1 Analisi concettuale

Domanda 1 (10 minuti) Raffinare la specifica dei requisiti eliminando inconsistenze, omissioni e ridondanze e producendo un elenco numerato di requisiti il meno ambiguo possibile. (La risposta a questa domanda non sarà valutata, ma si consiglia di svolgere accuratamente questo passo, in quanto può facilitare di molto le attività di progetto.)

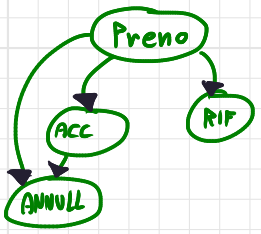
Risposta

1. Utente
   - nome
   - cognome
   - email
   - cell
   - GESTORE?
     - cod fisc
     - identificato?

2. Struttura
   - proprietario (uno o più)
   - nome
   - indirizzo
   - tipologia
   - stelle 1..7
   3. Sistemazione (Camera)
      - 3.1 nome
      - 3.2 posti letto
      - 3.3 TARIFFA (dipende dal giorno e numero di occupanti)

4 Prenotazione
- Utente richiedente
- date [da - a]
- Sistemazioni [di una stessa Struttura]
- occupanti per sistemazione
- metodo di pagamento

Albero accetta



[Versione 2024-06-05.UML]

[continua alla pagina seguente]

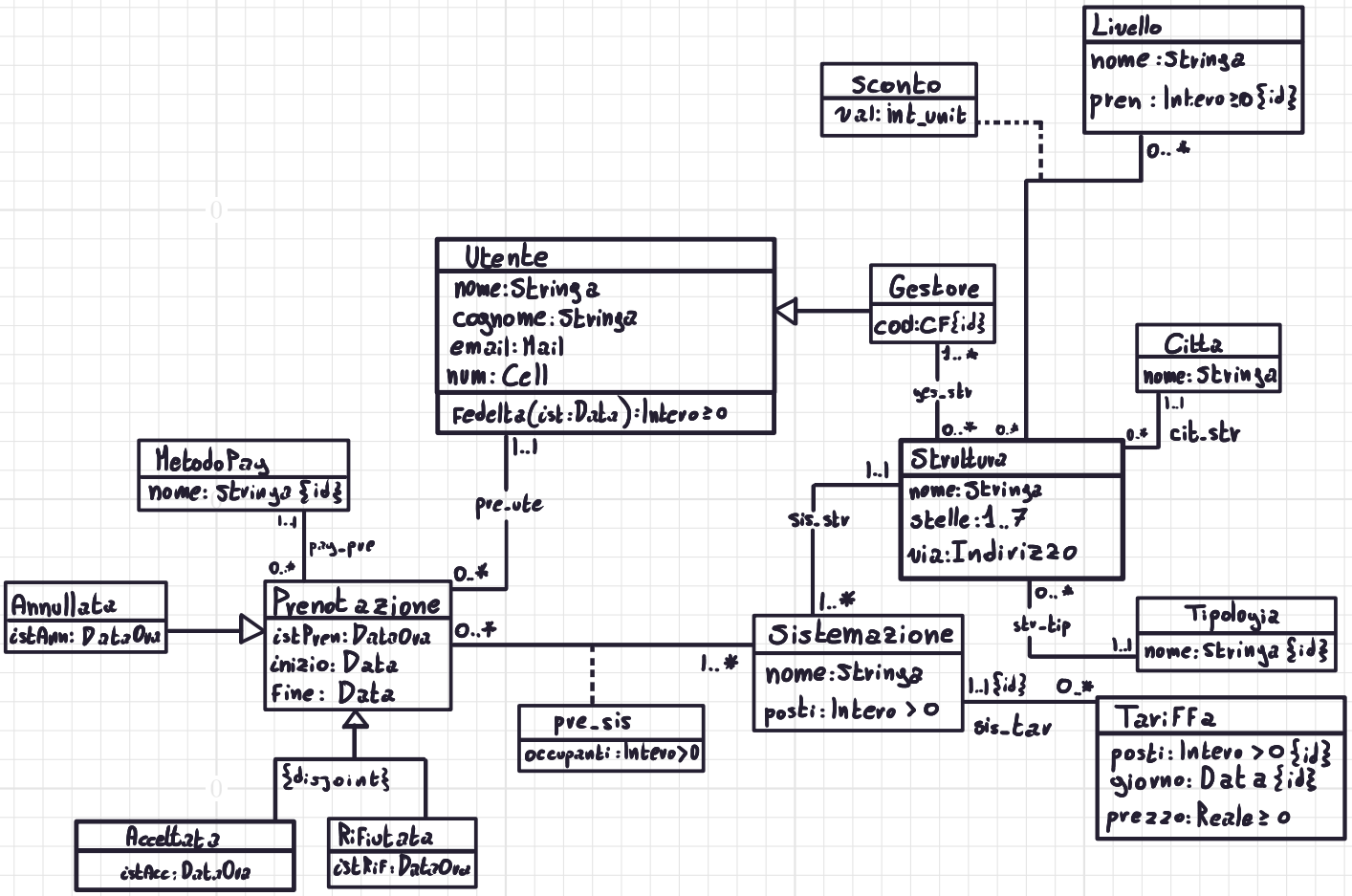
Risposta alla Domanda 1 (segue)

