Progetto 20080110 - (P.20080110) - QuickHospital

Sebastiano Deodati

26 maggio 2024

Indice

Dati di interesse e funzionalità richieste

1. Personale
    1.1. Nome
    1.2. Cognome
    1.3. Data di nascita
    1.4. Pazienti
        1.4.1. Telefono
        1.4.2. Email
        1.4.3. Casella postale
        1.4.4. Pazienti esterni
            1.4.4.1. Medico curante (req. 1.5)
            1.4.4.2. Prestazione richiesta (req. 5)
2. Medici
    2.1. Specializzazione primaria (req. 6)
    2.2. Eventuali specializzazioni secondarie (req. 6)
    2.3. Pazienti (req. 1.4)
3. Stanze
    3.1. Piano
    3.2. Settore
    3.3. Posti letto (da 1 a 8) (req. 3)
4. Posti letto
    4.1. Disponibilità
5. Ricoveri
    5.1. Paziente (req. 1.4)
    5.2. Posto letto (req. 3)
6. Prestazione medica
    6.1. Specializzazione (req. 6)
    6.2. Descrizione

7. Specializzazione

   7.1. Nome

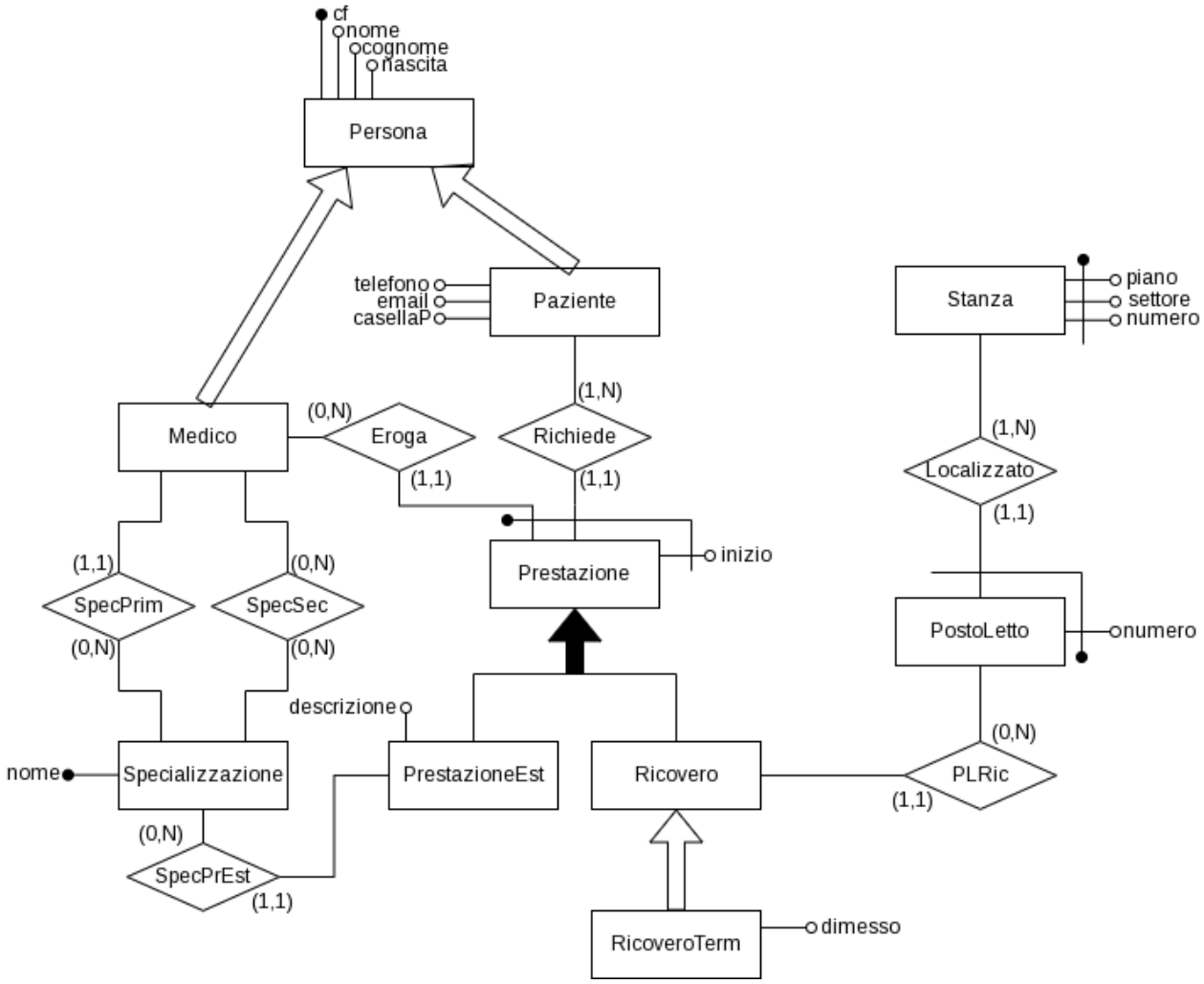
8. Funzionalità richieste:

