QuickHospital

Contents

1 Requisiti

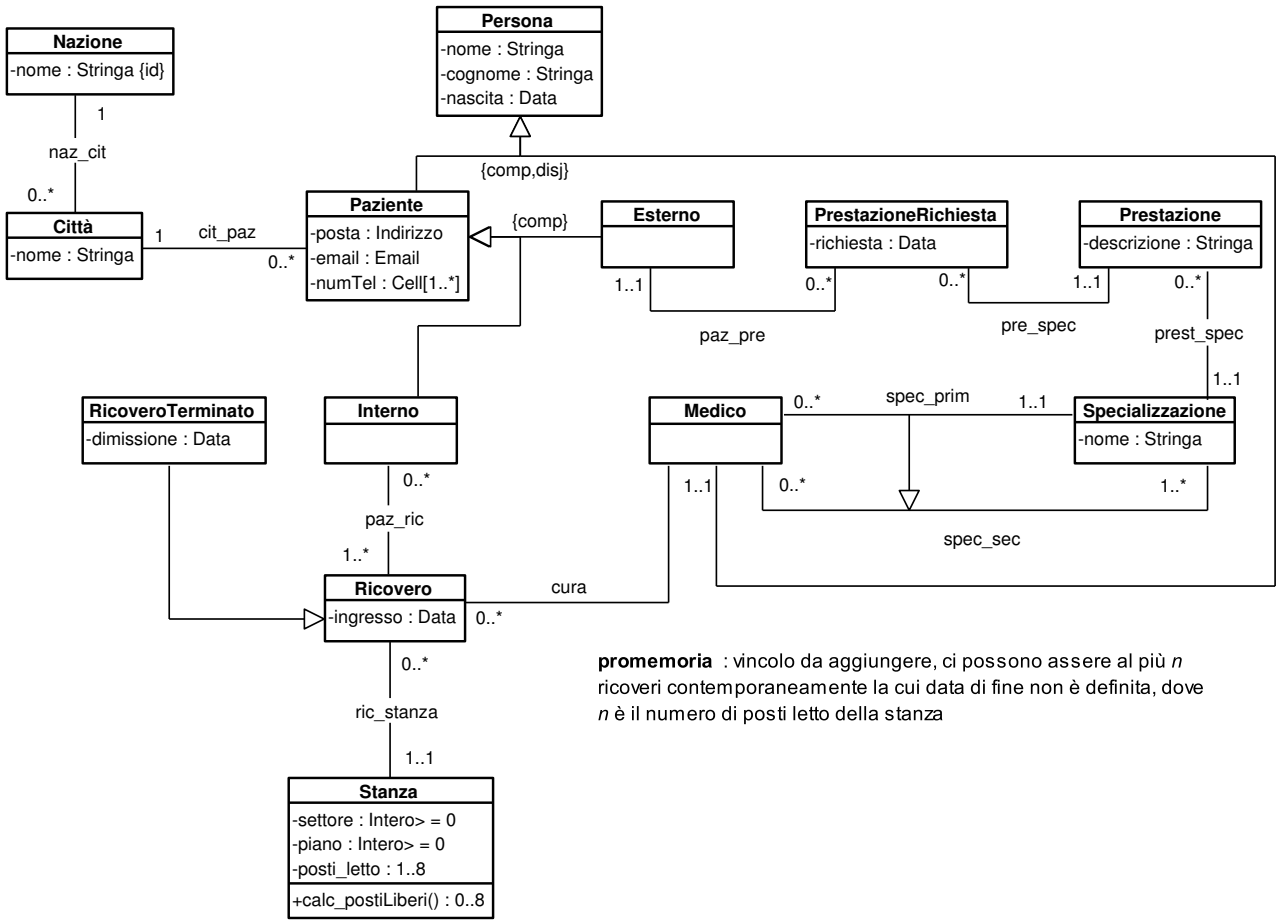
1. Paziente
    - 1.1 nome
    - 1.2 cognome
    - 1.3 data di nascita
    - 1.4 recapiti telefonici [1..*]
    - 1.5 email
    - 1.6 recapito postale
    - 1.7 interno o esterno?
        - 1.7.1 se esterno, prestazione medica richiesta (vedi REQ 4.)

