, il trigger effettua un
   fetch di tutti i dati relativi alle presenze di quest' ultimo e
   nel caso di contatto con altre persone, ha il compito di
   memorizzare i dati nella tabella contact che servira'
   successivamente al Tracking vero e proprio. La gravita' del
   contatto scaturisce da due fattori, dall' aver indossato o meno
   la mascherina e dall' essere o meno in un luogo all'aperto.

CREATE OR REPLACE TRIGGER same_place
AFTER INSERT ON Swab
FOR EACH ROW
WHEN (NEW.Results = 'p')
DECLARE

    /* Cursore per individuare le tuple relative a tutte le persone
    che hanno dichiarato di essere presenti in un determinato luogo
    ad una certa ora
       escluse quelle che sono risultate gi    positive ad un
   tampone naso-faringeo */
    CURSOR cur1 IS(
                SELECT *
                FROM Presence Pr JOIN Patient P ON P.idP=Pr.id_PRP
   FULL JOIN Contact C ON C.id_prc2=Pr.id_pr OR C.id_prc1=Pr.id_pr
                MINUS
                SELECT *
                FROM Presence Pr JOIN Patient P ON P.idP=Pr.id_PRP
   RIGHT JOIN Contact C ON C.id_prc2=Pr.id_pr OR C.id_prc1=Pr.id_pr
   );


    /* Cursore one-row che individua le informazioni relative al
   paziente che      risultato positivo al tampone escluse
   occorrenze ridondanti relative
       a presenze passate */
    CURSOR cur2 IS
            (SELECT *
                FROM Presence Pr JOIN Patient P ON P.idP=Pr.id_PRP
   FULL JOIN Contact C ON C.id_prc2=Pr.id_pr OR C.id_prc1=Pr.id_pr
   JOIN Visit V ON P.idP=V.id_PV JOIN Loc L ON L.idL=Pr.id_PRL
                WHERE :NEW.id_VSW = V.idV
                MINUS
                SELECT *
                FROM Presence Pr JOIN Patient P ON P.idP=Pr.id_PRP
   RIGHT JOIN Contact C ON C.id_prc2=Pr.id_pr OR C.id_prc1=Pr.id_pr
    JOIN Visit V ON P.idP=V.id_PV JOIN Loc L ON L.idL=Pr.id_PRL
```

```
28                  WHERE :NEW.id_VSW = V.idV);
29
30      np cur2%ROWTYPE;
31
32  BEGIN
33      FOR curVal IN cur1
34      LOOP
35          OPEN cur2;
36          -- Fetching del cursore relativo al paziente positivo.
37          FETCH cur2 INTO np;
38
39          /* Per individuare i pazienti a contatto e inseririli
    opportunamente nella tabella Contact. Un paziente non pu
    mischiare se stesso(curVal.id_PRP<>np.id_PRP),
40              pu    mischiarne un altro solo se quest'ultimo si trova
     nello stesso luogo (curVal.id_PRL=np.id_PRL) e nello stesso
    intervallo di tempo nel quale era presente
41              il paziente positivo */
42
43         IF ((curVal.id_PRP<>np.id_PRP) AND (curVal.id_PRL=np.id_PRL)
    AND
44              (np.time_Start>=curVal.time_Start AND np.time_Start<=
    curVal.time_End) OR
45              (curVal.time_Start>=np.time_Start AND curVal.time_Start
    <=np.time_End)) THEN
46              -- Nel caso il tipo di luogo sia al chiuso
47           IF np.l_type = 'Indoor' THEN
48                  -- Casi in cui il paziente positivo indossi la
    mascherina o meno
49              IF np.p_mask = 0 THEN
50                 INSERT INTO Contact(id_PR1,id_PR2,data_C,Gravity,
    id_PRC1,id_PRC2) VALUES(np.idP, curVal.idP,CAST((curVal.
    time_start) AS DATE),'Alta',np.id_PR,curVal.id_PR);
51              ELSE
52                 INSERT INTO Contact(id_PR1,id_PR2,data_C,Gravity,
    id_PRC1,id_PRC2) VALUES(np.idP, curVal.idP,CAST((curVal.
    time_start) AS DATE),'Media',np.id_PR,curVal.id_PR);
53              END IF;
54              -- Nel caso il tipo di luogo sia all'aperto
55           ELSE
56           -- Casi in cui il paziente positivo indossi la mascherina
    o meno
57                 IF np.p_mask = 1 THEN
58                 INSERT INTO Contact(id_PR1,id_PR2,data_C,Gravity,
    id_PRC1,id_PRC2) VALUES(np.idP, curVal.idP,CAST((curVal.
    time_start) AS DATE),'Bassa',np.id_PR,curVal.id_PR);
59              ELSE
60                 INSERT INTO Contact(id_PR1,id_PR2,data_C,Gravity,
    id_PRC1,id_PRC2) VALUES(np.idP, curVal.idP,CAST((curVal.
    time_start) AS DATE),'Media',np.id_PR,curVal.id_PR);
61                  END IF;
62           END IF;
63        END IF;
64          CLOSE cur2;
65          END LOOP;
66  END;
67  /
```

Listing 4.13: Trigger Presenze (Triggers.sql)