   8.1. Il sistema deve consentire al personale di accettazione di registrare ricoveri e dimissioni dei pazienti

   8.2. Il sistema deve assistere i medici nel programmare il loro itinerario di visite, cioè un insieme ordinato di stanze da visitare, dato da tutte e sole le stanze in cui sono ricoverati pazienti in cura presso quel medico. L'itinerario deve essere ottimizzato tramite ordinamento in primo luogo rispetto al piano, e in secondo luogo rispetto al settore

   8.3. Il sistema deve consentire, data una prestazione esterna di specializzazione s richiesta da un paziente esterno, di ottenere l'insieme dei medici maggiormente idonei ad erogarla. In particolare, se esistono medici con specializzazione primaria s, viene restituito l'insieme di tali medici, altrimenti viene restituito l'insieme dei medici che hanno s tra le specializzazioni secondarie

Diagramma ER



Dizionario dei dati

Entità Persona

attributo	tipo	note
cf	cf_t	introdotto per identiticare univocamente le persone
nome	stringa	
cognome	stringa	
nascita	data	

Entità Paziente

attributo	tipo	note
telefono	numtel	
email	email_t	
casellaP	indirizzo	

V.Paziente.ricoveri: $\forall p, r, i\, \text{Paziente}(p) \wedge \text{Ricovero}(r) \wedge \text{Richiede}(p, r)$
$\wedge\, \text{inizio}(r, i)\ \wedge\ \neg\exists d\, [\text{dimesso}(r, d)]\ \rightarrow\ \neg\exists i' > i, r' \neq r\, [\text{Richiede}(p, r') \wedge \text{inizio}(r', i')]$
(finché il paziente non è dimesso, non può essere ricoverato né richiedere prestazioni sanitarie)

Entità PostoLetto

attributo	tipo	note
numero	intero[1,8]	

Entità Stanza

attributo	tipo	note
piano	intero	0 indica piano terra, numeri negativi indicano sotterranei
settore	intero > 0	
numero	intero > 0	

