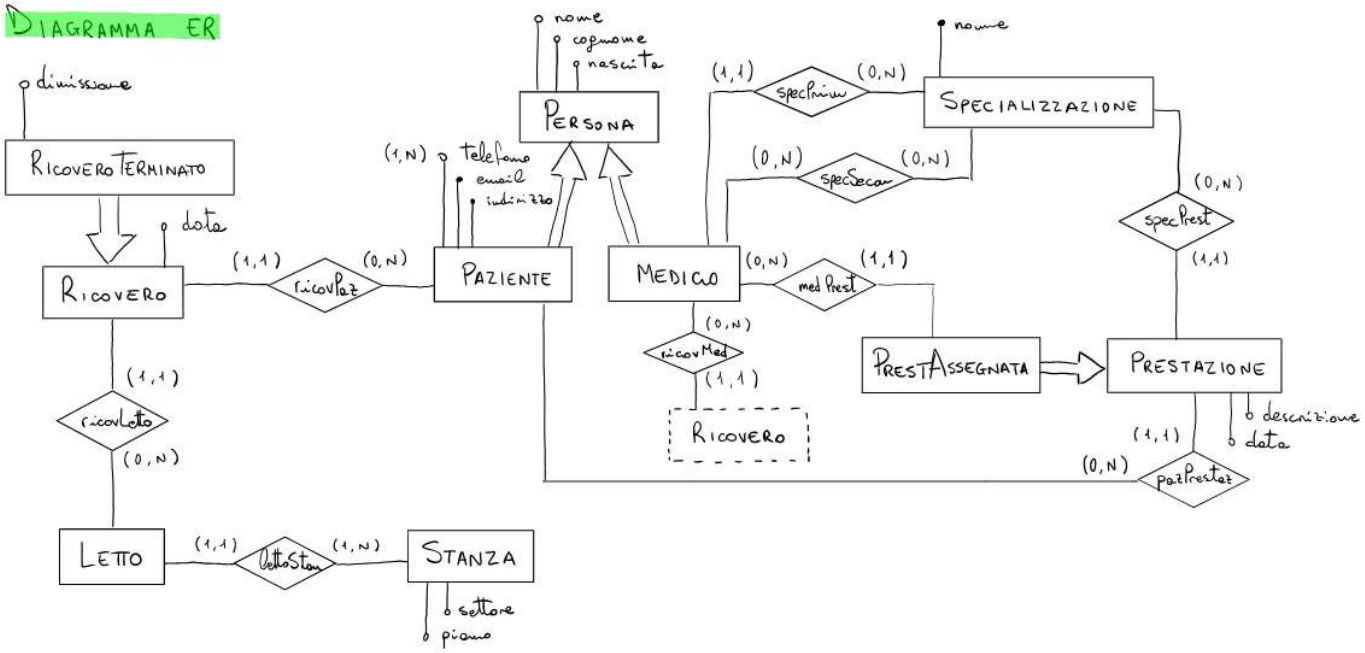
P. QuickHospital

mercoledì 3 giugno 2020 18:23

DIAGRAMMA ER



SPECIFICHE DEI DATI

Dominio Indirizzo

record composto dai seguenti campi:
- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa numerica di 5 cifre

Dominio Telefono:

record composto dai seguenti campi:
- prefisso: stringa numerica di 2 cifre
- numero: stringa numerica di 10 cifre

Entità Persona

nome	stringa
cognome	stringa
nascita	data

Entità Paziente

telefono	(1,N)	Telefono
email		stringa
indirizzo		Indirizzo

Entità Ricovero

data	data

Entità RicoveroTerminato

dimissione	data

Entità Stanza

piano	intero > 0
settore	intero > 0

Entità Specializzazione

nome	stringa

Entità Prestazione

data	data
descrizione	stringa

[V. RicoveroTerminato. dimissione. dopoDataRicovero]

$\forall rt, d, dd \quad RicoveroTerminato(rt) \wedge data(rt, d) \wedge dimissione(rt, dd) \longrightarrow d \leq dd$

[V. Ricovero. periodoStessoLetto]

$\forall r, d, l \quad Ricovero(r) \wedge data(r, d) \wedge ricovLetto(r, l) \longrightarrow$
$((\exists dd \quad RicoveroTerminato(r) \wedge dimissione(r, dd) \longrightarrow \neg \exists r2, d2 \quad Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge$
$\wedge d2 \leq dd \wedge ricovLetto(r2, l) \wedge r \neq r2) \vee (\neg \exists r2, d2 \quad Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge ricovLetto(r2, l) \wedge r2 \neq r))$