```
-- Il trigger ha il compito di settare in automatico la quarantena
   nel caso il paziente sia risultato positivo al tampone. Nel caso
    i sintomi siano eccessivi, il paziente dovra' rimanere in
   quarantena in una struttura ospedaliera, nel caso contrario la
   quarantena puo' essere svolta nella propria residenza.

CREATE OR REPLACE TRIGGER setQuarantine
AFTER INSERT ON Swab
FOR EACH ROW
WHEN (NEW.Results = 'p')
DECLARE
   s Symptoms.severity_L%TYPE;
   idn Patient.idP%TYPE;
BEGIN
   SELECT P.idP, S.severity_L INTO idn,s
   FROM Patient P JOIN Visit V ON idP=id_PV
         JOIN Symptoms S ON V.idV=S.id_SV
         WHERE :NEW.id_VSW = V.idV;

   IF s >=2 THEN
      INSERT INTO infoPatient VALUES(1,:NEW.date_test,:NEW.
   date_test+14,'Ospedale',idn);
   ELSE
      INSERT INTO infoPatient VALUES(1,:NEW.date_test,:NEW.
   date_test+14,'Casa',idn);
   END IF;
END;
/
```

Listing 4.14: Trigger Quarantena (Triggers.sql)

```
-- Nel caso di decesso di un paziente viene incrementato l'
   attributo deceased appartenente all'ospedale dove e' avvenuta la
    visita.

CREATE OR REPLACE TRIGGER h_deceased
AFTER UPDATE OF d_Date ON Patient
FOR EACH ROW
WHEN (OLD.d_Date IS NULL)
DECLARE
    hd Hospital.idh%TYPE;
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    SELECT idH INTO hd
    FROM ( SELECT *
           FROM Visit V JOIN Hospital H ON H.idH=V.v_Hospital
                JOIN Patient P ON V.id_PV=P.idP
           ORDER BY V.idV DESC )
    WHERE ROWNUM = 1;

    UPDATE Hospital
    SET deceased = deceased+1
    WHERE idH=hd;

    COMMIT;
END;
/
```

Listing 4.15: Trigger Deceduto

```
-- Nel caso di guarigione di un paziente, confermata tramite tampone, viene incrementato l'attributo healed appartenente all'ospedale dove e' avvenuta la visita.