2. Medico
    - 2.1 nome
    - 2.2 cognome
    - 2.3 data di nascita
    - 2.4 pazienti in cura
    - 2.5 specializzazione primaria
    - 2.6 specializzazione secondaria

3. Ricovero
    - 3.1 paziente coinvolto
    - 3.2 stanza del ricovero
        - 3.3 numero posti letto (da 1 a 8)
        - 3.4 piano
        - 3.5 settore
        - 3.6 data ricovero

4. Prestazione Medica
    - 4.1 Paziente esterno coinvolto
    - 4.2 data richiesta
    - 4.3 specializzazione medica richiesta
    - 4.4 descrizione estesa

2 Diagramma UML delle classi



promemoria : vincolo da aggiungere, ci possono assere al più *n* ricoveri contemporaneamente la cui data di fine non è definita, dove *n* è il numero di posti letto della stanza

3 Tipi di Dato

- $Cell = stringasecondoRegex :' +[0..9]2[0..9]10'$
- $Indirizzo = TipodiDatoEnum\{via : Stringa, n_civico : int\}$
- $Email = stringasecondoRegex :' [A..Za..z]@[a..z].[a..z]'$

4 Vincoli Esterni

[V.RicoveroTerminato.fineGinizio]

$\forall r_t, t_i, t_f \quad RicoveroTerminato(r_t) \wedge DataInizio(r_t, t_i) \wedge DataFine(r_t, t_f) \Rightarrow t_f \geq t_i$

[V.Stanza.num_link]
Sia $R_s := \{r | Ricovero(r) \wedge \neg RicoveroTerminato(r) \wedge ric\_st(r, this)\}$

$$|R_s| \leq 8$$

[V.Ricovero.nascita<ricovero]

$\forall d_n, d_r, r, p \quad dataNascita(d_n, p) \wedge DataInizio(d_r, r) \wedge Paziente(p) \wedge Ricovero(r) \Rightarrow d_n < d_r$

[V.Prestazione.nascita<prestazione]

$\forall d_n, d_p, pr, p \quad dataNascita(d_n, p) \wedge data(d_r, pr) \wedge Paziente(p) \wedge Prestazione(p) \Rightarrow d_n < d_p$

[V.Paziente.ric_ricterm]

$\nexists p, r_1, r_2 \quad Paziente(p) \wedge paz\_ric(p, r_1) \wedge paz\_ric(p, r_2) \wedge r_1 \neq r_2 \wedge \neg RicoveroTerminato(r_1) \wedge \neg Ricover$

