QuickHospital

Contents

1 Requisiti

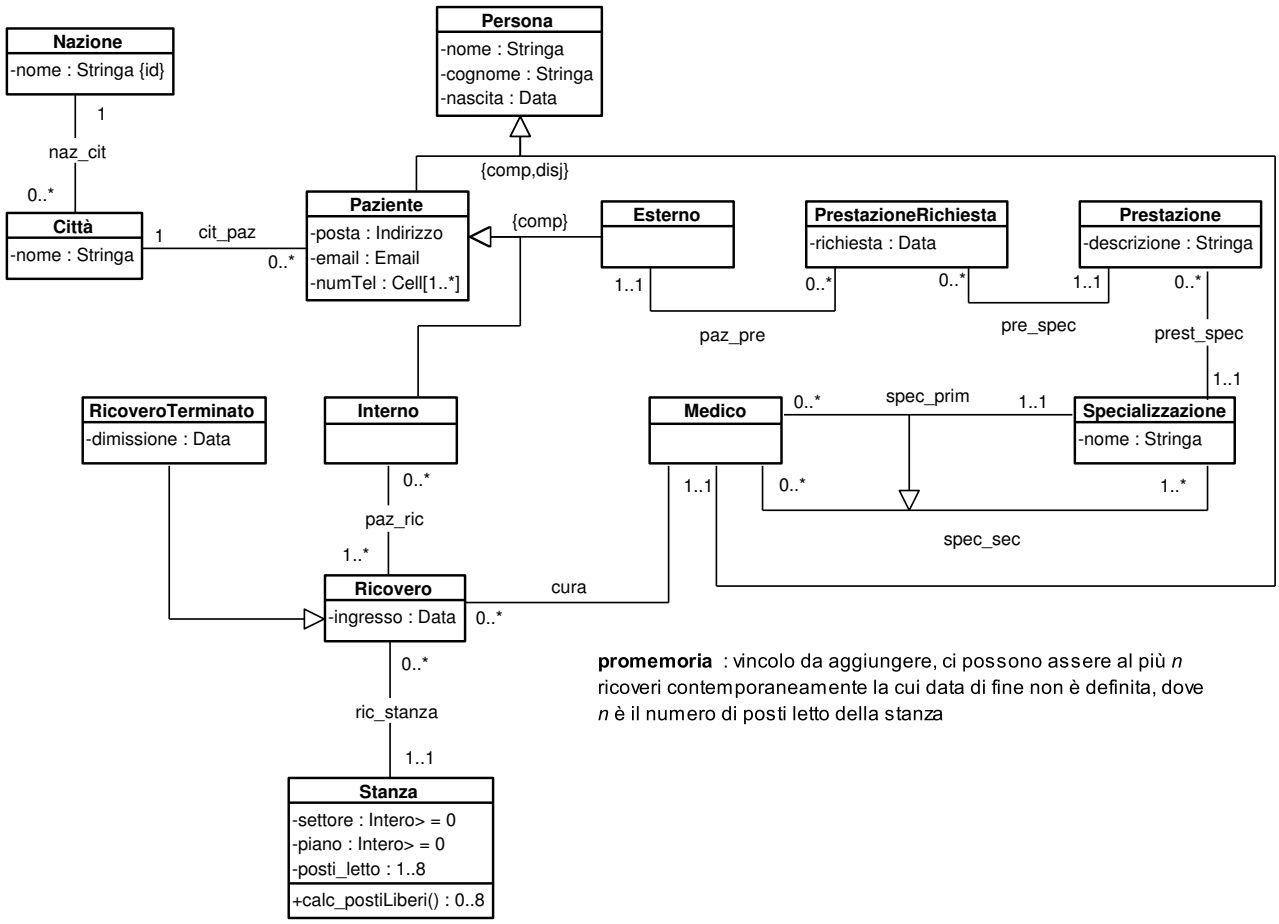
1. Paziente
    - 1.1 nome
    - 1.2 cognome
    - 1.3 data di nascita
    - 1.4 recapiti telefonici [1..*]
    - 1.5 email
    - 1.6 recapito postale
    - 1.7 interno o esterno?
        - 1.7.1 se esterno, prestazione medica richiesta (vedi REQ 4.)