[V. Ricovero. periodoStessoPaziente]

$\forall r, d, p \quad Ricovero(r) \wedge data(r, d) \wedge ricovPaz(r, p) \longrightarrow$
$((\exists dd \quad RicoveroTerminato(r) \wedge dimissione(r, dd) \longrightarrow \neg \exists r2, d2 \quad Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge$
$\wedge d2 \leq dd \wedge ricovPaz(r2, p) \wedge r \neq r2) \vee (\neg \exists r2, d2 \quad Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge ricovPaz(r2, p) \wedge r2 \neq r))$

[V. Persona . nascita . primaRicovero]

$\forall e, pe, d \left(Ricovero(e) \wedge (ricovPat(e, pe) \vee ricovMed(e, pe)) \wedge nascita(pe, d) \right) \longrightarrow \neg \exists_{de}\ data(e, de) \wedge de < d$

[V. Persona . nascita . primaPrestazione]

$\forall e, pe, d \left(Prestazione(e) \wedge (patPrestat(e, pe) \vee medPrest(e, pe)) \wedge nascita(pe, d) \right) \longrightarrow \neg \exists_{de}\ data(e, de) \wedge de < d$

[V. Persona . medicoPaziente . almenoUno]

$\forall p \quad Persona(p) \longrightarrow Medico(p) \vee Persona(p)$

[V. Stanza . maxLetto]

$\forall s \quad Stanza(s) \longrightarrow \left| \{ l \mid lettoStanza(s, l) \} \right| \leq 8$

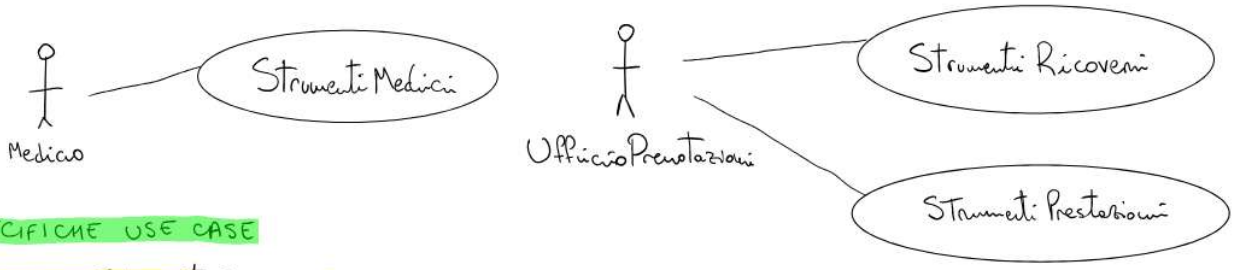
[V. Medico . spec . disp]

$\forall m \quad Medico(m) \longrightarrow \left(\exists s \ specPrinc(m, s) \longrightarrow \neg specSecon(m, s) \right)$

[V. Paziente . ricovPrestaz. almenoUno]

$\forall p \quad Paziente(p) \longrightarrow \exists e \quad ricovPat(p, e) \vee patPrestat(p, e)$

DIAGRAMMA UML USE-CASE



SPECIFICHE USE CASE

Use-Case StrumentiRicoveri

- postiDisponibili (d : data) : intero ≥ 0
  precondizioni : nessuna
  postcondizioni :
  Modifica livello estensionale dei dati : nessuno
  Valore di ritorno :

$$L_{tot} = \{ l \mid \exists s \ Stanza(s) \wedge lettoStan(l, s) \}$$

$$L_{occ} = \{ l \mid \exists r, dr \left(Ricovero(r) \wedge \neg RicoveroTerminato(r) \wedge data(r, dr) \wedge d > dr \right) \vee \left(RicoveroTerminato(r) \wedge data(r, dr) \wedge d < dr \right) \}$$

$$result = |L_{tot}| - |L_{occ}|$$

- registraRicovero (p : PazInterno , l : Letto , m : Medico) : Ricovero
  precondizioni : postiDisponibili(ADESSO) $\wedge \neg \exists r$ ricovLetto(r, l)
  postcondizioni :
  Modifica livello estensionale dei dati :
  Variazioni dominio di interpretazione : nuovo elemento a

Modifica livello estensionale dei dati:
Variazioni dominio di interpretazione: nuovo elemento α
Variazioni estensione di predicato:
- Ricovero(α)
- data(α, ADESSO)
- ricovPaz(α, p)
- ricovLetto(α, l)
- ricovMed(α, m)

