

Esame Es.20230911 – Prova scritta dell'11 Settembre 2023

Si vuole progettare e realizzare *CozyRooms*, un sistema per la gestione di strutture ricettive e dei soggiorni presso di esse.

Il sistema dovrà servire agli utenti per pubblicare offerte riguardo le proprie strutture (ad esempio alberghi, appartamenti e case vacanza), gestire le prenotazioni e i soggiorni, ricercare strutture e sistemazioni, e prenotare soggiorni.

Gli utenti si iscrivono a *CozyRooms* fornendo nome, cognome, indirizzo e-mail e numero di telefono. Gli utenti gestori di strutture devono fornire anche il codice fiscale e superare un processo di identificazione (il quale è portato avanti utilizzando un servizio esterno).

I gestori di strutture identificati con successo possono registrare le proprie strutture. Di una struttura interessa conoscere il nome, l'indirizzo, la tipologia (albergo, casa vacanze, bed & breakfast, ecc.), il numero di stelle (un intero compreso tra 1 e 7), ed i gestori (possano essere più d'uno).

Ogni struttura dispone di una o più sistemazioni. Ad esempio, un hotel dispone in genere di numerose camere, mentre un residence può includere diversi appartamenti. Di ogni sistemazione interessa conoscere il nome, e il numero di posti letto.

Il costo del pernottamento non è costante durante l'anno, ma dipende dalla data. Il sistema deve permettere ai gestori delle strutture di specificare, per ogni sistemazione, la tariffa applicata per ogni giorno dell'anno solare.

Inoltre, il costo del pernottamento in una sistemazione dipende anche dal numero di occupanti. Ad esempio, una camera doppia di un certo hotel, in una certa data, potrebbe costare 60 Euro a notte per un solo occupante e 80 Euro a notte per due occupanti. Il gestore deve specificare la tariffa di per ogni numero di occupanti da 1 al numero di posti letto presenti nella sistemazione.

Gli utenti di *CozyRooms* che vogliono soggiornare presso una struttura, lo fanno tramite un semplice servizio di prenotazione. Al momento della prenotazione, un utente sceglie un intervallo di date, una o più sistemazioni offerte da una stessa struttura e un numero di occupanti per ognuna di esse e fornisce un método di pagamento. Non deve essere possibile richiedere la prenotazione per una sistemazione che è già prenotata in almeno una data nel periodo richiesto. Il gestore della struttura può quindi accettare o rifiutare le prenotazioni. L'utente che effettua la prenotazione può cancellarla fino al momento del soggiorno (si assuma che la gestione delle penali per cancellazione tardiva sia gestita da un sistema esterno).

*CozyRooms* prevede un sistema di fedeltà per i propri utenti, che li premia in base ai soggiorni effettuati. Si vuole che tale sistema di fedeltà sia flessibile, dunque i gestori di *CozyRooms* devono poter definire dei livelli progressivi ognuno rappresentato da un nome univoco e raggiungibile completando un certo numero minimo di prenotazioni nei due anni solari precedenti. Ad esempio, si potrebbero definire i livelli *bronzo*, *argento* e *oro*, raggiungibili completando, rispettivamente, 3, 10 e 20 prenotazioni negli ultimi due anni solari.

I gestori delle strutture possono prevedere (ma non è obbligatorio), per ogni struttura, un tasso di sconto differente per ogni livello di fedeltà. Tale tasso di sconto è definito tramite un numero reale compreso tra 0 e 1 (esclusi) e si applica al costo totale della prenotazione.

Il sistema deve permettere ai propri utenti di registrare strutture, visualizzare le strutture ricettive in una certa città, controllare la disponibilità di una sistemazione in un certo intervallo di date,

prenotare una o più sistemazioni presso una struttura, conoscere il proprio livello di fedeltà, ottenere un resoconto di tutte le prenotazioni passate, future ed in corso, e calcolare il costo di un soggiorno.

Inoltre, il sistema deve permettere:

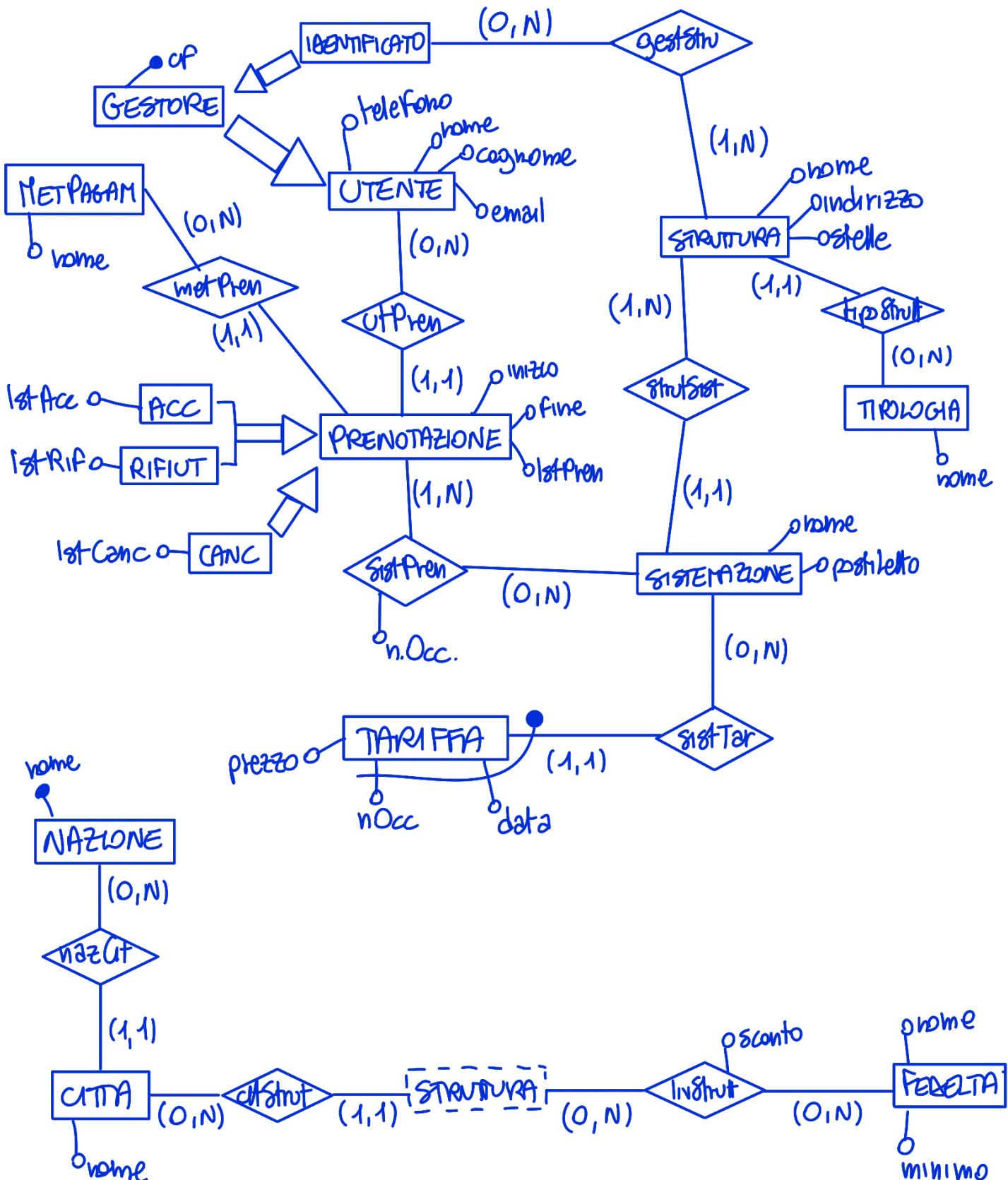
- agli utenti registrati interessati a prenotare soggiorni di ottenere le sistemazioni disponibili in un certo intervallo di date e che rispettino un certo insieme di requisiti. In particolare, le strutture restituite devono trovarsi in una data città, appartenere a una data tipologia, avere un dato numero minimo di stelle, e, per le date richieste, avere un prezzo totale (considerando il livello di fedeltà dell'utente) al di sotto di una data soglia. e il numero di posti letto richiesti
- a ogni utente gestore di strutture di calcolare, data una struttura gestita, per ognuno dei mesi dell'anno solare corrente, il numero di giorni in cui tale struttura era piena. Una struttura è considerata piena in una data se ogni sua sistemazione ha almeno una prenotazione per quella data, indipendentemente dal numero di posti letto prenotati.

**Domanda 2 (45 minuti; 75 minuti al massimo)** Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma ER concettuale per l'applicazione, il dizionario dei dati ed eventuali vincoli esterni.

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

### Diagramma ER

Produrre un diagramma ER concettuale per l'applicazione in termini di entità, relationship, attributi, relazioni is-a, generalizzazioni (disgiunte) complete e non.



**Dizionario dei dati** Per ogni entità e relationship del diagramma ER **con** attributi o vincoli:

- Definire il dominio e la molteplicità degli attributi (se diversa da (1,1))
- Definire eventuali vincoli esterni in logica del primo ordine estesa con teoria degli insiemi e semantica di mondo reale, usando il seguente alfabeto:
  - Un simbolo di predicato  $E/1$  per ogni entità  $E$ .  
Semantica di  $E(x)$ :  $x$  è una istanza di  $E$ .
  - Un simbolo di predicato  $D/1$  per ogni dominio  $D$ .  
Semantica di  $D(x)$ :  $x$  è un valore di  $D$ .
  - Un simbolo di predicato  $r/n$  ( $n > 0$ ) per ogni relationship  $n$ -aria  $r$ .  
Semantica di  $r(x_1, \dots, x_n)$ :  $x_1, \dots, x_n$  è una istanza di  $r$ .
  - Un simbolo di predicato  $a/2$  per ogni attributo  $a$  di entità  
Semantica di  $a(x, v)$ : uno dei valori dell'attributo  $a$  dell'istanza  $x$  è  $v$ .
  - Un simbolo di predicato  $a/(n+1)$  per ogni attributo  $a$  di relationship  $n$ -aria.  
Semantica di  $a(x_1, \dots, x_n, v)$ : uno dei valori dell'attr.  $a$  dell'istanza  $(x_1, \dots, x_n)$  della relat. è  $v$ .
  - Opportuni simboli di predicato (soggetti a *semantica di mondo reale*) per gestire confronti tra valori di domini numerici o comunque ordinati (tra cui  $</2$ ,  $\leq/2$ ,  $>/2$ ,  $\geq/2$ ).
  - Il predicato di uguaglianza  $=/2$  (la cui interpretazione è la relazione che lega ogni elemento del dominio di interpretazione solo con se stesso).
  - Opportuni simboli di costante (soggetti a *semantica di mondo reale*), tra cui *adesso*, interpretato come il valore del dominio DataOra che rappresenta l'istante corrente.