Entità Specializzazione

attributo	tipo	note
nome	stringa	

Entità Prestazione

attributo	tipo	note
inizio	dataora	

Entità PrestazioneEst

attributo	tipo	note
descrizione	stringa	

Entità RicoveroTerm

attributo	tipo	note
dimesso	dataora	

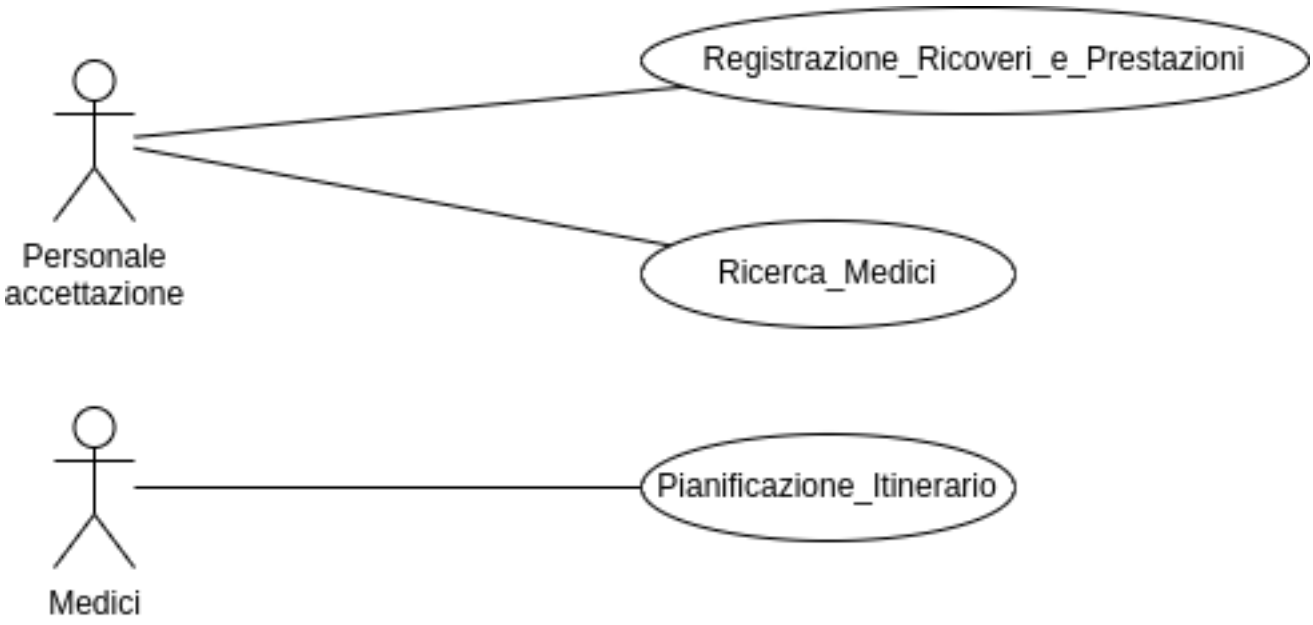
V.RicoveroTerm.dimissione: $\forall r, i, f \;\; \text{Ricovero}(r) \wedge \text{inizio}(r, i) \wedge \text{dimesso}(r, f) \rightarrow i \leq f$

3.1 Tipi di dati personalizzati

cf_t: stringa in formato codice fiscale
numtel: stringa che rappresenta un numero di telefono valido
email_t: stringa che rappresenta un'email valida
indirizzo: record:

- via_pza: stringa
- civico: intero ≤ 0 (0 per SNC)
- CAP: stringa CAP (5 cifre)
- Città: stringa

UML


Registrazione_Ricoveri_e_Prestazioni
Personale accettazione
Ricerca_Medici
Pianificazione_Itinerario
Medici

Specifiche degli use-case

5.1 Specifiche use-case Registrazione_Ricoveri_e_Prestazioni

Registra_Paziente(cf: cf_t, nome: stringa, cognome: stringa, nascita: data,
telefono: numtel, email: email_t, casella: indirizzo) : Paziente
precondizioni: $\neg\exists p[\text{Paziente}(p) \wedge \text{cf}(p, cf)] \wedge \text{nascita} \leq oggi$
postcondizioni:
modifica al livello estensionale dei dati:
nuovi elementi del dominio di interpretazione: p
nuove ennuple di predicati:
Persona(p)
Paziente(p)
cf(p, cf)
nome(p, nome)
cognome(p, cognome)
nascita(p, nascita)
telefono(p, telefono)
email(p, email)
casellaP(p, casella)
valore di ritorno: result = p