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

- Per ogni RicoveroTerminato e Ricovero, RicoveroTerminato.data $\geq$ Ricovero.data

  $\forall rt, r, dr, drtRicovero(r) \wedge RicoveroTerminato(rt) \wedge DataRicovero(dr) \wedge$

  $DataRicoveroTerminato(drt)$ $\rightarrow$DataRicoveroTerminato $\geq DataRicovero$

- Per Ogni PazienteEsterno, questo non può essere anche interno

  $\forall p, dt, dpPazienteInterno(p) \wedge DataRicovero(dt) \wedge DataPrestazioneEsterna(dp)$

  $\wedge PazienteInRicovero(c, p, dt) \rightarrow \neg PazienteEsterno(p) \wedge PazienteInEsterna(p, dp)$

- Un paziente Non può essere ricoverato prima della sua nascita

  $\forall p, n, r, drPaziente(p) \wedge nascita(p, n) \wedge Ricovero(r) \wedge dataR(dr, r)$

  $\rightarrow n >= dr$

- Un paziente Non può fare richiesta di prenotazione prima della sua nascita $\forall p, n, r, drPaziente(p) \wedge$ $nascita(p, n) \wedge PrenotazioneRichiesta(r) \wedge dataR(dr, r)$

  $\rightarrow n >= dr$

- Non possono esserci più link(Ricovero,Stanza): ric_stanza rispetto a Stanza.posti$_l$$etto$