### Risposta

<p>[1] Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <b>UTENTE</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>attributo</th><th>dominio</th><th>moltepl. (*)</th></tr> </thead> <tbody> <tr> <td>nome</td><td>str</td><td></td></tr> <tr> <td>cognome</td><td>str</td><td></td></tr> <tr> <td>telefono</td><td>Telefono</td><td></td></tr> <tr> <td>email</td><td>Email</td><td></td></tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)	nome	str		cognome	str		telefono	Telefono		email	Email		<p>[2] Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <b>GESTORE</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>attributo</th><th>dominio</th><th>moltepl. (*)</th></tr> </thead> <tbody> <tr> <td>cf</td><td>CodFiscale</td><td></td></tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)	cf	CodFiscale	
attributo	dominio	moltepl. (*)																				
nome	str																					
cognome	str																					
telefono	Telefono																					
email	Email																					
attributo	dominio	moltepl. (*)																				
cf	CodFiscale																					

<input type="checkbox"/> 3	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	<b>PRENOTAZIONE</b>	
attributo	dominio	moltep. (*)
Fine	dataora	
Inizio	dataora	
istPren	dataora	

(\*) solo se diversa da (1,1)

Vincoli:

[V. Prenotazione.period]

$$\forall p, i, f \text{ Prenotazione}(p) \wedge \text{Inizio}(i, p) \\ \wedge \text{fine}(f, p) \rightarrow i < p$$

[V. Prenotazione.istante]

$$\forall p, i, ip \text{ Prenotazione}(p) \wedge \text{Inizio}(i, p) \\ \wedge \text{istPren}(ip, p) \rightarrow ip < i$$

<input type="checkbox"/> 5	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	<b>sistPren</b>	
attributo	dominio	moltep. (*)
numOcc	int > 0	

(\*) solo se diversa da (1,1)

Vincoli:

$$\forall p, s, l, n$$

$$\text{Prenotazione}(p) \wedge \text{Sistemazione}(s) \\ \wedge \text{sistPren}(s, p) \wedge \text{postiLetto}(l, s) \\ \wedge \text{nOcc}(n, p, l) \\ \rightarrow n \leq l$$

<input type="checkbox"/> 4	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	<b>ACCETTATA</b>	
attributo	dominio	moltep. (*)
istAcc	dataora	

(\*) solo se diversa da (1,1)

Vincoli:

[V. Accett. pren]

$$\forall p, ia, ip, i \\ \text{Accettata}(p) \wedge \text{istAcc}(ia, p) \\ \wedge \text{istPren}(ip, p) \wedge \text{Inizio}(i, p) \\ \rightarrow ip < ia \wedge ia < i$$

<input type="checkbox"/> 6	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	<b>RIFIUTATA</b>	
attributo	dominio	moltep. (*)
istRif	dataora	

(\*) solo se diversa da (1,1)

Vincoli:

[V. Rifiut. pren]

$$\forall p, if, ip, i \\ \text{Accettata}(p) \wedge \text{istRifiut}(if, p) \\ \wedge \text{istPren}(ip, p) \wedge \text{Inizio}(i, p) \\ \rightarrow ip < if \wedge if < i$$

7	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	CANCELLATA	
attributo	dominio	moltep. (*)
istCanc	dataora	
(*) solo se diversa da (1,1)		
Vincoli:		
$[V.\text{CANC.pren}]$		
$\forall p, i_c, i_p, i$		
$\text{Accettata}(p) \wedge \text{istCanc}(i_c, p)$		
$\wedge \text{istPrez}(i_p, p) \wedge \text{inizio}(i, p)$		
$\rightarrow i_p < i_c \wedge i_c < i$		

9	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	SISTEMAZIONE	
attributo	dominio	moltep. (*)
name	str	
PostiLetto	int>0	
(*) solo se diversa da (1,1)		
Vincoli:		

8	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	TARIFFA	
attributo	dominio	moltep. (*)
data	data	
numOcc	int > 0	
prezzo	denaro	
(*) solo se diversa da (1,1)		
Vincoli:		
$[V.\text{TARIFFA.posti}]$		
$\forall s, t, n, l$		
$\text{Sistemazione}(s) \wedge \text{Tariffa}(t) \wedge$		
$\text{sistTar}(t, s) \wedge \text{nOcc}(n, t) \wedge$		
$\text{postiLetto}(l, s)$		
$\rightarrow n \leq l$		

10	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	STRUTTURA	
attributo	dominio	moltep. (*)
name	str	
indirizzo	Indirizzo	
stelle	int [1,7]	
(*) solo se diversa da (1,1)		
Vincoli:		

<p>11 Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <u>InStrutt</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>sconto</td> <td>reale(0,1)</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	sconto	reale(0,1)		<p>13 Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <u>Hveloforesta</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>nome</td> <td>str</td> <td></td> </tr> <tr> <td>premMin</td> <td>Int &gt; 0</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	nome	str		premMin	Int > 0	
attributo	dominio	moltep. (*)														
sconto	reale(0,1)															
attributo	dominio	moltep. (*)														
nome	str															
premMin	Int > 0															

<p>12 Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <u>TIPOLOGIA</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>nome</td> <td>str</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	nome	str		<p>14 Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <u>CIMA</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>nome</td> <td>str</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	nome	str	
attributo	dominio	moltep. (*)											
nome	str												
attributo	dominio	moltep. (*)											
nome	str												