CREATE OR REPLACE TRIGGER sw_healed
AFTER INSERT ON Swab
FOR EACH ROW
WHEN (NEW.Results = 'n')
DECLARE

   CURSOR cur IS (
        SELECT *
        FROM Patient P JOIN Visit V ON P.idP=V.id_PV JOIN Hospital H ON H.idH=V.v_Hospital
             JOIN Swab Sw ON V.iDV=Sw.id_VSW
       WHERE Sw.Results = 'p'
    );

   res number := 0;
   hp Patient.idP%TYPE;
    idhc Hospital.idH%TYPE;
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   SELECT P.idP INTO hp
    FROM Patient P JOIN Visit V ON P.idP=V.id_PV
    WHERE :NEW.id_VSW=V.idV;

    FOR ch IN cur
    LOOP
        IF ch.idP=hp THEN
       res := 1;
        idhc := ch.idH;
        EXIT;
   END IF;
    END LOOP;

    IF res=1 THEN
        UPDATE Hospital
        SET healed = healed+1
        WHERE idH=idhc;

      UPDATE infoPatient
        SET Lockdown = 0
        WHERE id_IP=hp;

        COMMIT;
    END IF;
END;
/
```

Listing 4.16: Trigger Guarito (Tampone)

```
-- Nel caso di guarigione di un paziente, confermata tramite test sierologico, viene incrementato l'attributo healed appartenente all'ospedale dove e' avvenuta la visita.

CREATE OR REPLACE TRIGGER se_healed
AFTER INSERT ON Serological
FOR EACH ROW
```

```
WHEN (NEW.Results = 'igg' OR NEW.Results = 'Negative')
DECLARE

   CURSOR cur IS (
        SELECT *
        FROM ((Patient P JOIN Visit V ON P.idP=V.id_PV) JOIN
   Hospital H ON H.idH=V.v_Hospital
             JOIN Swab Sw ON V.iDV=Sw.id_VSW)
        WHERE Sw.Results = 'p'
    );

   res number := 0;
   idhc Hospital.idH%TYPE;
   hp Patient.idP%TYPE;
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   SELECT P.idP INTO hp
    FROM Patient P JOIN Visit V ON P.idP=V.id_PV
    WHERE :NEW.id_VSE=V.idV;

    FOR ch IN cur
    LOOP
        IF ch.idP=hp THEN
            res := 1;
            idhc := ch.idH;
   END IF;
    END LOOP;

    IF res=1 THEN

        UPDATE Hospital
        SET healed = healed+1
        WHERE idH=idhc;

      UPDATE infoPatient
        SET Lockdown = 0
        WHERE id_IP=hp;

        COMMIT;
    END IF;

END;
/
```

Listing 4.17: Trigger Guarito (Sierologico)

```
-- La data del decesso non puo' essere minore di quella della
   nascita.

CREATE OR REPLACE TRIGGER wrong_date
BEFORE UPDATE ON Patient
FOR EACH ROW
DECLARE
BEGIN
   IF :NEW.d_Date < :NEW.b_Date THEN
      RAISE_APPLICATION_ERROR(-20111,'Error. Wrong date.');
    END IF;
END;
```

```
/
```

Listing 4.18: Trigger Data Decesso

```
-- Il  trigger permette l'inserimento di un nuovo paziente (vedi
   vista Residence)
CREATE OR REPLACE TRIGGER newpatient
INSTEAD OF INSERT ON Residence
FOR EACH ROW
DECLARE
    nl Loc.idL%TYPE;
BEGIN
    INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation)
   VALUES(:NEW.City,:NEW.Street,:NEW.s_Number,:NEW.l_type,:NEW.
   p_code,:NEW.Nation)
    RETURNING idL INTO nl;
    INSERT INTO Patient(p_Name,p_Surname,b_Date,Gender,Phone,eMail,
   id_PL) VALUES(:NEW.P_Name,:NEW.P_Surname,:NEW.B_Date,:NEW.Gender
   ,:NEW.Phone,:NEW.eMail,nl);
END;
/
```

Listing 4.19: Trigger Visita (Triggers.sql)