Domanda 2 (45 minuti; 75 minuti al massimo) Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma UML concettuale delle classi per l'applicazione, le specifiche di classi, associazioni, tipi di dato e vincoli esterni.

Una risposta soddisfacente a questa domanda è condizione *necessaria* (ma non sufficiente) per superare la prova.

Diagramma UML concettuale delle classi

Produrre un diagramma UML concettuale delle classi per l'applicazione in termini di classi, associazioni, attributi, generalizzazioni, operazioni di classe.



[continua alla pagina seguente]

Risposta alla Domanda 2 (segue)

Specifiche delle classi o associazioni Per ogni classe o associazione del diagramma **con operazioni o vincoli**:

- Definire la specifica formale di eventuali operazioni necessarie a modellare i requisiti contrassegnati dalla barra laterale, ed eventuali vincoli esterni. Usare la logica del primo ordine estesa con teoria degli insiemi e semantica di mondo reale vista nel corso, usando il seguente alfabeto:
  - Un simbolo di predicato $C/1$ per ogni classà C.
    Semantica di $C(x)$: x è una istanza di C.
  - Un simbolo di predicato $T/1$ per ogni tipo di dato T.
    Semantica di $T(x)$: x è un valore di T.
  - Un simbolo di predicato assoc$/2$ per ogni associazione binaria assoc.
    Semantica di assoc(c_1, c_2): (c_1, c_2) è una istanza di assoc.
  - Un simbolo di predicato attr$/2$ per ogni attributo attr di entità
    Semantica di attr(c, v): uno dei valori dell'attributo attr dell'istanza c è v.
  - Un simbolo di predicato attr$/3$ per ogni attributo attr di associazione binaria.
    Semantica di attr(c_1, c_2, v): uno dei valori dell'attr. attr del link (c_1, c_2) è v.
  - Un simbolo di predicato op$/(n+2)$ per ogni operazione di classe ad n argomenti.
    Semantica di op$(c, \text{arg}_1, \ldots, \text{arg}_n, v)$: uno dei valori di ritorno di op, quando invocata sull'istanza c e con argomenti $\text{arg}_1, \ldots, \text{arg}_n$ è v.
  - Il simbolo di $=/2$ (la cui interpretazione è la relazione che lega ogni elemento del dominio di interpretazione solo con se stesso) e opportuni simboli di predicato e di funzione, soggetti a semantica di modo reale, per relazioni e funzioni standard tra elementi dei tipi di dato, tra cui adesso$/0$, interpretato come il valore del dominio DataOra che rappresenta l'istante corrente.