Valore di ritorno: result = α

- dimissioneRicovero(r : Ricovero) : RicoveroTerminato

precondizioni: nessuna
postcondizioni:
Modifica dominio di interpretazione:
Variazioni dominio di interpretazione: nessuna
Variazioni estensione di predicato:
- RicoveroTerminato(r)
- dimissione(r, ADESSO)

Valore di ritorno: result = r

Use-Case StrumentoPrestazioni

- mediciIdonei(p : Prestazione) : Medico(0,N)

precondizioni: $\neg$ PrestAssegnata(p)
postcondizioni:
Modifica livello estensionale dei dati: nessuna
Valore di ritorno:

$$Primi = \{ m \mid \exists sp \ specPrest(p, sp) \wedge specPrim(m, sp) \}$$

$$Sec = \{ m \mid \exists sp \ specPrest(p, sp) \wedge specSecon(m, sp) \}$$

$$result = \begin{cases} Primi & \text{se } |Primi| > 0 \\ Sec & \text{altrimenti} \end{cases}$$

Use-Case StrumentoMedici

- itineranioVisite(m : Medico)

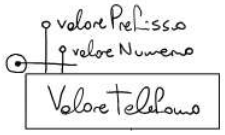
precondizioni: nessuna
postcondizioni:
Modifica dominio di interpretazione: nessuna
Valore di ritorno:

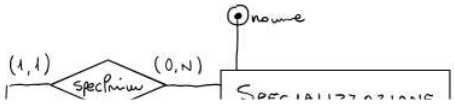
$$V = \{ (pi, se) \mid \exists r, l, s \ ricovMed(m, r) \wedge ricovLetto(r, l) \wedge lettoStan(l, s) \wedge piano(s, pi) \wedge settore(s, se) \wedge \neg RicoveroTerminato(r) \}$$

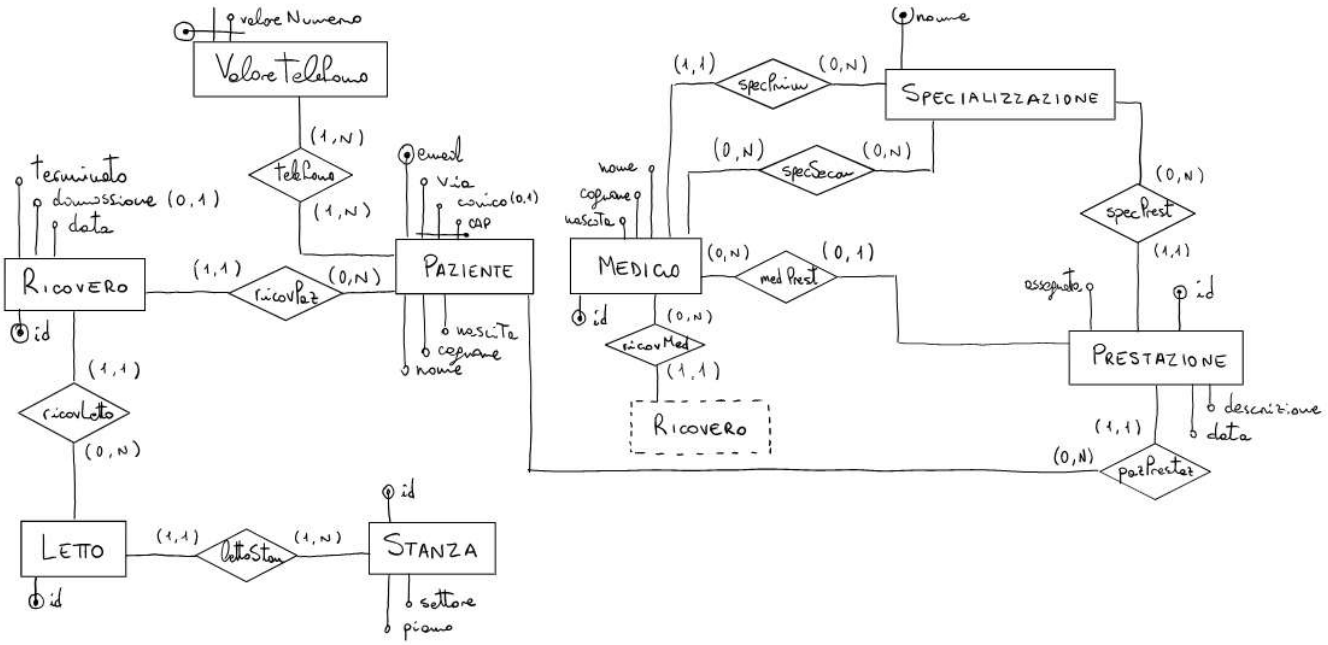
result = V

NOTA: l'insieme V dovrà, in fase di progettazione, essere ordinato in base al piano, e a parità di piano in base al settore.