```
-- Il  trigger permette l'inserimento di una nuova visita (vedi
   vista nVisit)
CREATE OR REPLACE TRIGGER newvisit
INSTEAD OF INSERT ON nVisit
FOR EACH ROW
DECLARE
    nv Visit.idV%TYPE;
BEGIN
    INSERT INTO Visit(date_V,v_Hospital,id_PV) VALUES(:NEW.date_V,:
   NEW.v_Hospital,:NEW.id_PV)
    RETURNING idV INTO nv;
    INSERT INTO Symptoms(Fever,Cough,r_Distress,Fatigue,Severity_l,
   id_SV) VALUES(:NEW.Fever,:NEW.Cough,:NEW.r_Distress,:NEW.Fatigue
   ,:NEW.Severity_l,nv);
END;
/
```

Listing 4.20: Trigger Visita (Triggers.sql)

```
-- Il  trigger permette l'inserimento di un nuovo ospedale (vedi
   vista nHospital)

CREATE OR REPLACE TRIGGER newhospital
INSTEAD OF INSERT ON nHospital
FOR EACH ROW
DECLARE
    nh Loc.idL%TYPE;
BEGIN
    INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation)
   VALUES(:NEW.City,:NEW.Street,:NEW.s_Number,:NEW.l_type,:NEW.
   p_code,:NEW.Nation)
    RETURNING idL INTO nh;
    INSERT INTO Hospital(id_HL,h_Name) VALUES(nh,:NEW.h_Name);
END;
```

```
/
```

Listing 4.21: Trigger Nuovo Ospedale (Triggers.sql)

4.4 Implementazione delle Viste

```
-- La vista mostra il nome della persona che ha diffuso il virus a
   fianco della persona che ne e' stata a contatto, la data dell'
   avvenuto contagio,il tasso di probabilita' di aver contratto il
   virus e l'eventuale relazione.

CREATE OR REPLACE VIEW Tracking (ID1,NomeDiffusore,CognomeDiffusore
   ,NomeContagiato,ID2,CognomeContagiato,DataContagio,Gravita,
   Relazione) AS
SELECT C.id_PR1,P1.p_Name,P1.p_Surname,C.id_PR2,P2.p_Name,P2.
   p_Surname,C.Data_C,C.Gravity,C.R_Type
FROM Contact C JOIN Patient P1 ON C.id_PR1=P1.idP JOIN Patient P2
   ON C.id_PR2=P2.idP;
```

Listing 4.22: Vista TrackingP (createDatabase.sql)

```
-- Vista che rappresenta tutte le occorrenze presenti nel Database.

CREATE OR REPLACE VIEW v_Presence AS
SELECT Pr.id_PR,P.p_Name,P.p_Surname, Pr.time_Start,Pr.time_End, Pr
   .p_Mask, L.City, L.Street, L.s_Number
FROM Patient P JOIN Presence Pr ON P.idP=Pr.id_PRP JOIN Loc L ON L.
   idL=Pr.id_PRL;
```

Listing 4.23: Vista Occorrenze (createDatabase.sql)

```
-- La vista mostra il totale di persone che si trovano in qurantena
    a casa in una certa citta'.

CREATE OR REPLACE VIEW quarantenacasa AS
SELECT L.City, COUNT(P.idP) QP
FROM Patient P JOIN Visit V ON V.id_PV=P.idP JOIN Hospital H ON V.
   v_Hospital=H.idH
JOIN Loc L ON L.idL=H.id_HL JOIN infoPatient I ON P.idP=I.id_IP
WHERE I.Lockdown=1 AND I.p_Location='Casa'
GROUP BY L.City;
```

Listing 4.24: Vista Quarantena Casa (createDatabase.sql)

```
-- La vista mostra il totale di persone che si trovano in qurantena
    in ospedale in una certa citta'.

CREATE OR REPLACE VIEW quarantenaospedale AS
SELECT L.City, COUNT(P.idP) QO
FROM Patient P JOIN Visit V ON V.id_PV=P.idP JOIN Hospital H ON V.
   v_Hospital=H.idH
JOIN Loc L ON L.idL=H.id_HL JOIN infoPatient I ON P.idP=I.id_IP
WHERE I.Lockdown=1 AND I.p_Location='Ospedale'
GROUP BY L.City;
```

Listing 4.25: Viste Quarantena Ospedale (createDatabase.sql)

```sql
-- La vista mostra il totale di persone guarite in una citta'.