Registra_Ricovero(p: Paziente, medico: Medico, pl: PostoLetto) : Ricovero
precondizioni: $\forall r$ Ricovero$(r) \wedge$ Richiede$(p, r)\exists d[\text{dimesso}(r, d)]$
postcondizioni:
modifica al livello estensionale dei dati:
nuovi elementi del dominio di interpretazione: r
nuove ennuple di predicati:
Ricovero(r)
Richiede(p, r)
Eroga(medico, r)
inizio(r, *adesso*)
valore di ritorno: result = r

Dimetti(r: Ricovero) : RicoveroTerm
precondizioni: nessuna
postcondizioni:
modifica al livello estensionale dei dati:
dominio di interpretazione: invariato
nuove ennuple di predicati:
RicoveroTerm(r)
dimesso(r, *adesso*)
valore di ritorno: result = r

Registra_Prestazione(p: Paziente, medico: Medico, desc: Descrizione,
spec: Specializzazione) : PrestazioneEst
precondizioni: SpecPrim(*medico*, *spec*) $\vee$ SpecSec(*medico*, *spec*)
postcondizioni:
modifica al livello estensionale dei dati:
nuovi elementi del dominio di interpretazione: pr
nuove ennuple di predicati:
PrestazioneEst(pr)
Richiede(p, pr)
Eroga(medico, pr)
SpecPrEst(pr, spec)
inizio(pr, *adesso*)
descrizione(pr, desc)
valore di ritorno: result = pr

5.2 Specifiche use-case Ricerca_Medici

Ricerca_Medico(spec: Specializzazione) : Medico(0,N)
precondizioni: nessuna
postcondizioni:
modifica al livello estensionale dei dati: nessuna
valore di ritorno:
sia $S_1 = \{m|\text{Medico}(m) \wedge \text{SpecPrim}(m, spec)\}$
sia $S_2 = \{m|\text{Medico}(m) \wedge \text{SpecSec}(m, spec)\}$
result = S_1 se $|S_1| > 0$ S_2 altrimenti

5.3 Specifiche use-case Pianificazione_Itinerario

Pianifica_Itinerario(m: Medico) : (intero, intero)(0,N)

precondizioni: nessuna

postcondizioni:

modifica al livello estensionale dei dati: nessuna

valore di ritorno:

sia $P = \{(r, pa, pl, s, p, st) | \text{Ricovero}(r) \wedge \text{Richiede}(pa, r) \wedge \text{Eroga}(m, r) \wedge \text{PLRic}(r, pl) \wedge \text{Localizzato}(pl, s) \wedge \text{piano}(s, p) \wedge \text{settore}(s, st)\}$