RISTRUTTURAZIONE ER







NUOVI VINCOLI

$$\forall r\ (Ricovero(r) \wedge Terminato(r, true)) \longleftrightarrow \exists d\ dimissione(r, d)$$

$$\forall pr\ (Prestazione(pr) \wedge assegnata(pr, true)) \longleftrightarrow \exists m\ medPrest(m, pr)$$

VINCOLI MODIFICATI

[V. RicoveroTerminato. dimissione. dopoDataRicovero]

$$\forall rt, d, dd\ \ Ricovero(rt) \wedge Terminato(rt, true) \wedge data(rt, d) \wedge dimissione(rt, dd) \longrightarrow d \leq dd$$

[V. Ricovero. periodoStessoLetto]

$$\forall r, d, l\ \ Ricovero(r) \wedge data(r, d) \wedge ricovLetto(r, l) \longrightarrow$$
$$((\exists dd\ \ terminato(r, true) \wedge dimissione(r, dd) \longrightarrow \neg \exists r2, d2\ \ Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge d2 \leq dd \wedge$$
$$\wedge ricovLetto(r2, l) \wedge r \neq r2) \vee (\neg \exists r2, d2\ \ Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge ricovLetto(r2, l) \wedge r2 \neq r))$$

[V. Ricovero. periodoStessoPaziente]

$$\forall r, d, p\ \ Ricovero(r) \wedge data(r, d) \wedge ricovPaz(r, p) \longrightarrow$$
$$((\exists dd\ \ Terminato(r, true) \wedge dimissione(r, dd) \longrightarrow \neg \exists r2, d2\ \ Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge d2 < dd \wedge$$
$$\wedge ricovPaz(r2, p) \wedge r \neq r2) \vee (\neg \exists r2, d2\ \ Ricovero(r2) \wedge data(r2, d2) \wedge d2 \geq d \wedge ricovPaz(r2, p) \wedge r2 \neq r))$$

RISTRUTTURAZIONE SPECIFICHE DATI

Dominio StringS

```
create domain StringS as varchar(100);
```

Dominio StringM

```
create domain StringM as varchar(200);
```

Dominio IntegerGZ

```
create domain IntegerGZ as integer
  check (value > 0);
```

Dominio PrefissoTelefono

```
create domain PrefissoTelefono as varchar(5)
  check (value ~ '^\+[0-9]+$');
```

Dominio NumeroTelefono

```
create domain NumeroTelefono as char(10)
  check (value ~ '^[0,9]{0,6}$');
```

Dominio CAP

```
create domain CAP as char(5)
  check (value ~ '^[0,9]{5,5}$');
```

Entità Medico

id	integer
nome	StringS
cognome	StringS
nascita	date

Entità Paziente

nome	StringS
cognome	StringS
nascita	date
email	StringS
via	StringM
civico	IntegerGZ
CAP	CAP

Entità Ricovero

id	integer
data	date
terminato	boolean
dimissione	date

Entità Letto

id	integer

Entità Stanza

id	integer
piano	IntegerGZ
settore	IntegerGZ

Entità Specializzazione

nome	StringS

Entità Prestazione

id	integer
data	data
descrizione	stringa
assegnata	boolean

Entità ValoreTelefono

valorePrefisso	PrefissoTelefono
valoreNumero	NumeroTelefono

SCHEMA RELAZIONALE

Stanza (id : serial , piano : IntegerGZ , settore : IntegerGZ)
inclusione : id ⊆ Letto(stanza)

Letto (id : serial , stanza : integer)
foreign key : stanza references Stanza(id)

Ricovero (id : serial , data : date , terminato : boolean , dimissione* : date ,
letto : integer , paziente : StringS , medico : integer)
foreign key : letto references Letto(id)
foreign key : paziente references Paziente(email)
foreign key : medico references Medico(id)
ennupla : terminato = true ⟷ dimissione ≠ NULL
ennupla : dimissione ≠ NULL ⟶ dimissione ≥ data