CREATE OR REPLACE VIEW vhealed AS
SELECT L.City,SUM(H.healed) hh
FROM Hospital H JOIN Loc L ON L.idL=H.id_HL
GROUP BY L.City;
```

Listing 4.26: Vista Guariti (createDatabase.sql)

```sql
-- La vista mostra il totale di persone decedute in una citta'.

CREATE OR REPLACE VIEW vdeceased AS
SELECT L.City, SUM(H.deceased) hd
FROM Hospital H JOIN Loc L ON L.idL=H.id_HL
GROUP BY L.City;
```

Listing 4.27: Vista Deceduti (createDatabase.sql)

```sql
-- La vista mostra il totale di test avvenuti in una citta'.

CREATE OR REPLACE VIEW totalv AS
SELECT L.City, COUNT(V.idV) QV
FROM Visit V JOIN Hospital H ON V.v_Hospital=H.idH
JOIN Loc L ON L.idL=H.id_HL
GROUP BY L.City;
```

Listing 4.28: Vista Totale Test (createDatabase.sql)

```sql
-- La vista raggruppa le precedenti per rappresentare la situazione
    nella quale si trova l' intera Italia.

CREATE OR REPLACE VIEW current_s(QuarantenaCasa,QuarantenaOspedale,
   Deceduti,Guariti,TotaleTest) AS
SELECT
        (
        SELECT SUM(qp) FROM quarantenacasa
        ),
        (
        SELECT SUM(QO) FROM quarantenaospedale
        ),
        (
         SELECT SUM(hh) FROM vhealed
         ),
        (
        SELECT SUM(hd) FROM vdeceased
        ),
        (
        SELECT SUM(qv) FROM totalv
        )
FROM DUAL;
```

Listing 4.29: Vista Situazione Italia (createDatabase.sql)

```sql
-- Le seguenti sono viste per effettuare inserimenti
   rispettivamente per una nuova visita, per un nuovo paziente (
   dati anagrafici e dati residenza) e per un nuovo ospedale.

CREATE OR REPLACE VIEW nVisit AS
SELECT V.date_V,V.v_Hospital,V.id_PV,S.Fever,S.Cough,S.r_Distress,S
   .Fatigue,S.Severity_l,S.id_SV
```

```
FROM Visit V, Symptoms S;

------------

CREATE OR REPLACE VIEW Residence AS
SELECT L.City,L.Street,L.s_Number,L.l_type,L.p_code,L.Nation,P.
   P_Name,P.P_Surname,P.B_Date,P.Gender,P.Phone,P.eMail,P.id_PL
FROM Loc L, Patient P;
------------

CREATE OR REPLACE VIEW nHospital AS
SELECT L.City,L.Street,L.s_Number,L.l_type,L.p_code,L.Nation,H.
   id_HL,H.h_Name
FROM Loc L, Hospital H;
```

Listing 4.30: Viste per inserimenti (createDatabase.sql)

Capitolo 5

Esempio d'uso

5.1 Data Entry

Di seguito lo Script per popolare il database con dati d'esempio.

```
1  -- LOC
2  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Via Napoli',9,'Indoor',80124,'Italia');
3  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Via Marina',10,'Indoor',80121,'Italia');
4  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Via Montagna Spaccata',5,'Indoor',80126,'Italia');
5  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Roma','Via Del Corso',48,'Indoor',80987,'Italia');
6  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Roma','Via Cesare',2,'Indoor',80986,'Italia');
7  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Roma','Corso Italia',2,'Indoor',80986,'Italia');
8  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Milano','Corso Buenos Aires',33,'Indoor',80032,'Italia');
9  INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Milano','Via Paolo Sarpi',25,'Indoor',80039,'Italia');
10 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Milano','Corso Italia',3,'Indoor',80039,'Italia');
11 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Piazzale Tecchio',1,'Outdoor',80125,'Italia');
12 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Via Medina',13,'Outdoor',80121,'Italia');
13 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Piazza Sannazzaro',2,'Outdoor',80123,'Italia');
14 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Via Manzoni',58,'Outdoor',80124,'Italia');
15 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Napoli','Vico Belledonne',9,'Indoor',80121,'Italia');
16 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Roma','Piazza Del Popolo',20,'Outdoor',80985,'Italia');
17 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Roma','Viale Trastevere',9,'Indoor',80987,'Italia');
18 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Milano','Via Porta Romana',22,'Outdoor',80200,'Italia');
19 INSERT INTO Loc(City,Street,s_Number,l_Type,p_Code,Nation) VALUES('
       Milano','Via della Moscova',12,'Indoor',80200,'Italia');
20 COMMIT;
21
```