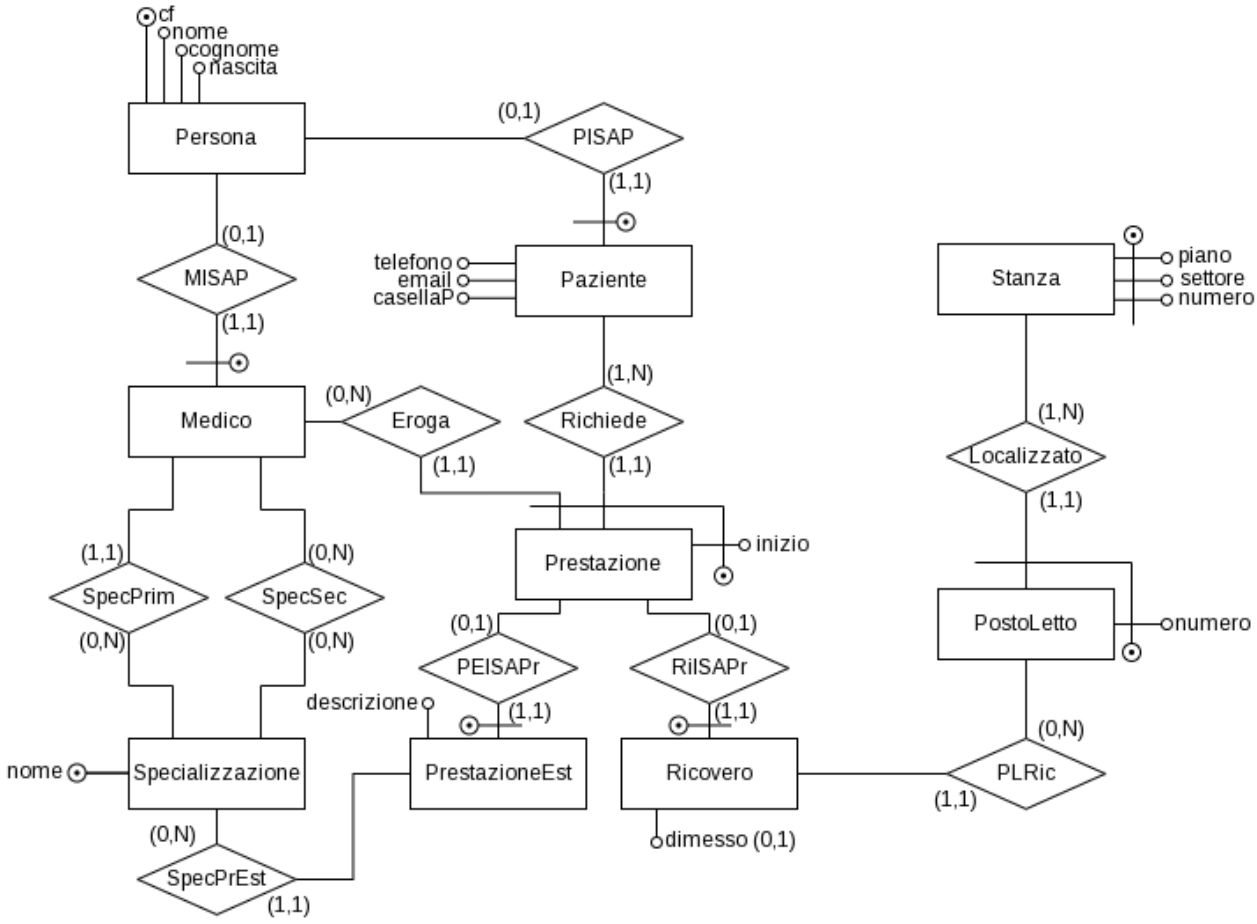
sia $it = \{(p, st) \forall (r, pa, pl, s, p, st) \in P\}$

result = it ordinato per p in primo luogo, per st in secondo luogo

Scelta del DBMS

Il database verrà implementato sul DBMS PostGreSQL.

Ristrutturazione del diagramma ER



Transizione dei tipi di dati concettuali in tipi standard SQL

Entità Persona

attributo	tipo	note
cf	cf_t	introdotto per identiticare univocamente le persone
nome	stringM	
cognome	stringM	
nascita	date	

Entità Paziente

attributo	tipo	note
telefono	numtel	
email	email_t	
casellaP	indirizzo	

V.Paziente.isa: $\forall pa\,\text{Paziente}(p) \;\rightarrow\; \exists p\,\text{PISAP}(pa, p)$
V.Paziente.ricoveri: $\forall p, r, pr, i\,\text{Paziente}(p) \wedge \text{Richiede}(p, pr) \wedge \text{RiISAPr}(r, pr)$
$\wedge\,\text{inizio}(pr, i) \;\wedge\; \neg\exists d\,[\text{dimesso}(r, d)]$
$\rightarrow\; \neg\exists i' > i, pr' \neq r\,[\text{Richiede}(p, pr') \wedge \text{inizio}(pr', i')]$
(finché il paziente non è dimesso, non può essere ricoverato né richiedere prestazioni sanitarie)