5 Operazioni di classe

5.1 Operazioni di Stanza

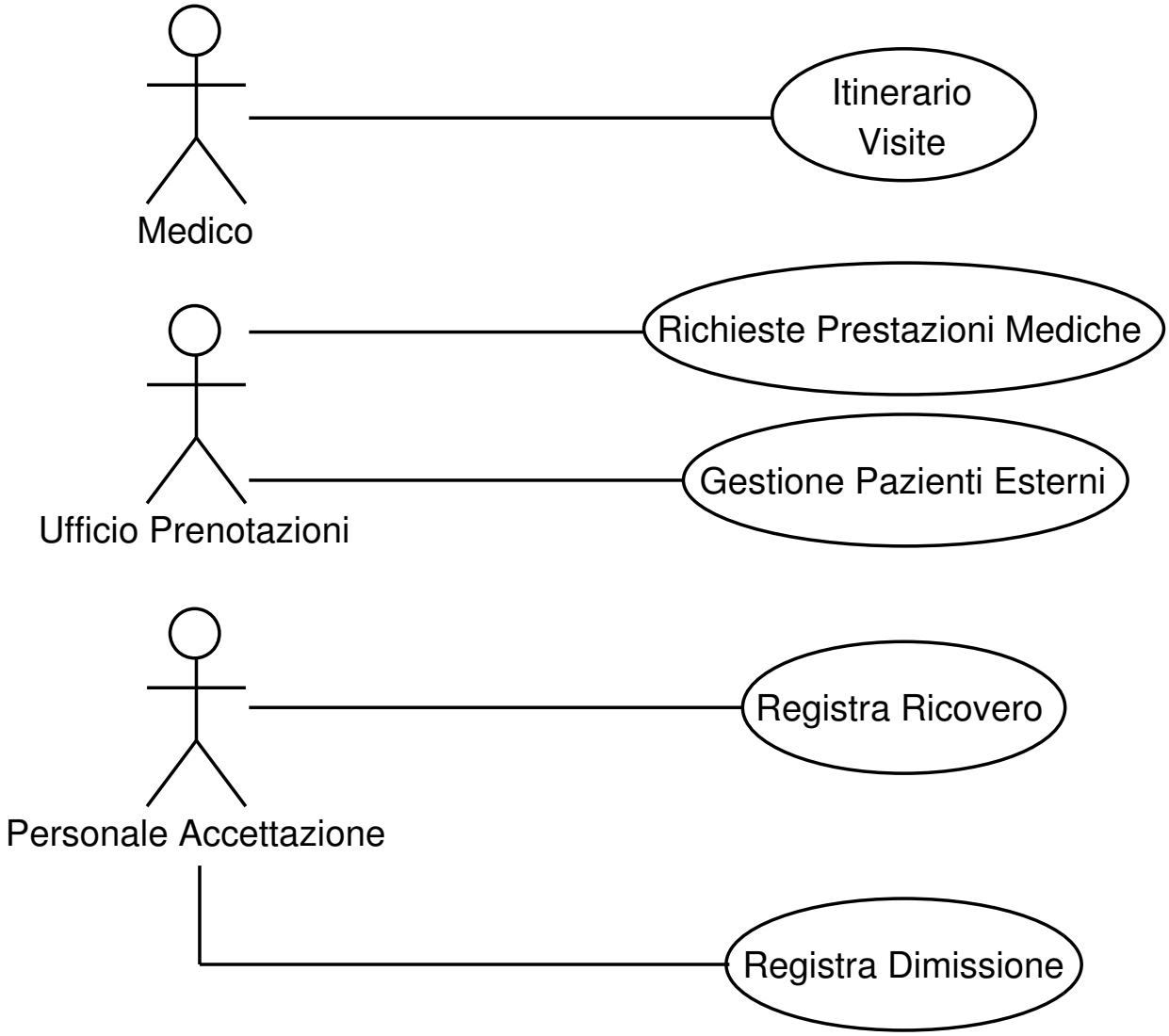
postiDisponibili() : Intero ≥ 0

- pre: Nessuna
- post: Non modifica il livello estensionale dei dati (nessuna nuova ennupla o modifiche a $\mathcal{D}$)
  $M_{in} = M_{out}$.
  Sia $this, p \quad postiLetto(this, p)$, $Stanza(this)$ e $R := \{r | ric\_st(r, this) \wedge Ricovero(r) \wedge \neg RicoveroTerminato(r)\}$.

$$result = p - |R|$$

6 Specifica degli Use-Case

6.1 Diagramma



6.2 Registra Anagrafica

registra_paziente(n : Stringa, c: Stringa, nascita: Date, tel : $\mathcal{T}$, email : Email , post: Indirizzo, est : Booleano): Paziente

- pre:
- post: Modifica il livello estensionale quindi $M_{in} \neq M_{out}$.
  Elementi di $\mathcal{D}$ aggiunti : ρ
  Elementi di $\mathcal{D}$ rimossi : nessuno
  Tuple aggiunte:
  - $est = true \Rightarrow PazienteEsterno(\rho)$, altrimenti $Paziente(\rho) \Rightarrow \neg PazienteEsterno(\rho)$
  - $nome(\rho, n)$
  - $cognome(\rho, c)$
  - $dataNascita(\rho, nascita)$
  - $recapitoEmail(\rho, email)$
  - $recapitoPostale(\rho, post)$
  - $\forall t \in tel \quad recapitoTelefonico(\rho, t)$