Risposta

1 Tipo: ~~Classe~~ | **Associazione** (cerchiare)

Nome: Utente

Operazioni, vincoli:

fedelta(ist:Data):Intero ≥0

- pre-cond: nessuna
- post-cond:

$$P = \left\{ p \;\middle|\; \begin{array}{l} \text{Accettata}(p) \wedge \neg\text{Annullata}(p) \wedge \\ \neg\text{Rifiutata}(p) \wedge \text{pre\_ute}(p, \text{this}) \wedge \\ \exists i \ \text{inizio}(p, i) \wedge i \leq \text{ist} \wedge \exists K \\ K = \text{ist} - \text{`2 anni'} \wedge i \geq K \end{array} \right\}$$

Result = |P|

2 Tipo: ~~Classe~~ | **Associazione** (cerchiare)

Nome: Sistemazione

Operazioni, vincoli:

[∀. tariffa_per_ogni_posto]

$\forall s, p, t, g \ [\text{Sistemazione}(s) \wedge \text{posti}(s, p) \wedge \text{giorno}(t, g) \wedge \text{Tariffa}(t) \wedge \text{sis\_tar}(s, t)] \rightarrow [\forall n \ \text{Intero}(n) \wedge n \leq p \wedge n > 0] \rightarrow \exists t1 \ \text{Tariffa}(t1) \wedge \text{posti}(t1, n) \wedge \text{giorno}(t1, g) \wedge s \ \text{is\_tar}(s, t1)]$

3 Tipo: **Classe** | **Associazione** (cerchiare)

Nome: Prenotazione

Operazioni, vincoli:

[V.no_intersezione_pren]
$\forall p1,p2,s,i1,i2,f1,f2\ [Prenotazione(p1) \wedge inizio(p1,i1) \wedge Prenotazione(p2) \wedge inizio(p2,i2) \wedge Fine(p1,f1) \wedge Fine(p2,f2) \wedge p1 \neq p2 \wedge pre\_sis(p1,s) \wedge pre\_sis(p2,s)] \rightarrow f1 < i2 \vee f2 < i1$

[V.posti_pren]
$\forall p,s,ps,pp\ [Prenotazione(p) \wedge pre\_sis(p,s) \wedge posti(s,ps) \wedge occupanti(p,s,pp)] \rightarrow pp \leq ps$

[V.continuita_pren]
$\forall p,i,f\ [Prenotazione(p) \wedge inizio(p,i) \wedge Fine(p,f)] \rightarrow i \leq F$

6 Tipo: **Classe** | **Associazione** (cerchiare)

Nome: Prenotazione

Operazioni, vincoli:

[V.prenota_poi_va]
$\forall p,i,ip,y\ [Prenotazione(p) \wedge inizio(p,i) \wedge istPren(p,ip) \wedge Giorno(ip,y)] \rightarrow g \leq i$

[V.prenota_stessa_str]
$\forall p,s1,s2,st1,st2\ [s1 \neq s2 \wedge Prenotazione(p) \wedge pre\_sis(p,s1) \wedge pre\_sis(p,s2) \wedge sis\_str(s1,st1) \wedge sis\_str(s2,st2)] \rightarrow st1 = st2$

[V.annullata_se_non_rif]
$\forall p\ [\exists i\ istAnn(p,i)] \rightarrow \neg[\exists i'\ istRif(p,i')]$

[V.ann_dopo_acc]
$\forall p,ian,iac\ [Prenotazione(p) \wedge istAcc(p,iac) \wedge istAnn(p,ian)] \rightarrow ian > iac$

4 Tipo: **Classe** | **Associazione** (cerchiare)

Nome: Prenotazione

Operazioni, vincoli:

[V.interazioni_pren]
$\forall p,ip,i,i2,ii\ [Prenotazione(p) \wedge istPren(p,ip) \wedge inizio(p,i) \wedge DataOra(i,ii) \wedge [istAcc(p,i2) \vee istRif(p,i2) \vee istAnn(p,i2)]] \rightarrow ip \leq i2 < ii$

7 Tipo: **Classe** | **Associazione** (cerchiare)

Nome: .....

Operazioni, vincoli:

5 Tipo: **Classe** | **Associazione** (cerchiare)

Nome: .....

Operazioni, vincoli:

8 Tipo: **Classe** | **Associazione** (cerchiare)

Nome: .....