Entità Medico

V.Medico.isa: $\forall m\,\text{Medico}(m) \;\rightarrow\; \exists p\,\text{MISAP}(m, p)$

Entità PostoLetto

attributo	tipo	note
numero	pl_t	

Entità Stanza

attributo	tipo	note
piano	integer	0 indica piano terra, numeri negativi indicano sotterranei
settore	int_pos	
numero	int_pos	

Entità Specializzazione

attributo	tipo	note
nome	stringL	

Entità Prestazione

attributo	tipo	note
inizio	datetime	

V.Prestazione.isa: $\forall p\,\text{Prestazione}(p) \rightarrow [\exists pr\text{RiISAPr}(pr,p)] \vee [\exists pe\text{PEISAPr}(pe,p)]$
V.Prestazione.disg: $\forall p\,\text{Prestazione}(p) \rightarrow [\exists pr\text{RiISAPr}(pr,p)] \rightarrow [\neg\exists pe\text{PEISAPr}(pe,p)]$

Entità PrestazioneEst

attributo	tipo	note
descrizione	stringL	

Entità Ricovero

attributo	tipo	note
dimesso(0,1)	datetime	NULL indica che il paziente non è ancora stato dimesso

V.Ricovero.dimissione: $\forall p, r, i, f$ $\text{Prestazione}(p) \wedge \text{inizio}(r,i) \wedge \text{RiISAPr}(r,p) \wedge \text{dimesso}(r,f)$
$\rightarrow i \leq f$

8.1 Tipi di dati personalizzati

```
create domain stringM as varchar(50);
create domain stringL as varchar(100);
create domain int_pos as integer check value > 0;
create domain pl_t as integer check value between 1 and 8;
create domain cf_t as char(16)
    check value ~* '^[a-z]{6}[0-9]{2}[a-z][0-9]{2}[a-z][0-9]{3}[a-z]$';
create domain numtel as char(13) check value ~* '^\+[0-9]{12}$';
create domain email_t as stringL check value ~* '^[a-z0-9_.-]+@[a-z0-9_.-]+\.[a-z]+$';
create type indirizzo (
    via_pza stringL not null,
    civico int_pos,       // NULL per SNC
    cap char(5) not null check value ~* '^[0-9]{5}$',
    citta stringM not null
);
```

Schema concettuale

Persona(cf: cf_t, nome: stringM, cognome: stringM, nascita: date)
vincolo ennupla: nascita $\leq$ CURRENT_DATE

Paziente(cf: cf_t, telefono: numtel, email: email_t, casellaP: indirizzo)
vincolo foreign key: (cf) references Persona(cf)
vincolo ennupla: (cf) $\subseteq$ Prestazione(paziente)

Medico(cf: cf_t, specPrim: stringL)
vincolo foreign key: (cf) references Persona(cf)
vincolo foreign key: (specPrim) references Specializzazione(nome)

SpecSec(medico: cf_t, spec: stringL)
vincolo foreign key: (cf) references Medico(cf)
vincolo foreign key: (spec) references Specializzazione(nome)

PostoLetto(numero: pl_t, piano: integer, settore: int_pos, stanza: int_pos)
vincolo foreign key: (piano, settore, stanza) references Stanza(piano, settore, numero)

Stanza(numero: int_pos, piano: integer, settore: int_pos)

Specializzazione(nome: stringL)

Prestazione(paziente: cf_t, medico: cf_t, inizio: datetime)
vincolo foreign key: (paziente) references Paziente(cf)
vincolo foreign key: (medico) references Medico(cf)

PrestazioneEst(paziente: cf_t, medico: cf_t, inizio: datetime, specializzazione: stringL, descrizione: stringL)
vincolo foreign key: (paziente, medico) references Prestazione(paziente, medico)
vincolo foreign key: (specializzazione) references Specializzazione(nome)