  Tuple rimosse : nessuna
  $result = \rho$

6.3 Registra richiesta di prestazione

registra_prestazione(ρ: PazienteEsterno, d : Stringa, s: Specializzazione, δ : Data) : Prestazione

- pre : Sia $D := \{[d_i, d_f] \quad | Data(d_f) \wedge Data(d_i) \wedge paz\_ric(p, r) \wedge RicoveroTerminato(r) \wedge DataFine(r, d_f) \wedge DataInizio(r, d_i)\}$
  $\delta \notin [d_i^j, d_f^j] \quad \forall d_i^j, d_f^j \in D$

- post : $M_{in} \neq M_{out}$
  Elementi di $\mathcal{D}$ aggiunti: p
  Elementi di $\mathcal{D}$ rimossi: nessuno
  Tuple aggiunte:
  - $descrizione(d, p)$
  - $data(\delta, p)$
  - $paz\_prest(\rho, p)$
  - $prest\_spec(p, s)$

  $result = p$

lista_medici(π: Prestazione): Medico[0..*]

- pre: nessuna

- post : $M_{in} = M_{out}$
  Sia $\mathcal{M} := \{m \quad | \exists s \quad Medico(m) \wedge Specializzazione(s) \wedge prest\_spec(\pi, s) \wedge primaria(s, m)\}$.
  Sia $\mathcal{M}_s := \{m \quad | \exists s \quad Medico(m) \wedge Specializzazione(s) \wedge prest\_spec(\pi, s) \wedge secondaria(s, m)\}$.
  $|\mathcal{M}| > 0 \Rightarrow result = \mathcal{M}$
  $\wedge$
  $|\mathcal{M}| = 0 \rightarrow result = \mathcal{M}_s$

6.4 Registra ricovero

registra_ricorvero(ρ: Paziente, μ : Medico, d : Data): Ricovero

- pre: Sia $S := \{(s,p)|Stanza(s) \wedge postiDisponibili(s,p) \wedge p > 0\}$, $\sigma = \sum_{(s,p)\in S} p$

$$\sigma > 0$$

- post: $M_{in} \neq M_{out}$
  Elementi di $\mathcal{D}$ aggiunti: ω
  Elementi di $\mathcal{D}$ rimossi: nessuno
  Tuple aggiunte:
  - DataInizio(ω,$adesso$)
  - $med\_ric(\mu, \omega)$
  - $paz\_ric(\rho, \omega)$
  - Sia $S := \{s|Stanza(s) \wedge postiDisponibili(s) > 0\}$, scelta arbitraria di $s \in S$, $ric\_st(\omega, s)$.

  $result = \omega$

registra_dimissione(r: Ricovero, δ: Data): RicoveroTerminato

- pre: Sia d_i, $DataInizio(r, d_i) \Rightarrow \delta > d_i \wedge \neg RicoveroTerminato(r)$
- post: $M_{in} \neq M_{out}$
  Elementi di $\mathcal{D}$ aggiunti: nessuno
  Elementi di $\mathcal{D}$ rimossi: nessuno
  Tuple aggiunte:
  - $RicoveroTerminato(r)$
  - $datFine(r, \delta)$