Operazioni, vincoli:

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

Specifiche dei tipi di dato, specifiche di ulteriori vincoli esterni ed altre specifiche

Indirizzo : (via : Stringa, civico : Intero > 0)

Mail : stringa secondo standard

Cell : stringa secondo standard

CF : stringa secondo standard

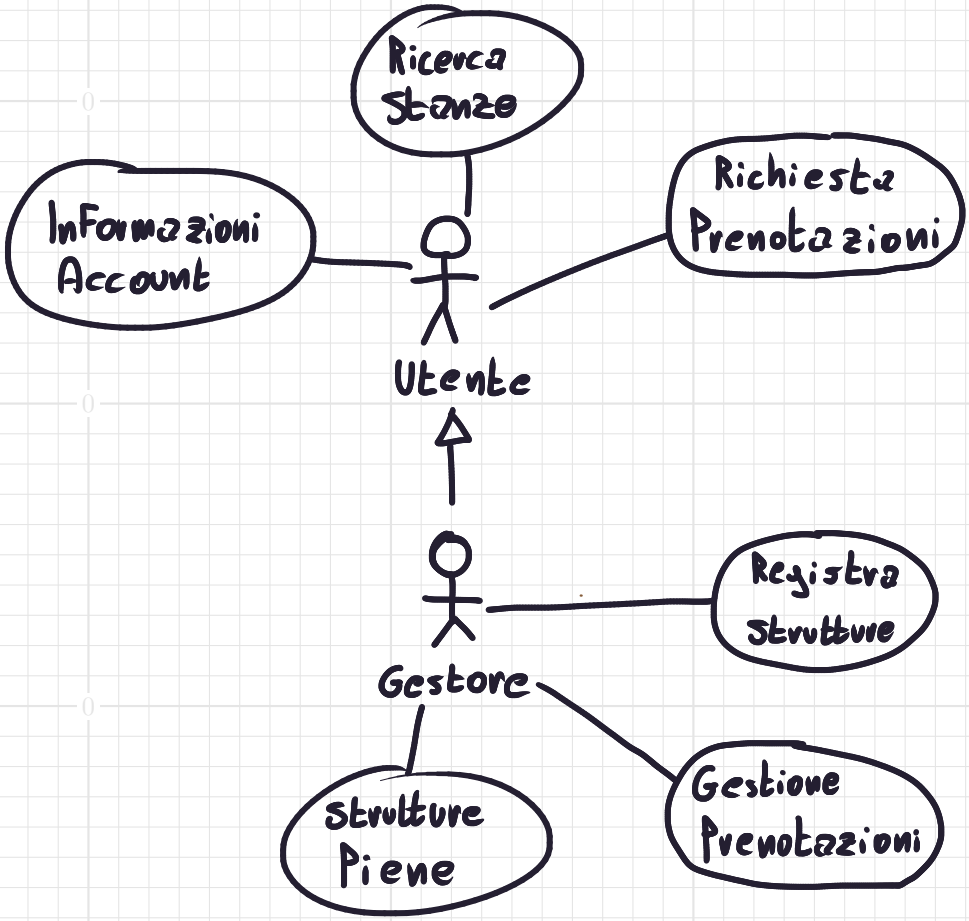
int_unit : Reale ∈ (0,1)

[continua alla pagina seguente]

Risposta alla Domanda 2 (segue)

Domanda 3 (5 minuti; 10 minuti al massimo) Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma UML degli use-case che definisca ad alto livello tutte le funzionalità richieste al sistema.

Risposta



Domanda 4 (10 minuti) Proseguire la fase di Analisi Concettuale dei requisiti definendo la **segnatura** delle operazioni in ogni use-case.

Risposta

Domanda 5 (30 minuti; 60 minuti al massimo) Proseguire la fase di Analisi Concettuale dei requisiti producendo le specifiche concettuali per le operazioni di use-case, **limitandosi** a quelle necessarie a modellare i requisiti contrassegnati dalla barra laterale (come quella qui a sinistra), ed includendo eventuali operazioni ausiliarie. In particolare, per ogni operazione, definire segnatura, precondizioni e postcondizioni utilizzando il linguaggio della logica del primo ordine. Si assuma lo stesso vocabolario definito alla Domanda 2.

Una risposta soddisfacente a questa domanda è condizione *necessaria* (ma non sufficiente) per superare la prova.