  $\forall s, r, rs, pStanza(s) PostiLettoStanza(s, p) \wedge Ricovero(r)$

  $\wedge E = \{(s, r) | StanzaRicovero(s, r)\}$ $|E| <= p$

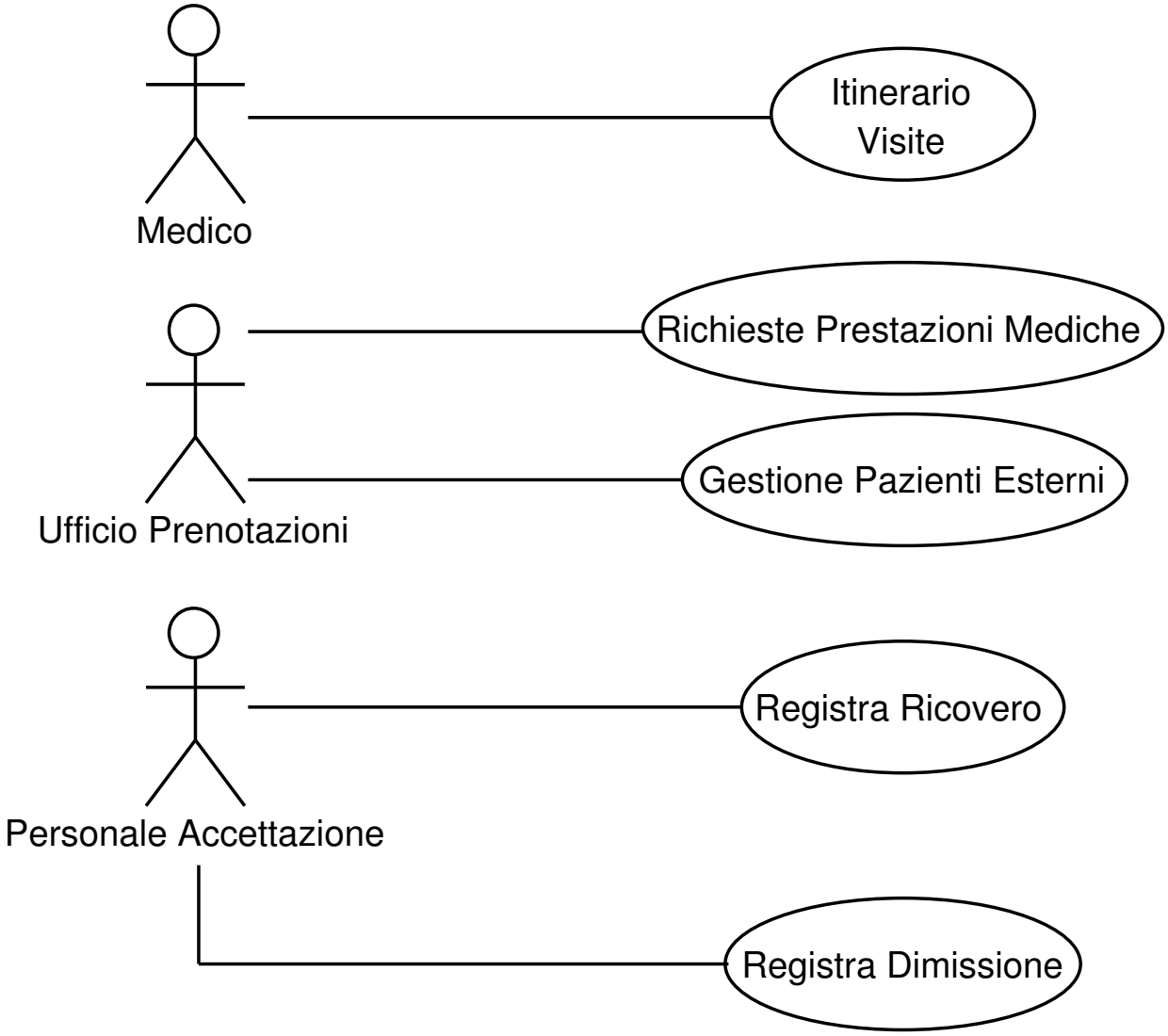
5 Operazioni di Classe

5.1 Classe Stanza

calcola_postiLiberi(): 0..8

- preCondizioni: nessuna
- postCondizioni:
  $R = \{r | ricoveriStanza(r, this \wedge \neg RicoveroTerminato(r))\}$
  sia $p = PostiLettoStanza(this, p)$
  return p-—R—

6 Diagramma UseCase



7 Segnatura UseCase

Segnatura di **tutti** gli useCase

7.1 Itinerario Visite

- calcola_itinerario (m:Medico): Itinerario (un insieme)

7.2 Registra ricovero

- calcola_posti_letto_liberi()
- ricovera_paz(p:Paziente, s:Stanza, i:Data): Ricovero

7.3 Richieste Prestazioni

- accetta_esterna(p:PazienteEsterno, i: Data, d: Stringa): Prestazione e PrestazioneRichiesta

7.4 Gestione Pazienti Esterni

- calcola_medici_idonei(s:Specializzazione): Insieme di Medici

7.5 Registra Dimissione

- dimetti(p:Paziente,r:Ricovero,d:Data): RicoveroTerminato

8 Specifica UseCase

Veranno scritte le specifiche delle Operazioni di UseCase solo più importanti

8.1 Itinerario Visite

Il professore a lezione ha detto che ci avrebbe mostrato la soluzione.

8.2 Gestione Esterni

Questo UseCase utilizza l'operazione calcola_medici_idonei(s:Specializzazione) per ritornare un insieme di medici che possono svolgere la prenotazione.

- preCondizioni: $\exists s Specializzazione(s)$
- postCondizioni:
  $Medici\_spec\_prim = \{m | Medico(m) \wedge SpecializzazionePrimaria(m, s)\}$
  $Medici\_spec\_sec = \{m | Medico(m) \wedge SpecializzazioneSecondaria(m, s)\}$
  Result:
  - Medici_spec_prim se $|Medici\_spec\_prim| > 0$
  - Medici_spec_sec se $(|Medici\_spec\_prim| = 0 \wedge |Medici\_spec\_sec| > 0)$
  - Null se $(|Medici\_spec\_prim| = 0 \wedge |Medici\_spec\_sec| = 0)$