  $result = r$

6.5 Calcola itinerario

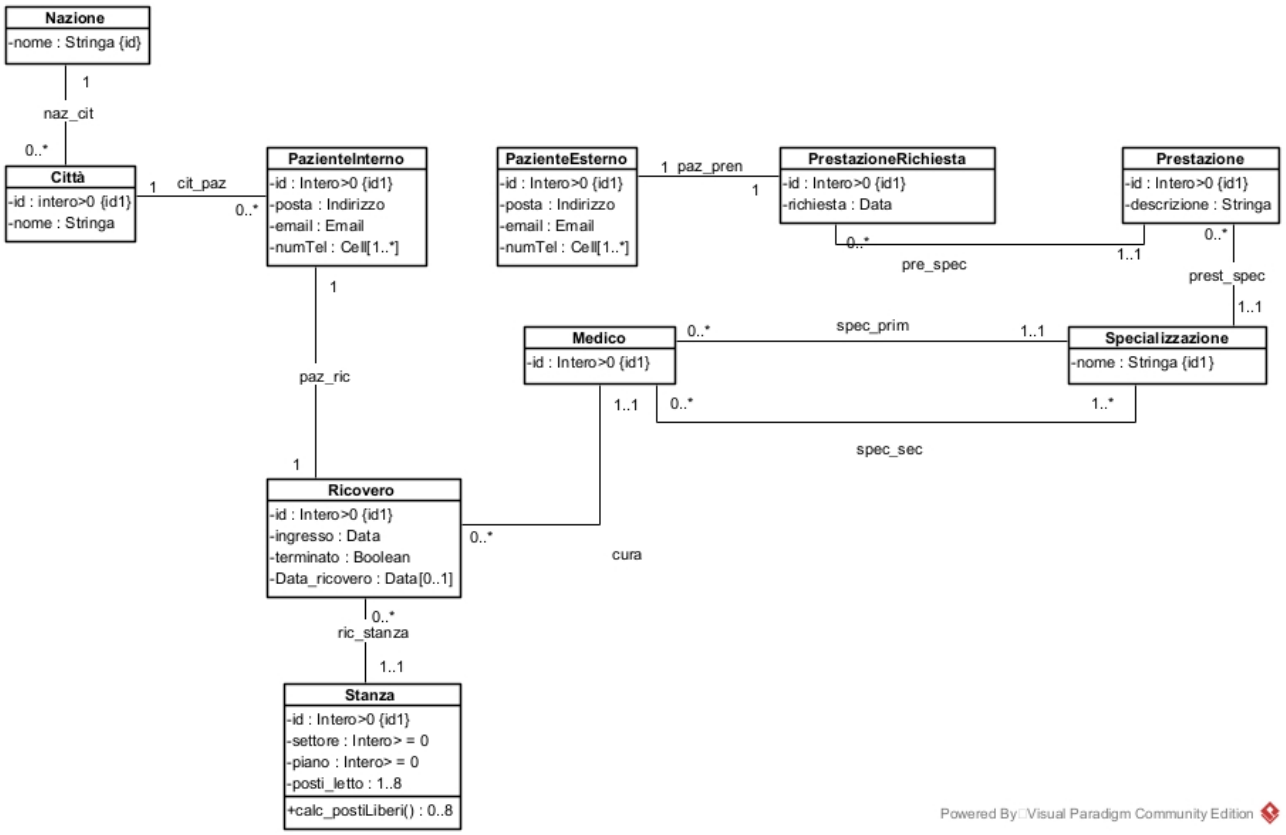
itinerario(m : Medico): $\mathcal{I}$

- pre : Sia m l'istanza di Medico che rappresenta l'attore che invoca l'operazione
- post: $M_{in} = M_{out}$
  Sia $R := \{r|Ricovero(r) \wedge \neg RicoveroTerminato(r) \wedge med\_ric(m, r)\}$
  Sia $S := \{s|\exists r \in R \wedge Stanza(s) \wedge ric\_st(r, s)\}$
  Sia $T := \{(p, \sigma)|\exists s \in S \wedge piano(s, p) \wedge settore(s, \sigma)\}$
  $result = \mathcal{I}_T$

Arrivati a questo punto la fase di **Analisi** è **finita**.

7 Ristrutturazione

7.1 Diagramma UML ristrutturato



Modifiche sulle generalizzazioni effettuate:

- Generalizzazione PazienteEsterno−PazienteInterno:
  E' stata preferita la divisione tra pazienti interni ed esterni poichè è più facile e veloce nella ricerca avere due tabelle separate.
  Se un paziente è sia interno che esterno, avrò 2 tuple uguali sulle due tabelle.

- Generalizzazione Ricovero−RicoveroTerminato:
  La generalizzazione per RicoveroTerminao è stata eliminata.