Risposta

struttura_piene(s:Struttura, u:Gestore): (1..12, Intero ≥ 0) [12..12]

- pre-cond: ges_str(u,s)
- post-cond

$$M = \left\{ (m, n) \;\middle|\; \begin{array}{l} Mese(m) \wedge \\ n = \left\{ g \;\middle|\; \begin{array}{l} Data(g) \wedge Mese(g,n) \wedge \exists\, an\ Anno(adesso, an) \\ \wedge\ Anno(g, an) \wedge [\forall\, sis\ Sistemazione(s) \wedge \\ sis\_str(sis, s)] \rightarrow [\exists\, p, i, f\ Accettata(p) \wedge \neg Annullato(p) \wedge \\ pre\_sis(p, sis) \wedge inizio(p, i) \wedge fine(p, f) \wedge i \leq g \leq f] \end{array} \right\} \end{array} \right\}$$

Result = M

---

cerca_sistemazioni(i:Data, f:Data, c:Citta, t:Tipologia, st:1..7, pr:Reale ≥ 0, nper:Intero ≥ 0, u:utente): Struttura[0..*]

- pre-cond: $i \leq f$
- post:

$$R = \left\{ str \;\middle|\; \begin{array}{l} str\_tip(str, t) \wedge \exists K\ stelle(str, K) \wedge \\ K \geq st \wedge cit\_str(str, c) \wedge \\ SG \left\{ (g, S) \;\middle|\; S = \left\{ (sis, np, cos) \;\middle|\; \begin{array}{l} sis\_str(sis, str) \wedge \exists K'\ posti(sis, K') \wedge K' \geq np \wedge \exists\, tar \\ sis\_tar(sis, tar) \wedge posti(tar, np) \wedge prezzo(tar, cos) \\ \wedge\ giorno(tar, g) \wedge \forall pren, ip, fp\ pre\_sis(pren, sis) \\ \wedge inizio(pren, ip) \wedge fine(pren, fp) \rightarrow g < i \vee f < g \end{array} \right\} \right. \right. \\ \qquad i \leq g \leq f \left. \right\} \\ \wedge \left[\forall g'\ (g', S) \in SG \rightarrow \left[\sum_{\substack{(sis, np, cos) \\ \in S \wedge (S, g') \\ \in SG}} np = nper \right] \right] \wedge \sum_{\substack{(sis, np, cos) \in S \\ \wedge \exists g'\ (g', S) \\ \in SG}} cos \cdot m \leq pr \\ \exists\, m \\ \left[\left[\exists\, mol, npren, fed, npl \right. \right. \\ \quad fedelta(u, adesso, npren) \wedge Livello(fed) \wedge pren(fed, npl) \wedge npren \geq npl \wedge sconto(str, fed) \wedge \\ \quad val(str, fed, mol) \wedge [\neg \exists\, fed2, npl2\ sconto(str, fed2) \wedge pren(fed2, npl2) \wedge npren \geq npl2 \\ \quad \left. npl2 > npl] \rightarrow m = mol \right] \wedge \\ \left[\left[\neg \exists\, mol, npren, fed, npl \right. \right. \\ \quad fedelta(u, adesso, npren) \wedge Livello(fed) \wedge pren(fed, npl) \wedge npren \geq npl \wedge sconto(str, fed) \wedge \\ \quad val(str, fed, mol) \wedge [\neg \exists\, fed2, npl2\ sconto(str, fed2) \wedge pren(fed2, npl2) \wedge npren \geq npl2 \\ \quad \left. npl2 > npl] \rightarrow m = 1 \right] \end{array} \right\}$$

Result = R

[continua alla pagina seguente]

Risposta alla Domanda 5 (segue)

2 Progettazione della base dati e delle funzionalità

Domanda 6 (20 minuti; 30 minuti al massimo) Iniziare la fase di progettazione logica della base di dati decidendo il DBMS da utilizzare e ristrutturando lo schema UML delle classi concettuale, il dizionario dei dati e i vincoli esterni. In particolare:

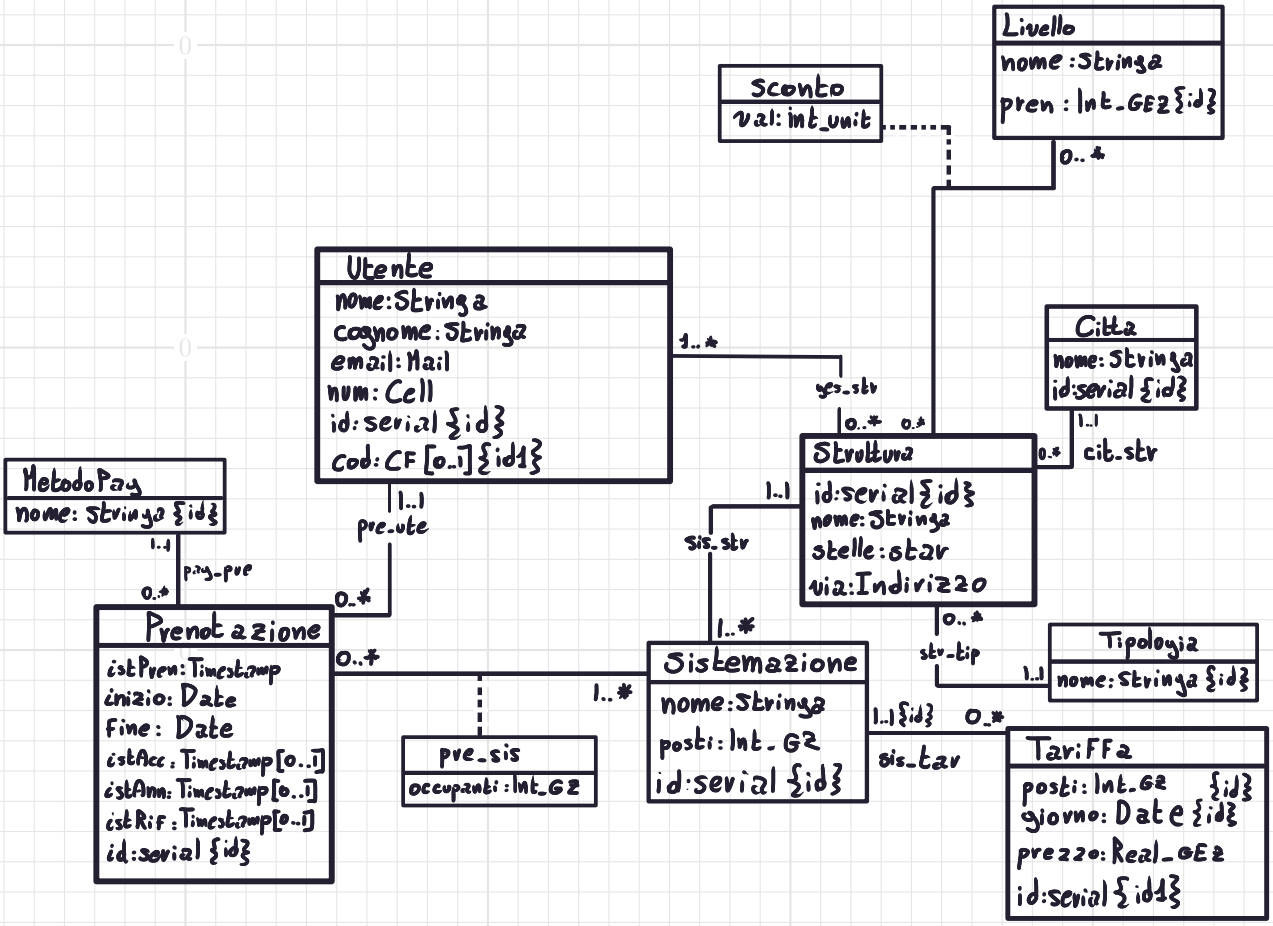
- progettare una corrispondenza tra i tipi di dato concettuali ed opportuni domini SQL (domini base o utente, oppure realizzati mediante relazioni aggiuntive) supportati dal DBMS scelto
- eliminare attributi multivalore o composti
- eliminare relazioni is-a e generalizzazioni
- definire un identificatore primario per ogni classe
- ristrutturare i vincoli esterni per renderli consistenti con la struttura del nuovo diagramma.

Descrivere brevemente le principali scelte effettuate.

DBMS da utilizzare PostgreSQL.....