**[15] Tipo: Entità | Relationship (cerchiare)**

Nome: NARRATIVA

attributo	dominio	moltep. (*)
<i>name</i>	<i>str</i>	

---

(\*) solo se diversa da (1,1)

Vincoli:

**[17] Tipo: Entità | Relationship (cerchiare)**

Nome: .....

attributo	dominio	moltep. (*)

---

(\*) solo se diversa da (1,1)

Vincoli:

**[16] Tipo: Entità | Relationship (cerchiare)**

Nome: .....

attributo	dominio	moltep. (*)

---

(\*) solo se diversa da (1,1)

Vincoli:

**[18] Tipo: Entità | Relationship (cerchiare)**

Nome: .....

attributo	dominio	moltep. (*)

---

(\*) solo se diversa da (1,1)

Vincoli:

Ulteriori vincoli esterni, specifica di eventuali operazioni ausiliarie invocate da tali vincoli, e specifica dei domini concettuali non di tipo base

Domino Email  
str secondo standard

Domino Telefono

CodicePaese: str al più di 5 cifre numeriche

numero: str al più di 15 cifre ,

Domino CadFisc

stringa di 16 caratt. secondo standard

Domino Denaro

valuta: str di 3 caratteri

importo: reale  $\geq 0$

Domino Indirizzo

via: str

civico: int  $\geq 0$  (0,1)

CAP: str di 5 caratteri

[V. Prenotazione, sovrapposte]

$\forall p, p', s, i, f, i', f'$

Prenotazione(p)  $\wedge$  Prenotazione(p')  $\wedge$  inizio(i, p)  $\wedge$  fine(f, p)

$\wedge$  inizio(i', p')  $\wedge$  fine(f', p')  $\wedge$   $p \neq p'$   $\wedge$  Sistemazione(p)  $\wedge$

sistPren(s, p)  $\wedge$  sistPren(s, p')  $\wedge$   $\neg$  Cancellata(p)  $\wedge$   $\neg$  Cancellata(p')  $\wedge$

Accettata(p)  $\wedge$  Accettata(p')  $\longrightarrow$   $\nexists t$  data(t)  $\wedge$  ( $i \leq t \wedge t \leq f$ )  
 $\wedge$  ( $i' \leq t \wedge t \leq f'$ )

## Risposta alla Domanda 2 (segue)

[V. CANC. nonRifiut]

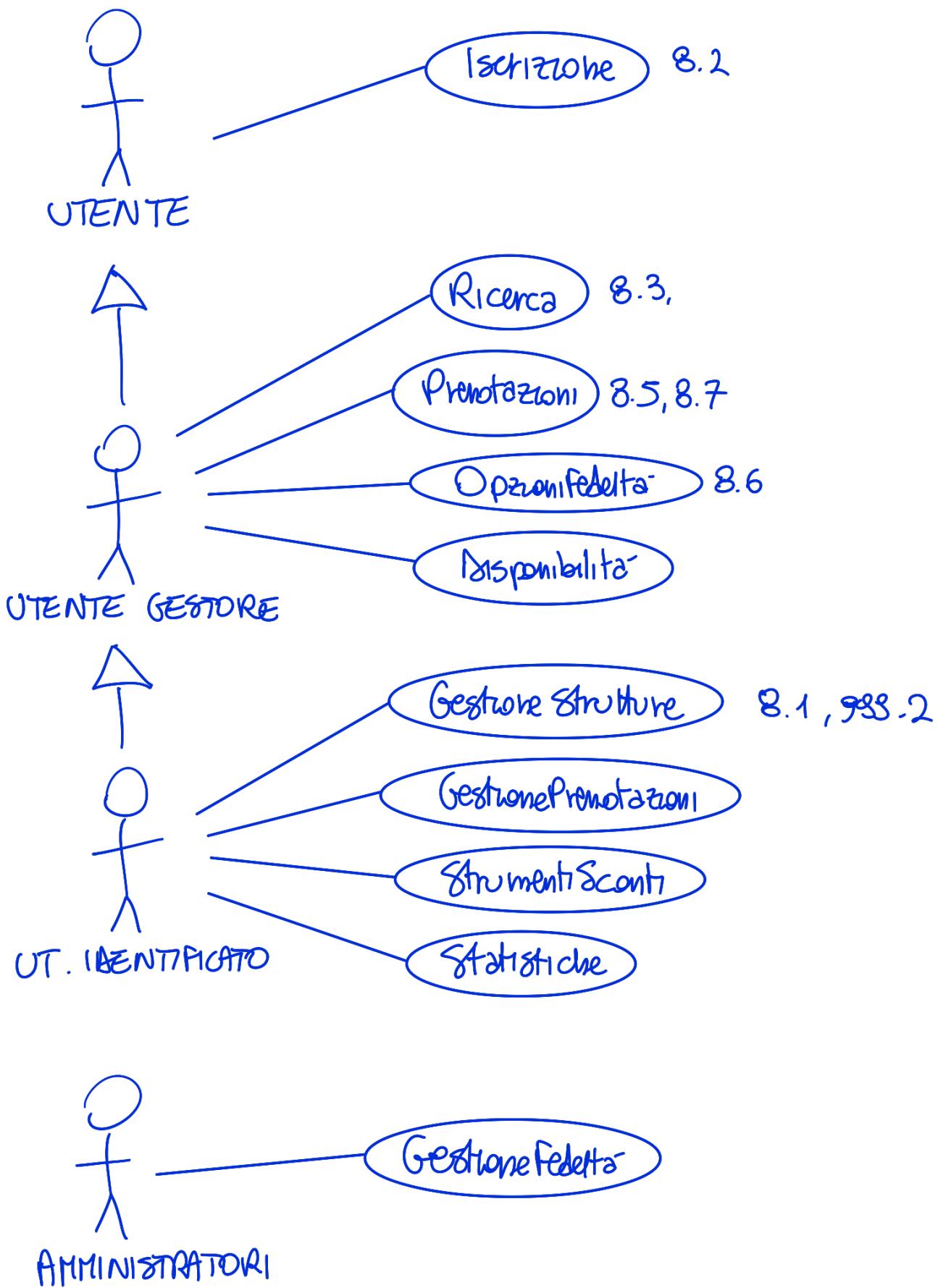
$\forall p \text{ Cancellata}(p) \rightarrow \neg \text{Rifiutata}(p)$