Ricovero(paziente: cf_t, medico: cf_t, inizio: datetime, dimesso*: datetime,
npl: pl_t, ppl: integer, spl: int_pos, stpl: int_pos)
vincolo foreign key: (paziente, medico) references Prestazione(paziente, medico)
vincolo foreign key: (npl, ppl, spl, stpl) references
PostoLetto(numero, piano, settore, stanza)
vincolo ennupla: (inizio $\leq$ dimesso) or (dimesso = NULL)

Progettazione dei vincoli esterni

```
Trigger per V.Paziente.ricoveri:
    - Operazioni:
        - inserimento in Prestazione
    - Istante di invocazione: prima dell'operazione
    - Funzione:
        1. sia new l'ennupla che si sta inserendo
        2. error := select exists(select 1 from Ricovero where paziente = new.paziente and dimesso = NULL)
        3. se error = TRUE blocca l'operazione
        4. altrimenti consenti l'operazione

Trigger per V.Prestazione.disg:
    - Operazioni:
        - inserimento in Ricovero
        - inserimento in PrestazioneEst
    - Istante di invocazione: prima dell'operazione
    - Funzione:
        1. se l'operazione inserisce in Ricovero
        1.1. sia new l'ennupla inserita
        1.2. error := select exist(select 1 from PrestazioneEst where (paziente, medico, inizio) = (new.paziente, new.medico, new.inizio));
        1.3. se error = TRUE blocca l'operazione
        1.4. altrimenti consenti l'operazione
        2. se l'operazione inserisce in Ricovero
        2.1. sia new l'ennupla inserita
        2.2. error := select exist(select 1 from Ricovero where (paziente, medico, inizio) = (new.paziente, new.medico, new.inizio));
        2.3. se error = TRUE blocca l'operazione
        2.4. altrimenti consenti l'operazione
```