8.3 Registra Ricovero

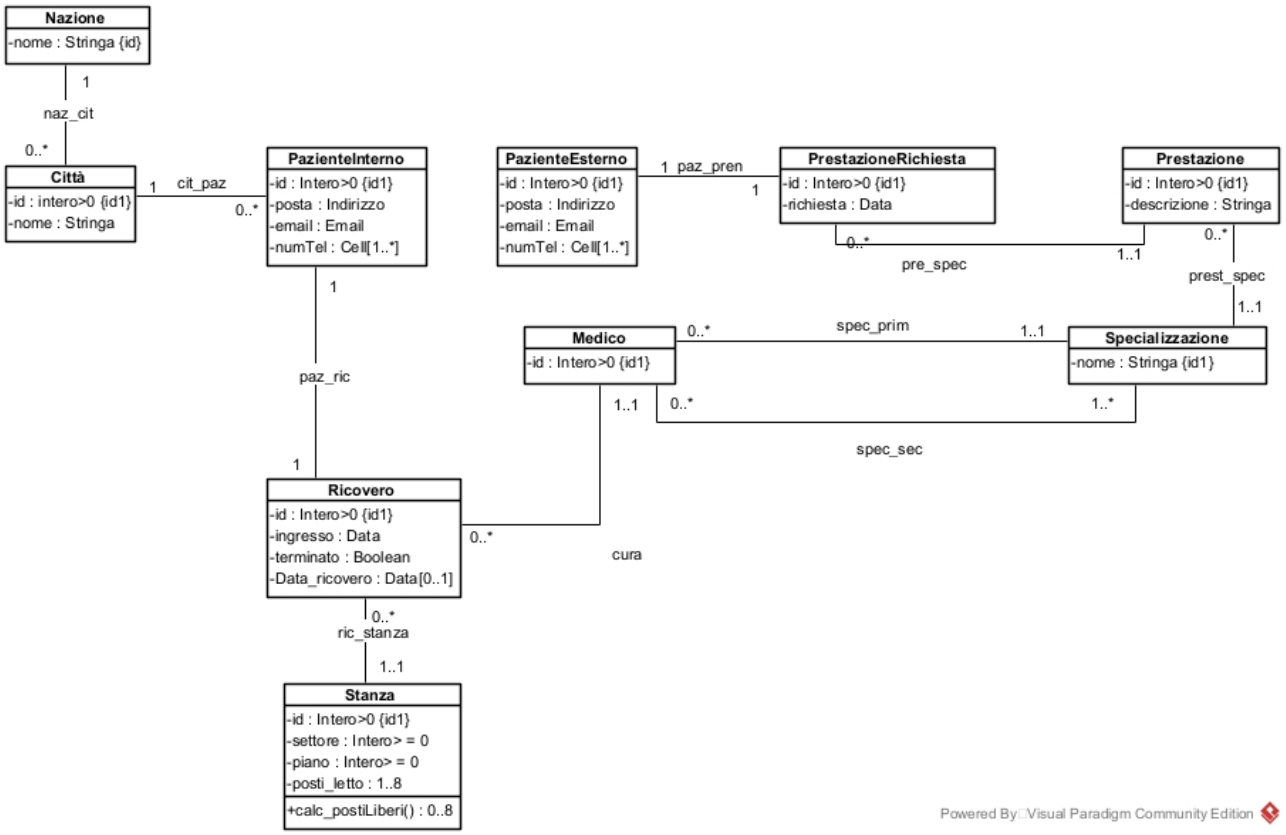
Questo useCase usa l'operazione calcola_posti_letto_liberi() per verificare i posti letto liberi

- preCondizioni: nessuna
- postCondizioni:
  Sia s Stanza(s), P={ (s,posti) | calcola_postiLiberi(s,posti) }
  Result = $\sum_{(s,posti) \in P} posti$

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

9 Ristrutturazione

9.1 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.