[V. Sist. tariffaPosti]

$\forall l,s,d \quad \text{Sistemazione}(s) \wedge \text{postiLetto}(s,l) \wedge \text{data}(d)$   
 $\rightarrow |\{x \mid \text{sistTar}(s,x) \wedge \text{data}(x,d)\}| = po$

**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 le operazioni degli use-case.

In particolare, per ogni use-case definito nella risposta alla **Domanda 3** definire la **segnatura** di tutte le operazioni che lo compongono, in termini di nome dell'operazione, nomi e dominio concettuale degli argomenti, dominio concettuale dell'eventuale valore di ritorno.

**[1] Specifica use-case:** Registrazione ..... (nome use-case)

Operazioni dello use-case:

iscriviUtente (no: str, co: str, t: Telefono, email: Email) : Utente

**[2] Specifica use-case:** Prenota ..... (nome use-case)

Operazioni dello use-case:

prenotazione (u: Utente, i: dataora, f: dataora  
(s: Sistemazione, nOcc: int > 0)(1,N) : Prenotazione

cancellaPrenotazione (p: Prenotazione) : Cancellata

**[3] Specifica use-case:** Ricerca ..... (nome use-case)

Operazioni dello use-case:

visualizzaStrutture (c: Città) : Struttura (0,N)

sistemazioniDisponibili (da: data, a: date, c: Città, t: Tipologia, st: [1,7],  
u: Utente, p: Denaro, nOcc: int > 0) : Sistemazioni (0,N)

4 Specifica use-case: ..... Disponibilità ..... (nome use-case)

Operazioni dello use-case:

controllaDisponibilità (da :data, a :data, s:Sistematizzazione) : booleano

5 Specifica use-case: ..... OttieniLivello ..... (nome use-case)

Operazioni dello use-case:

livello (u:Utente) : LivelloPefdelta

6 Specifica use-case: ..... GestioneStrutture ..... (nome use-case)

Operazioni dello use-case:

aggiungiStruttura (n: str, i:Indirizzo, st: [1,7], t:Tipologia, s:Sistematizzazione, c:Città) : Struttura

aggiungiGestore (s:Struttura, u:Identificato)

7 Specifica use-case: ..... StrumentiPrenotazione ..... (nome use-case)

Operazioni dello use-case:

RifiutaPrenotazione (p:Prenotazione) : Rifiutata

AccettaPrenotazione (p:Prenotazione) : Accettata

4 Specifica use-case: ..... *Strumenti Sconti* ..... (nome use-case)

Operazioni dello use-case:

*calcolaSconto(s:Struttura, F:LivelloDelta^-, sc:(0,1) (0,1))*

5 Specifica use-case: ..... *Statistiche* ..... (nome use-case)

Operazioni dello use-case:

*strutturaPiena(s : Struttura) : (m:int [1,12], n:int ≥ 0)*

6 Specifica use-case: ..... *Strumenti Livelli* ..... (nome use-case)

Operazioni dello use-case:

*aggiungiLivello(n:Str, n':int > 0) : LivelloDelta^-*

7 Specifica use-case: ..... (nome use-case)

Operazioni dello use-case:

**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). 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

**strutturaPiena**( $s : \text{Struttura}$ ) : ( $m : \text{int}[1, 12], n : \text{int} \geq 0$ )

precondizioni: nessuna

postcondizioni:

No side-effect.

Valore di ritorno: Siano  $di, df$  t.c.

$\exists a \ anno(\text{A}E\text{SSO}, a) \wedge data(di) \wedge data(df) \wedge anno(di, a)$   
 $\wedge anno(df, a) \wedge mese(di, 1) \wedge mese(df, 12) \wedge$   
 $\text{giorno}(di, 1) \wedge \text{giorno}(df, 31)$

$$M = \left\{ \begin{array}{l} (m, k) \\ \left| \begin{array}{l} \text{Mese}(m) \\ k = \{d \mid data(d) \wedge mese(m) \wedge (di \leq d \wedge d \leq df) \wedge \forall s' \ Sistematizzazione(s') \wedge struttura(s, s') \rightarrow \exists p, i, f \ Prenotazione(p) \wedge acettata(p) \wedge \neg cancellata(p) \wedge inizio(i, p) \wedge fine(f, p) \wedge i \leq d \wedge d \leq f \} \end{array} \right. \end{array} \right\}$$

result = M

## Risposta alla Domanda 5 (segue)

$\text{calcolaLivUtente}(u: \text{Utente}): \text{LivelloFedelta'}$

precond: nessuna

postcond: No side-effect

Val. di ritorno:

Siano  $a, a'$  t.c.  $\text{anno}(\text{adesso}, a) \wedge a' = a - 1$

$$L = \left\{ (l, n) \mid \begin{array}{l} \text{Livello}(l) \wedge \text{preMin}(l, n) \wedge \\ n \leq |\{p \mid \text{Prenotazione}(p) \wedge \text{utPren}(u, p) \wedge \\ \text{accettata}(p) \wedge \neg \text{cancellata}(p) \wedge \\ (\exists da, a, a_i, a_f \text{ inizio}(da, p) \wedge \\ \text{fine}(a, p) \wedge \text{anno}(da, a_i) \wedge \\ \text{anno}(a, a_f) \wedge a' \leq a_i \wedge a_f \leq a\}) \end{array} \right\}$$

$(l', n') \in \underset{(l, n) \in L}{\text{ArgMax}}(n)$

result =  $l'$

$\text{disponibile}(s: \text{Sistematone}, da: \text{dataora}, a: \text{dataora}): \text{Boolean}$

pre:  $da < a$

post:

No side effect

Val Ritorno:

$$P = \left\{ p \mid \begin{array}{l} \text{Prenotazione}(p) \wedge \text{accettata}(p) \wedge \neg \text{cancellata}(p) \\ \wedge (\exists da', a', t \text{ inizio}(da', p) \wedge \text{fine}(a', p) \\ \wedge \text{dataora}(t) \wedge da \leq da' \wedge da' \leq t \\ \wedge t \leq a' \wedge a \leq a') \end{array} \right\}$$

$(P = \emptyset \rightarrow \text{result} = \text{True}) \wedge (P \neq \emptyset \rightarrow \text{result} = \text{False})$

## Risposta alla Domanda 5 (segue)

*costoSistematizzazione (s: Sistematizzazione, da: data, a: data,  
f: LinFedelta, n: int > 0) : Denaro*

*pre: da < a  $\wedge (\forall d \text{ data}(d) \wedge da \leq d \wedge d \leq a)$   
 $\rightarrow \exists t \text{ Tariffa}(t) \wedge \text{data}(t, d) \wedge \text{nOcc}(t, n)$   
 $\wedge \text{sigtTar}(t, s))$*

*post:*

No side-effect

Val. di ritorno:

$$T = \left\{ (t, p) \mid \begin{array}{l} \text{Tariffa}(t) \wedge \text{prezzo}(t, p) \wedge \text{nOcc}(t, n) \wedge \\ (\exists d \text{ data}(t, d) \wedge da \leq t \wedge t \leq a) \end{array} \right\}$$

$$\left[ \begin{array}{l} \exists se, s' \text{ struttSigt}(s, s') \wedge \text{struttLin}(l, s') \wedge \text{sconto}(l, s', sc) \\ \rightarrow \text{result} = \sum_{(t, p) \in T} p \cdot (1 - se) \end{array} \right]$$

$\wedge$

$$\left[ \begin{array}{l} \nexists se, s' \text{ struttSigt}(s, s') \wedge \text{struttLin}(l, s') \wedge \text{sconto}(l, s', sc) \\ \rightarrow \text{result} = \sum_{(t, p) \in T} p \end{array} \right]$$

## Risposta alla Domanda 5 (segue)

sistemazioniDisponibili (da: data, a: date, c: Citta', t: Tipologia, st: [1,7], u: Utente, p: Denaro, nOcc: int > 0) : Sistemazioni (0, N)

precondizioni: da < a

postcondizione:

No side-effect

Valore di ritorno:

Sia l t.c.  $l = \text{calcolaLivelloUtente}(u)$

$$S = \left\{ \begin{array}{ll} S & \left| \begin{array}{l} \text{sistemazione}(s) \wedge \\ (\exists s', n', st' \text{ postIletto}(s, n') \wedge n' \geq nOcc \wedge \\ \text{struttSist}(s, s') \wedge \text{ctStrutt}(c, s') \wedge \\ \text{tipStrutt}(t, s') \wedge \\ \text{costoSistemazione}(s, da, a, l, n) \leq p \wedge \\ \text{disponibile}(s, da, a) \wedge \text{stelle}(s, st') \wedge st' \geq st \end{array} \right. \end{array} \right\}$$

## 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 ER concettuale, il dizionario dei dati e i vincoli esterni. In particolare:

- progettare una corrispondenza tra i domini concettuali ed opportuni domini SQL (domini base o utente, oppure realizzati mediante relazioni aggiuntive) supportati dal DBMS scelto
- eliminare attributi multivale o composti
- eliminare relazioni is-a e generalizzazioni
- definire un identificatore primario per ogni entità
- valutare se e come aggiungere ridondanza in maniera controllata
- 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 domini concettuali e domini supportati dal DBMS

```

create domain StringS as varchar(50);

create domain StringM as varchar(200);
create domain StringL as varchar(1000);

create domain Email as stringM check (isValid(value));

create domain TelCodPaese as char(5)
    check(value is not null and value ~'^[0-9]*$');

create domain TelefonoNuovo as char(15)
    check(value is not null and value ~'^[0-9]*$');

create type Telefono as(
    CodicePaese: TelCodPaese
    numero: TelefonoNuovo );

create domain CF as char(16) check (isValidCodfiscale(value))

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

create domain Stelle as integer check (value ≥ 1 AND value ≤ 7)