Corrispondenza tra tipi di dato concettuali e domini supportati dal DBMS

Diagramma UML delle classi ristrutturato



[Versione 2024-06-05.UML]

Breve descrizione delle scelte effettuate durante la ristrutturazione

Vincoli esterni introdotti o modificati durante la fase di ristrutturazione
(si omettano i vincoli esterni la cui formulazione è rimasta identica a seguito della ristrutturazione)

[V. utente_gestore]

$\forall u\ \text{Utente}(u) \wedge \exists s\ \text{ges\_str}(u,s) \rightarrow \exists c\ \text{cod}(u.c)$

[V. acc_o_rif]

$\forall p\ \text{Prenotazione}(p) \wedge \exists i\ \text{istAcc}(p,i) \rightarrow \neg\exists i'\ \text{istRif}(p,i')$

[continua alla pagina seguente]

Risposta alla Domanda 6 (segue)

Domanda 7 (30 minuti; 60 minuti al massimo) Proseguire la fase di progettazione logica della base di dati producendo lo schema relazionale della base dati e i relativi vincoli a partire dallo schema UML delle classi ristrutturato.

Una risposta soddisfacente a questa domanda è condizione *necessaria* (ma non sufficiente) per superare la prova.

1 **Relazione** MetodoPag ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	nome						
Domini	Stringa						

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

2 **Relazione** Utente ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	nome	cognome	email	num	id	cod *		
Domini	Stringa	Stringa	Mail	Cell	serial	CF		

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): unique(cod);

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

3 **Relazione** Tipologia ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	nome						
Domini	Stringa						

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

4 **Relazione** Citta ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	nome	id					
Domini	Stringa	serial					

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

5 **Relazione** Struttura ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	id	nome	stelle	via	citta	tipo		
Domini	serial	Stringa	Star	Indirizzo	Integer	nome		

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): Fk citta ref Citta(id);

Fk tipo ref Tipologia(nome);

La relazione accorpa le relazioni che implementano le seguenti associazioni: cit.str., str.tip .....

[Versione 2024-06-05.UML]

6 **Relazione** ges_str ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	gestore	struttura2						
Domini	CF	Integer						

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

FK gestore ref Utente(cod);

FK struttura ref Struttura(id);

Struttura(id) ⊆ ges_str(struttura);

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

7 **Relazione** Prenotazione ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	istPren	inizio	Fine	istAcc *	istRif *	istAnn *	id	metodo
Domini	TimeStamp	Date	Date	TimeStamp	TimeStamp	TimeStamp	serial	Stringa

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): FK metodo ref MetodoPay(nome);

check(inizio <= Fine); check(istRif ≥ istPren ∧ istAcc ≥ istPren ∧ istAnn ≥ istPren ∧

istRif ≤ cast(inizio as Date) ∧ istAnn ≤ cast(inizio as Date) ∧ istAcc ≤ cast(inizio as Date));

La relazione accorpa le relazioni che implementano le seguenti associazioni: pre_pay .....

8 **Relazione** Prenotazione ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	utente							
Domini	Integer							

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): check(istAnn > istAcc);

FK utente ref Utente(id); check(istRif is NULL OR istAcc is NULL); check(cast(istPren as Date) <= inizio);

check(istRif is NULL OR istAnn is NULL);

La relazione accorpa le relazioni che implementano le seguenti associazioni: pre_ute .....

9 **Relazione** Sistemazione ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	nome	posti	id	struttura				
Domini	Stringa	Int_GZ	serial	Integer				

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): FK struttura ref Struttura(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: sis_str .....

10 **Relazione** pre_sis ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	occupanti	prenotazione	sistemazione					
Domini	Int_GZ	Integer	Integer					

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

FK prenotazione ref Prenotazione(id); FK sistemazione ref Sistemazione(id);

Prenotazione(id) ⊆ pre_sis(prenotazione);

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

[Versione 2024-06-05.UML]

11 **Relazione** Tariffa ..... (nome) Derivante da: ~~**classe**~~ | **associazione** (cerchiare)

Attributi	id	posti	prezzo	giorno	sistemazione			
Domini	serial	Int_GZ	real_GEZ	Date	Integer			

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

FK sistemazione ref Sistemazione(id);

unique(sistemazione, giorno, posti);

La relazione accorpa le relazioni che implementano le seguenti associazioni: sist_tar .....