- Generalizzazione SpecializzazionePrimaria−SpecializzazioneSecondaria:
  La generalizzazione tra SpecializzazionePrimaria e SpecializzazioneSecondaria è stata eliminata e aggiunto un vincolo esterno.

7.2 Tipi e Domini

7.2.1 Tipi

- create type Indirizzo as enum { via: varchar(100), n_civico: Intero>0, }

7.2.2 Domini

- create domain Intero>0 as integer check(value>0 not NULL)
- create domain Telefono as integer secondo regex ('+[1..9]{2} [0..9]{10}')
- create domain Email as varchar secondo regex ('[a..zA..Z]@[a..z].[a..z]')

7.3 Vincoli Esterni

I precedenti vincoli esterni non violano la nuova ristrutturazione.
Nuovi vincoli esterni:

> [V.Medico.specializzazioni]
> Data Una SpecializzazionePrimaria essa non può essere SpecializzazioneSecondaria per lo stesso medico.

$$\forall m, sMedico(m) \land Specializzazione(s) \land SpecializzazionePrimaria(m, s)$$

$$\rightarrow \neg SpecializzazioneSecondaria(m, s)$$

7.4 Use Case

Gli use case non violano la nuova ristrutturazione.

7.5 Traduzione diretta del diagramma UML delle classi ristrutturato

Saranno scritte tutte le tabelle da creare.

- Nazione(**nome**: Stringa)
- Citta(**id**: nome, nazione: Nazione)
  - fk: nazione references Nazione(**nome**)
- PazienteInterno(**id**: intero¿0, posta: Indirizzo, email: Email, numTel, citta_residenza: Citta)
  - fk: citta_residenza references Citta(**id**)
- Ricovero(**id**: integer, ingresso: Date, terminato: Boolean: data_ricovero*, stanza: Stanza, specialista: Medico)
  - // UNISCO A STANZA PER AVERE UNA TABELLA CON LE INFORMAZIONI UNICHE DI UN RICOVERO
  - // UNISCO A MEDICO PER AVERE OGNI RICOVERO LE INFORMAZIONI DEL MEDICO
  - fk: stanza references Stanza(**id**)
  - fk: specialista references Medico(**id**)
- paz_ric(paz: Paziente, ric: Ricovero)
  - //LO TENGO PERCHè VOGLIO UNA TABELLA CON LA LISTA DEI PAZIENTI ATTUALMENTE RICOVERATI
  - foreign key: paz references Paziente(**id**)
  - foreign key: ric references Ricovero(**id**)
- Stanza(**id**: integer, settore: intero¿0, piano: intero¿=0 posti_letto: 1..8)
- PazienteEsterno(**id**: integer, posta: Indirizzo, email: Email, numTel, prenotazione: PrenotazioneRichiesta)
  - fk: prenotazione references PrenotazioneRichiesta(**id**)
  - // UNISCO PER AVERE UNA TABELLA CON I PAZIENTI ESTERNI DIRETTAMENTE COLLEGATA ALLA PRENOTAZIONE
- PrenotazioneRichiesta(**id**, richiesta: Date, prestazione: Prestazione)
  - fk: prestazione references Prestazione(**id**)
  - // UNISCO PER AVERE UNA TABELLA CON LE PRENOTAZIONI DIRETTAMENTE COLLEGATA ALLE PRESTAZIONI
- Prestazione(**id**: integer, descrizione: varchar(100), spec: Specializzazione)
  - fk: spec references Specializzazione(**nome**)
- Specializzazione(**nome**: varchar(100))

- Medico(**id**: integer, spec_prim: Specializzazione)
  - fk: spec_prim references Specializzazione(**nome**)
  - v.inclusione Medico(**id**) occorre in spec_sec(Medico)
- spec_sec(Medico: intero, Specializzazione: varchar(100))
  - foreign key: Medico references Medico(**id**)
  - foreign key: Specializzazione references Specializzazione(**nome**)
- // UNISCO MEDICO E SPECIALIZZAZIONE PRIMARIA, LASCIO UNA TABELLA SEPARATA PER LE SPECIALIZZAZIONI SECONDARIE