```

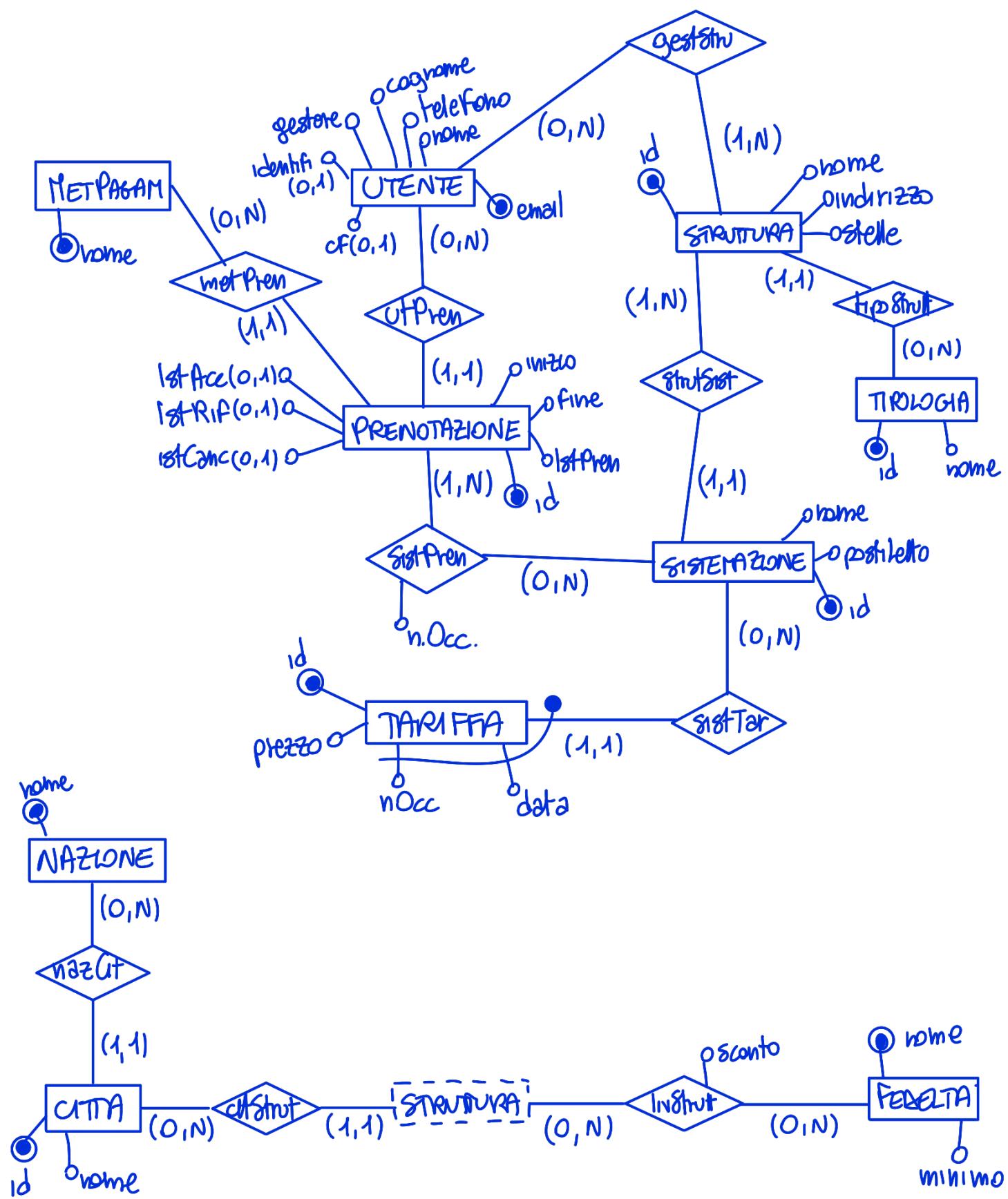
create domain RealGEZ as real check (value  $\geq 0$ )

create type Demano as (  
    valuta : char(3)  
    importo : RealGEZ);

create type Indirizzo as (  
    via : StringM  
    civico : IntegerGZ (0,1)  
    cap : char(5) );

create domain Sconto as real check (value > 0 AND value < 1)

## Diagramma ER ristrutturato



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. Prenotazione. sovrapposte]

$\forall p, p', s, i, f, i', f'$

Prenotazione( $p$ )  $\wedge$  Prenotazione( $p'$ )  $\wedge$  inizio( $i, p$ )  $\wedge$  fine( $f, p$ )  
 $\wedge$  inizio( $i', p'$ )  $\wedge$  fine( $f', p'$ )  $\wedge$   $p \neq p'$   $\wedge$  Sistemazione( $p$ )  $\wedge$   
 sistPren( $s, p$ )  $\wedge$  sistPren( $s, p'$ )  $\wedge$   $\exists i_c, i'_c \quad istCanc(i_c, p) \wedge istCanc(i'_c, p')$   
 $\left[ \exists i_a, i'_a \quad istAcc(i_a, p) \wedge istAcc(i'_a, p') \right] \rightarrow \nexists t \text{ data}(t) \wedge (i \leq t \wedge t \leq f) \wedge (i' \leq t \wedge t \leq f')$

Risposta alla Domanda 6 (segue)