12 **Relazione** Livello ..... (nome) Derivante da: ~~**classe**~~ | **associazione** (cerchiare)

Attributi	nome	pren						
Domini	Stringa	Int_GEZ						

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

13 **Relazione** Sconto ..... (nome) Derivante da: **classe** | ~~**associazione**~~ (cerchiare)

Attributi	struttura	livello	vcl					
Domini	Integer	Int_GEZ	int_unit					

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

FK livello ref Livello(pren); FK struttura ref Struttura(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

14 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

15 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

16 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

17 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

18 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

19 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

20 **Relazione** ..... (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi								
Domini								

Gli attributi chiave primaria sono sottolineati, quelli i cui valori possono essere NULL sono contrassegnati con *

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni: .....

[continua a pagina ..... delle minute]

Ulteriori vincoli esterni

Per ogni ulteriore vincolo esterno (non ancora espresso perché non definibile mediante vincoli di chiave, foreign key, ennupla, dominio, inclusione), progettare un trigger che lo implementi, definendo: (a) gli eventi da intercettare (inserimento, modifica, eliminazione di ennuple); (b) quando intercettare tali eventi (appena prima o subito dopo l'evento intercettato); (c) la relativa funzione in pseudo-codice con SQL immerso che implementa il controllo del vincolo.

T. no_intersezioni_pren
Insert o Update pre_sis

```
Error: EXISTS( SELECT *
               FROM Prenotazione p1,Prenotazione p2,pre_sis ps
               WHERE p1.id = ps.prenotazione AND
                     p2.id = new.prenotazione AND
                     new.sistemazione = ps.sistemazione

              AND (p1.inizio, p1.Fine) OVERLAPS (p2.inizio, p2.Fine));
```

T. prenota_stessa_str
Insert o Update pre_sis

```
Error: EXISTS(SELECT * FROM Sistemazione s, pre_sis ps, Sistemazione s2
              WHERE ps.prenotazione = new.prenotazione AND
              s.id = ps.sistemazione AND s2.id = new.sistemazione AND
              s.struttura <> s2.struttura);
```

T. posti_pren
Insert o Update pre_sis

```
Error: EXISTS( SELECT * FROM Sistemazione WHERE s.id = new.sistemazione AND s.posti < new.occupanti);
```

T. tariffa_per_ogni_posto
Insert o Update sistemazione

so che è sbagliata ma il concetto è quello

```
ERROR = EXISTS(WITH Q as (SELECT t.giorno, count(*) as c
                          FROM Tariffa t
                          WHERE t.sistemazione = new.id
                          GROUP BY t.giorno)

               SELECT * FROM Q WHERE c <> 0 AND c <> new.posti);
```


[continua alla pagina seguente]

Risposta alla Domanda 7 (segue)

Domanda 8 (30 minuti; 45 minuti al massimo) Proseguire la fase di progettazione dell'applicazione producendo le specifiche realizzative delle operazioni di classe e/o use-case definite per modellare i requisiti contrassegnati dalla barra laterale della specifica dei requisiti.

In particolare, per ogni operazione definire la segnatura, in termini di nome dell'operazione, nomi e dominio SQL degli argomenti, dominio SQL dell'eventuale valore di ritorno, e un algoritmo in pseudo-codice con SQL immerso che verifichi le precondizioni e garantisca il raggiungimento delle postcondizioni definite in fase di Analisi. Specificare, per ogni operazione, se debba essere implementata nel DBMS o nel *back-end*.

Una risposta soddisfacente a questa domanda è condizione *necessaria* (ma non sufficiente) per superare la prova.

Risposta

```
create Function struttura_piena(str:Integer) : Insieme(<Int_G2, Int_GE2>)

    WITH Q as

    (SELECT pr.inizio, count(DISTINCT s.id) as sis_occ
     FROM Sistemazione S, Prenotazione pr, pre_sis ps
     WHERE S.struttura = str AND pr.id=ps.prenotazione AND
     ps.sistemazione = S.id AND extract('year' From pr.inizio)=
     extract('year' From NOW())
     GROUP BY pr.inizio)

     N as ( SELECT COUNT(*) as n FROM Sistemazione WHERE struttura = str)

     SELECT extract('month' From pr.inizio), count(pr.inizio)
     FROM Q, N
     WHERE Q.sis_occ = N.n
     GROUP BY extract('month' From pr.inizio);


create Fu
```

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

Risposta alla Domanda 8 (segue)

Tempo totale stimato per svolgere questa prova: 180 minuti (tempo totale concesso: 300 minuti).

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]