7.6 Trigger

I vincoli esterni da controllare con i trigger sono:

7.6.1 V.Medico.specializzazioni

```
Trigger per il vincolo V.Medico.specializzazioni:
    Operazioni: inserimento o modifica in Medico o in spec_sec
    Istante di invocazione: prima dell'operazione intercettata
    Funzione:
        Sia isError=FALSE;
        Sia new l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica
        Se si sta inserendo o modificando una ennupla in Medico:
            isError:= exists(select *
                             from Medico m, spec_sec s
                             where m.id=new.id
                             and m.spec_prim=s.Specializzazione and s.Medico=new.id)

        Se si sta inserendo o modificando una ennupla in spec_sec:
            isError:= exists(select *
                             from Medico m, spec_sec s
                             where m.id=new.Medico
                             and m.spec_prim=s.Specializzazione and s.Medico=new.Medico)

        Se isError = TRUE blocca l'operazione;
        Altrimenti permetti l'operazione
```

7.6.2 V.Ricovero.fineGinizio

```
Trigger per il vincolo V.Ricovero.fineGinizio:
    Operazioni: inserimento o modifica in Ricovero
    Istante di invocazione: prima dell'operazione intercettata
    Funzione:
        Sia isError=FALSE;
        Sia new l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica
        isError:= exists(select *
                         from Ricovero r
                         where r.id=new.id
                         and r.igresso>new.data_ricovero)

        Se isError = TRUE blocca l'operazione;
        Altrimenti permetti l'operazione
```

7.6.3 V.Stanza.num_link

```
Trigger per il vincolo V.Stanza.num_link:
    Operazioni: inserimento o modifica in Stanza
    Istante di invocazione: prima dell'operazione intercettata
    Funzione:
        Sia isError=FALSE;
        Sia new l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica
        isError:= exists(select count(r)
                        from Stanza s, Ricovero r
                        where s.id=new.id and r.stanza= s.id and r.terminato=false)
        Se isError = TRUE blocca l'operazione;
        Altrimenti permetti l'operazione
```

7.6.4 V.Ricovero.nascita<ricovero

```
Trigger per il vincolo V.Ricovero.nascita$<$ricovero:
    Operazioni: inserimento o modifica in Ricovero e PazienteInterno
    Istante di invocazione: prima dell'operazione intercettata
    Funzione:
        Sia isError=FALSE;
        Sia new l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica
        isError:= exists(select *
                        from Ricovero r, Paziente p
                        where r.id=new.id and r.paziente=p.id and p.nascita<new.data_ric
        Se isError = TRUE blocca l'operazione;
        Altrimenti permetti l'operazione
```

7.6.5 V.PazienteEsterno.nascita<PrenotazioneRichiesta.richiesta

```
Trigger per il vincolo V.PazienteEsterno.nascita$<$PrenotazioneRichiesta.richiesta:
    Operazioni: inserimento o modifica in PazienteEsterno e PrenotazioneRichiesta
    Istante di invocazione: prima dell'operazione intercettata
    Funzione:
        Sia isError=FALSE;
        Sia new l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica
        isError:= exists(select *
                        from PazienteEsterno p, PrenotazioneRichiesta pr
                        where p.id=new.id and p.prenotazione=pr.id and p.nascita<pr.rich
        Se isError = TRUE blocca l'operazione;
        Altrimenti permetti l'operazione
```

7.6.6 V.Paziente.ric_ricterm

```
Trigger per il vincolo V.Paziente.ric_ricterm:
    Operazioni: inserimento o modifica in Paziente e Ricovero
    Istante di invocazione: prima dell'operazione intercettata
    Funzione:
        Sia isError=FALSE;
        Sia new l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica
        isError:= exists(select *
                        from Paziente p, Ricovero r1, Ricovero r2
                        where p.id=new.id and r1.paziente=p.id and r2.paziente=p.id
                        and r1.id!=r2.id and r1.terminato=false and r2.terminato=false)
        Se isError = TRUE blocca l'operazione;
        Altrimenti permetti l'operazione
```