Paziente (email : StringS , nome : StringS , cognome : StringS , nascita : date ,
via : StringM , civico* : IntegerGZ , CAP : CAP)
inclusione : email ⊆ Telefono(paziente)
chiave : (via , civico, CAP)

Telefono (valorePrefisso : PrefissoTelefono , valoreNumero : NumeroTelefono , paziente : StringS)
foreign key : paziente references Paziente(email)

Prestazione (id : serial , descrizione : StringM , data : date , assegnata : boolean ,
medico* : integer , paziente : StringS , specializzazione : StringS)
foreign key : medico references Medico(id)
foreign key : paziente references Paziente(email)
foreign key : specializzazione references Specializzazione(nome)
ennupla : medico ≠ NULL ⟷ assegnata = true

Medico (id : serial , nome : StringS , cognome : StringS , nascita : date , specPrim : StringS)
foreign key : specPrim references Specializzazione(nome)

Specializzazione (nome : StringS)

Specializzazione (nome : String)

specSecon (medico : integer , specializzazione : String)
  foreign key : medico references Medico (id)
  foreign key : specializzazione references Specializzazione (nome)

PROGETTAZIONE VINCOLI ESTERNI

Vincolo V.Stanza.maxLetti

Trigger :
  Operazioni intercettate : inserimento esempio relazione Letto
  Istante di invocazione : prima operazione intercettata
  Funzione :
    new = esempio da inserire
    Q =
      select count(*) < 8 as lettoAggiungibile
      from Stanza s
      where s.id = new.stanza
    IF il valore della colonna lettoAggiungibile di Q == true
      permetti l'operazione
    ELSE blocca l'operazione

Vincolo V.Medico.spec.Dsp

Trigger :
  Operazioni intercettate : inserimento o modifica esempio relazione Medico
  Istante di invocazione : prima operazione intercettata
  Funzione :
    new = esempio da inserire o risultato modifica
    hasError =
      exists (
        select *
        from specSecon ss
        where ss.medico = new.id
          and ss.specializzazione = new.specPrim
)
    IF hasError THEN blocca operazione
    ELSE permetti operazione

Trigger :
  Operazioni intercettate : inserimento o modifica esempio relazione specSecon
  Istante di invocazione : prima operazione intercettata
  Funzione :
    new = esempio da inserire o risultato modifica
    hasError =
      exists (
        select *
        from Medico m
        where m.id = new.medico
          and m.specPrim = new.specializzazione
)
    IF hasError THEN blocca operazione
    ELSE permetti operazione

IF hasError THEN blocca operazione
ELSE permetti operazione

Vincolo V. Ricovero. periodo StessoLetto

Trigger:
Operazioni intercettate: inserimento o modifica ennupla relazione Ricovero
Istante invocazione: prima operazione intercettata
Funzione:
new = ennupla da inserire o risultato modifica
hasError = exists (
select *
from Ricovero r
where new.id <> r.id
and new.letto = r.letto
and (new.data, new.dimissione) overlaps (r.data, r.dimissione));
IF hasError THEN blocca operazione
ELSE permetti operazione

Vincolo V. Ricovero. periodo StessoParente

Trigger:
Operazioni intercettate: inserimento o modifica ennupla relazione Ricovero
Istante invocazione: prima operazione intercettata
Funzione:
new = ennupla da inserire o risultato modifica
hasError = exists (
select *
from Ricovero r
where new.id <> r.id
and new.parente = r.parente
and (new.data, new.dimissione) overlaps (r.data, r.dimissione));
IF hasError THEN blocca operazione
ELSE permetti operazione

Vincolo V. Persona. nascita. primaRicovero

Trigger:
Operazioni intercettate: inserimento o modifica ennupla relazione Ricovero
Istante invocazione: prima operazione intercettata
Funzione:
new = ennupla da inserire o risultato modifica
hasError = exists (
select *
from Medico m, Parente p
where new.parente = p.email
and new.data < p.nascita
or new.medico = m.id
and new.data < m.nascita);
IF hasError THEN blocca operazione
ELSE permetti operazione

Trigger:
operazioni intercettate: modifica ennupla relazione Medico

Trigger:
operazioni intercettate: modifica entità relazione Medico
Istante invocazione: prima operazione intercettata
Funzione:
new = entità risultante modifica
hasError = exists(
select *
from Ricovero r
where r.medico = new.id
and r.data < new.nascita);
IF hasError THEN blocca operazione
ELSE permetti operazione