```
22 -- PATIENT
23 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Mario','Mari',TO_DATE('1985-08-05','YYYY-MM
      -DD'),NULL,'M',323457854,'mariomari@gmail.com',1);
24 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Al','Fonso',TO_DATE('1970-09-01','YYYY-MM-
      DD'),NULL,'M',3295261234,'alfonso@live.it',2);
25 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Ciro','Esposito',TO_DATE('1994-02-06','YYYY
      -MM-DD'),NULL,'M',3245681212,'ciro.e@gmail.com',3);
26 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Anna','Donnanna',TO_DATE('2000-01-23','YYYY
      -MM-DD'),NULL,'F',3332587845,'annadonn@gmail.com',1);
27 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Serena','Pasqua',TO_DATE('1978-07-12','YYYY
      -MM-DD'),NULL,'F',3401236521,'serena.p@hotmail.com',4);
28 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Elena','Fabrizi',TO_DATE('1935-05-17','YYYY
      -MM-DD'),NULL,'F',3317742544,'elena.f@live.com',5);
29 INSERT INTO Patient(p_Name,p_Surname,b_Date,d_Date,Gender,Phone,
      eMail,id_PL) VALUES('Brambilla','Fumagalli',TO_DATE('1985-08-05'
      ,'YYYY-MM-DD'),NULL,'M',3384952110,'cadrega@gmail.com',7);
30
31 --PRESENCE
32
33 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-02 18:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-02 23:30','YYYY-MM-DD HH24:MI:SS')),0,1,1)
      ; --Mario,Anna
34 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-02 18:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-02 23:30','YYYY-MM-DD HH24:MI')),0,4,1);
35 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-04 10:15','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-04 12:30','YYYY-MM-DD HH24:MI')),1,4,10);
      -- Anna,Ciro
36 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-04 10:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-04 12:30','YYYY-MM-DD HH24:MI')),1,3,10);
37 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-04 11:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-05 12:30','YYYY-MM-DD HH24:MI')),0,3,11);
      -- Ciro,Serena,Al
38 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-05 11:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-05 16:45','YYYY-MM-DD HH24:MI')),0,5,11);
39 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-05 15:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-05 17:45','YYYY-MM-DD HH24:MI')),0,2,11);
40 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-06 09:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-06 14:00','YYYY-MM-DD HH24:MI')),0,5,15);
      --Serena,Elena
41 INSERT INTO Presence(time_Start,time_End,p_Mask,id_PRP,id_PRL)
      VALUES(TO_TIMESTAMP('2020-10-06 09:30','YYYY-MM-DD HH24:MI'),(
      TO_TIMESTAMP('2020-10-06 14:00','YYYY-MM-DD HH24:MI')),0,6,15);
42 COMMIT;
43
```

```
44 --HOSPITAL
45 INSERT INTO Hospital(id_HL,h_name) VALUES(1,'Ospedale San Paolo');
46 INSERT INTO Hospital(id_HL,h_name) VALUES(1,'Ospedale San Leonardo'
      );
47 INSERT INTO Hospital(id_HL,h_name) VALUES(1,'Ospedale Gemelli');
48 COMMIT;
```

Listing 5.1: Popolamento Database (DataEntry.sql)