Specifiche realizzative degli use-case

```
Use-case Registrazione_Ricoveri_e_Prestazioni
Registra_Paziente(cf: cf_t, nome: stringM, cognome: stringM, nascita: date, telefono: numtel, email: email_t, casella: indirizzo) : cf_t
    algoritmo:
         1    exists <- risultato della query SQL ottenuta sostituendo a :cf il valore dell'omonimo parametro attuale
                 select exist (select 1 from Paziente where cf = :cf);
         2    if exists rappresenta un errore then
         3        inoltra l'errore;
         4    else if exists = TRUE then
         5        inoltra l'errore 'paziente già esistente';
         6    else
         7        res <- risultato della query SQL ottenuta sostituendo a :cf, :nome, :cognome, :nascita
                     i valori degli omonimi parametri attuali
                     insert ignore into Persona values (:cf, :nome, :cognome, :nascita);
         8    if res rappresenta un errore then
         9        inoltra l'errore;
        10    else
        11        res <- risultato della query SQL ottenuta sostituendo a :cf, :telefono, :email, :casella
                     i valori degli omonimi parametri attuali
                     insert into Paziente values (:cf, :telefono, :email, :casella);
        12        if res rappresenta un errore then
        13            inoltra l'errore;
        14        else return :cf;

Registra_Ricovero(p: cf_t, m: cf_t, pl: pl_t, piano: integer, set: int_pos, st: int_pos) : (cf_t, cf_t, datetime)
    algoritmo:
         1    ric <- risultato della query SQL ottenuta sostituendo a :p il valore dell'omonimo parametro attuale
                  select exists(select 1 from Ricovero where paziente = :p and dimesso = NULL);
         2    if ric rappresenta un errore then
         3        inoltra l'errore;
         4    else if ric = TRUE then
         5        inoltra l'errore 'paziente già ricoverato';
         6    else
         7        time <- CURRENT_TIMESTAMP
         8        res <- risultato della transazione SQL ottenuta sostituendo a :p, :m, :pl, :piano, :set, :st
                     i valori degli omonimi parametri attuali
                     begin transaction;
                     insert into Prestazione values (:p, :m, time);
                     insert into Ricovero values(:p, :m, time, :pl, :piano, :set, :st);
                     commit work;
         9        if res rappresenta un errore then
        10            inoltra l'errore;
        11        else return (:p, :m, time);

Dimetti(p: cf_t, m: cf_t, i: datetime) : (cf_t, cf_t, datetime)
    algoritmo:
         1    ric <- risultato della query SQL ottenuta sostituendo a :p, :m, :i i valori degli omonimi parametri attuali
                  select exists(select 1 from Ricovero where (paziente, medico, inizio, dimesso) = (:p, :m, :i, NULL);
         2    if ric rappresenta un errore then
         3        inoltra l'errore;
         4    else if ric = FALSE then
         5        inoltra l'errore 'ricovero non trovato';
         6    else
         7        time <- CURRENT_TIMESTAMP
         8        u <- risultato della query SQL ottenuta sostituendo a :p, :m, :i i valori degli omonimi parametri attuali
                      update Ricovero set dimesso = time where (paziente, medico, inizio) = (:p, :m, :i);
         9    if u rappresenta un errore then
        10        inoltra l'errore;
        11    else return (:p, :m, :i);

Registra_Prestazione(p: cf_t, m: cf_t, desc: stringL, spec: stringL) : (cf_t, cf_t, datetime)
    algoritmo:
         1    ric <- risultato della query SQL ottenuta sostituendo a :p il valore dell'omonimo parametro attuale
                  select exists(select 1 from Ricovero where paziente = :p and dimesso = NULL);
         2    if ric rappresenta un errore then
         3        inoltra l'errore;
         4    else if ric = TRUE then
         5        inoltra l'errore 'paziente ricoverato: impossibile erogare prestazioni';
         6    else
         7        time <- CURRENT_TIMESTAMP
         8        i <- risultato della transazione SQL ottenuta sostituendo a :p, :m, :desc, :spec
                      i valori degli omonimi parametri attuali
                      begin transaction;
                      insert into Prestazione values (:p, :m, time);
                      insert into PrestazioneEst values(:p, :m, time, :desc, :spec);
                      commit work;
         9        if i rappresenta un errore then
        10            inoltra l'errore;
        11        else return (:p, :m, time);
```

```
Ricerca_Medici(spec: stringL) : cf_t(0,N)
    algoritmo:
         1    S1 <- risultato della query SQL ottenuta sostituendo a :spec il valore dell'omonimo parametro attuale
                  select cf from Medico where specPrim = :spec;
         2    if S1 rappresenta un errore then
         3        inoltra l'errore;
         4    else if S1 ≠ NULL then
         5        return S1;
         6    else
         7        S2 <- risultato della query SQL ottenuta sostituendo a :spec il valore dell'omonimo parametro attuale
                    select m.cf from Medico m, SpecSec s where s.medico = m.cf and s.spec = :spec;
         8        if S1 rappresenta un errore then
         9            inoltra l'errore;
        10        else return S2;

Pianifica_Itinerario(m: cf) : (integer, int_pos)(0,N)
    algoritmo:
         1    isM <- risultato della query SQL ottenuta sostituendo a :m il valore dell'omonimo parametro attuale
                  select exist(select 1 from Medico where cf = :m);
         2    if isM rappresenta un errore then
         3        inoltra l'errore;
         4    else if isM = FALSE then
         5        inoltra l'errore ':m non è un medico';
         6    else
         7        it <- risultato della query SQL ottenuta sostituendo a :m il valore dell'omonimo parametro attuale
                      select ppl as p, stpl as s from Ricovero where medico = :m
                      order by p, s asc;
         8        if it rappresenta un errore then
         9            inoltra l'errore;
        10        else return it;
```