9.2 Tipi e Domini

9.2.1 Tipi

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

9.2.2 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]')

9.3 Vincoli Esterni

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

- Data Una SpecializzazionePrimaria essa non può essere SpecializzazioneSecondaria per lo stesso medico.
  $\forall m, sMedico(m) \land Specializzazione(s) \land SpecializzazionePrimaria(m, s)$
  $\rightarrow \neg SpecializzazioneSecondaria(m, s)$

9.4 Use Case

Gli use case non violano la nuova ristrutturazione.

9.5 Traduzione diretta del diagramma UML delle classi ristrutturato

Saranno scritte tutte le tabelle da creare.

- Nazione(**nome**:varchar)
- Citta(**id**:integer, nome:varchar)
  v.inclusione: Città(Nome) occorre in *naz_cit*(nazione)
- *naz_cit*(**Nazione**:varchar,**Citta**:integer)
  foreign key: Nazione references Nazione(nome)
  foreign key: Citta references Citta(id)

- PazienteInterno(**id**:integer, posta: Indirizzo, email: Email, numTel:Cell)
  v.inclusione PazienteInterno(id) occorre in *cit_paz*(Paziente)
- *cit_paz*(**Citta**:varchar,**Paziente**:integer)
  foreign key: Nazione references Paziente(id)
  foreign key: Citta references Citta(id)

- Ricovero(**id**: integer, ingresso: Date, terminato: Boolean: *data_ricovero*:Date*)
  v.inclusione Ricovero(id) occorre in *ric_stanza*(Ricovero) v.inclusione Ricovero(id) occorre in *cura*(Ricovero)
- *paz_ric*(**Paziente**:integer,**Ricovero**:integer)
  foreign key: Nazione references Paziente(id)
  foreign key: Citta references Ricovero(id)

- Stanza(**id**: integer, settore: intero>0,piano: intero>=0 *posti_letto* : 1..8)
- *ric_stanza*(**Ricovero**: intero,**Stanza**:integer)
  foreign key: Ricovero references Ricovero(id)
  foreign key: Stanza references Stanza(id)

- PazienteEsterno(**id**:integer, posta: Indirizzo, email: Email, numTel:Cell)
- PrenotazioneRichiesta(**id**:integer, richiesta: Date)
  v.inclusione PrenotazioneRichiesta(id) occorre in *pre_spec*(Prenotazione)
- *paz_pren*(**Paziente**:integer,**Prenotazione**:integer)
  foreign key: Paziente references PazienteEsterno(id)
  foreign key: Prenotazione references PrenotazioneRichiesta(id)

- Prenotazione(**id**:integer, descrizione: varchar(100))
  v.inclusione Prenotazione(id) occorre in *prest_spec*(Prenotazione)

- *pre_spec*(**PrenotazioneRichiesta**: integer,**Prenotazione**: integer)
  foreign key: Prenotazione references Prenotazione(id)
  foreign key: PrenotazioneRichiesta references PrenotazioneRichiesta(id)

- Specializzazione(**nome**: varchar(100))

- *prest*$_s$*pec*(**Prenotazione**: integer,**Specializzazione**: varchar(100))
  foreign key: Prenotazione references Prenotazione(id)
  foreign key: Specializzazione references Specializzazione(varchar)

- Medico(**id**: integer))
  v.inclusione Medico(id) occorre in *spec_prim*(Medico)
  v.inclusione Medico(id) occorre in *spec_sec*(Medico)

- *spec_prim*(**Medico**: intero,**Specializzazione**: varchar(100))
  foreign key: Medico references Medico(id)
  foreign key: Specializzazione references Specializzazione(nome)

- *spec_sec*(**Medico**: intero,**Specializzazione**: varchar(100))
  foreign key: Medico references Medico(id)
  foreign key: Specializzazione references Specializzazione(nome)

- cura(**Ricovero**: intero,**Medico**:integer)
  foreign key: Ricovero references Ricovero(id)
  foreign key: Medico references Medico(id)