Trigger:
Operazioni intercettate: modifica entità relazione Paziente
Istante invocazione: prima operazione intercettata
Funzione:
new = entità risultante modifica
hasError = exists(
select *
from Ricovero r
where r.paziente = new.email
and r.data < new.nascita);
IF hasError THEN blocca operazione
ELSE permetti operazione

Vincolo V.Persona.nascita.primaPrestazione

Trigger analoghi a quello in V.Persona.nascita.primaRicovero sostituendo la relazione Ricovero con la relazione Prestazione

Vincolo V.Paziente.nRicovPrestaz.almenoUno

Trigger:
Operazioni intercettate: inserimento entità relazione Paziente
Istante invocazione: dopo operazione intercettata
Funzione:
new = entità da inserire
hasError = not exists(
select *
from Ricovero r, Prestazione p
where r.paziente = new.email
or p.paziente = new.email);
IF hasError THEN blocca operazione
ELSE permetti operazione

SPECIFICHE REALIZZATIVE USE-CASE

Use-case StrumentiPrestazioni

- mediciIdonei (p : integer) : Insieme (< id : integer, nome : StringS, cognome : StringS, nascita : date, specPrim : StringS >)

- mediciIdonei (p : integer) : Insieme (< id : integer, nome : StringS, cognome : StringS, nascita : date, specPrim : StringS >)
  res = risultato del comando SQL seguente sostituendo a ':p' il valore dell anonimo param. attuale
  ```
  select m.id, m.nome, m.cognome, m.nascita, m.specPrim
  from Medico m, Prestazione pr
  where :p = pr.id and pr.specializzazione = m.specPrim
  ```
  IF res == NULL THEN
  res = risultato del comando SQL seguente sostituendo a ':p' il valore del param. attuale p
  ```
  select m.id, m.nome, m.cognome, m.nascita, m.specPrim
  from Medico m, Prestazione pr, specSecon ss
  where :p = pr.id
    and pr.specializzazione = ss.specializzazione
    and ss.medico = m.id
  ```
  IF res == NULL THEN genera l'errore 'non esistono medici con questa specializzazione'
  ELSE restituisci res

Use-case StrumentiMedici

- iTuoiRicoveriVisite (m : integer) : Collezione (< piano : IntegerGZ, settore : IntegerGZ >)
  res = risultato del comando SQL seguente sostituendo a ':m' il valore del parametro attuale m
  ```
  select s.piano, s.settore
  from Stanza s, Ricovero r, Letto l
  where :m = r.medico
    and r.Terminato = false
    and r.letto = l.id
    and l.stanza = s.id
  order by s.piano ASC, s.settore ASC;
  ```
  IF res == NULL THEN genera l'errore 'nessuna visita da mostrare'
  ELSE restituisci res

Use-case StrumentiRicoveri

- registraRicovero (p: integer, l : integer, m : integer) : integer
  IF DB.postiDisponibili (CURRENT_DATE) restituisce un errore
  THEN inoltra l'errore
  ELSE IF DB.postiDisponibili (CURRENT_DATE) < 1
  THEN termina con l'errore 'posti disponibili non sufficienti'
  res = risultato del comando SQL seguente sostituendo a ':p', ':l', ':m' i valori dei parametri
  ```
  insert into Ricovero (data, terminato, letto, paziente, medico)
  values (CURRENT_DATE, false, :l, :p, :m)
  returning id
  ```
  IF res rappresenta un errore THEN inoltra l'errore
  ELSE restituisci il valore della colonna 'id' dell'unica ennupla di res

- dimissioneRicovero (r : integer)
  ```
  update Ricovero set terminato = True, dimissione = CURRENT_DATE
  where id = r;
  ```

Si definiscono le seguenti funzioni nel DBMS:

- DB.postiDisponibili (d : date) : IntegerGEZ
  res = risultato della query SQL seguente sostituendo a ':d' il valore del parametro attuale d
  ```
  select x.postiTotali - y.postiOccupati as postiDisponibili
  ```

res = risultato della query SQL seguente sostituendo a ':d' il valore del parametro attuale d

```
select x.postiTotali - y.postiOccupato as postiDisponibili
from ( select count(*) as postiTotali
       from Letto ) x,
     ( select count(*) as postiOccupato
       from Ricovero r
       where (r.Terminato = false
         and r.data ≤ :d)
         or (r.Terminato = true
         and r.data ≤ :d
         and r.dimissione > :d )) y
```

IF res == NULL THEN genera l'errore 'non ci sono letti o ricoveri'
ELSE restituisci la colonna 'postiDisponibili' dell'unica ennupla di res