[V.Utente.cfUnico]

$$\forall u, u' \text{ Utente}(u) \wedge \text{Utente}(u') \wedge u \neq u' \rightarrow (\exists c, c' \text{ cf}(c, u) \wedge \text{cf}(c', u') \rightarrow c \neq c')$$

[V.Utente.Gestore]

$$\forall u \text{ Utente}(u) \rightarrow (\exists c \text{ cf}(c, u) \leftrightarrow \text{Gestore}(u, \text{TRUE}))$$

[V.Utente.Identif]

$$\forall u \text{ Utente}(u) \rightarrow (\text{Identificato}(u, \text{TRUE}) \rightarrow \text{Gestore}(u, \text{TRUE}))$$

[V.Strutt.Gestore]

$$\forall u, s \text{ Struttura}(s) \wedge \text{gestStrutt}(s, u) \rightarrow \text{Gestore}(u, \text{TRUE})$$

[A.Pren.Hip]

$$\forall p \text{ Prenotazione}(p) \rightarrow (\exists i \text{ istFace}(i, p) \rightarrow \nexists i' \text{ istRif}(i', p))$$

**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 ER ristrutturato.

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

<b>1 Relazione</b>	<b>NAZIONE</b>	..... (nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>name</u>				
Domini	<u>StringM</u>				

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 accorda le relazioni che implementano le seguenti relationship: .....

<b>2 Relazione</b>	<b>CITTA</b>	..... (nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>name</u>	<u>id</u>	<u>nazione</u>		
Domini	<u>StringM</u>	<u>Integer</u>	<u>StringM</u>		

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):

*seriale : id*

*FK: nazione references Nazione(name)*

La relazione accorda le relazioni che implementano le seguenti relationship: .....

<b>3 Relazione</b>	<b>STRUTTURA</b>	..... (nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>id</u>	<u>name</u>	<u>Indirizzo</u>	<u>Stelle</u>	<u>tipologia</u>
Domini	<u>integer</u>	<u>StringM</u>	<u>Indirizzo</u>	<u>Stelle</u>	<u>integer</u>

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): *seriale : id*

*FK: città references Citta- (id)      inclusione : id ∈ struttSist(struttura)*

*inclusione : id ∈ gestStru(struttura)      FK: tipologia refer Tipologia(id)*

La relazione accorda le relazioni che implementano le seguenti relationship: .....

<b>4 Relazione</b>	<b>FEDERATA</b>	..... (nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>name</u>	<u>premMin</u>			
Domini	<u>StringM</u>	<u>IntegerGZ</u>			

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 accorda le relazioni che implementano le seguenti relationship: .....

<b>5 Relazione</b>	<b>LINSTRUFT</b>	..... (nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>Fedelta'</u>	<u>Struttura</u>	<u>sconto</u>		
Domini	<u>Integer</u>	<u>Integer</u>	<u>Sconto</u>		

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: fedelta refer Fedelta(id)*

*FK: struttura refer Struttura(id)*

La relazione accorda le relazioni che implementano le seguenti relationship: .....

<b>6 Relazione</b>	<b>metPagam</b>	(nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>name</u>				
Domini	<u>StringM</u>				

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 accorda le relazioni che implementano le seguenti relationship: .....

<b>7 Relazione</b>	<b>PRENOTAZIONE</b>	(nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>id</u>	initial	fine	<u>istPren</u>	utente metPagam
Domini	<u>integer</u>	timestamp	timestamp	<u>timestamp</u>	<u>Email</u> <u>StringM</u>

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): seriale: id

FK: utente refer Utente(email) inclusione: id ≤ istPren (Prenotazione)

FK: metPagam refer MetPagam(name) ennupla: istAcc ≠ NULL → istRif = NULL

La relazione accorda le relazioni che implementano le seguenti relationship: utPren, metPren

<b>8 Relazione</b>	<b>PRENOTAZIONE</b>	(nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>istCanc</u> *	<u>istAcc</u> *	<u>istRif</u> *		
Domini	timestamp	timestamp	timestamp		

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):

ennupla: istRif ≠ NULL → istPren < istRif ∧ istRif < iniZIO

ennupla: istCanc ≠ NULL → istPren < istAcc ∧ istAcc < iniZIO

ennupla: istAcc ≠ NULL → istPren < istAcc ∧ istAcc < iniZIO

La relazione accorda le relazioni che implementano le seguenti relationship: .....

<b>9 Relazione</b>	<b>gestStrutt</b>	(nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>utente</u>	<u>struttura</u>			
Domini	<u>Email</u>	<u>integer</u>			

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):

PK: utente refer Utente(email)

PK: struttura refer Struttura(id)

La relazione accorda le relazioni che implementano le seguenti relationship: .....

<b>10 Relazione</b>	<b>TIROLOGIA</b>	(nome)	Derivante da:	<b>entità</b>	<b>relationship</b> (cerchiare)
Attributi	<u>id</u>	<u>nome</u>			
Domini	<u>integer</u>	<u>StringM</u>			

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 accorda le relazioni che implementano le seguenti relationship: .....

11 Relazione <u>SISTEMAZIONE</u> (nome)	Derivante da: entità   relationship (cerchiare)
Attributi   <u>id</u>   <u>postlotto</u>   <u>name</u>   <u>struttura</u>	
Domini   <u>integer</u>   <u>integer</u>   <u>StringM</u>   <u>integer</u>	

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): seriale:id

PK: struttura refer Sistemat(id)

La relazione accorda le relazioni che implementano le seguenti relationship: strutast .....

12 Relazione <u>sistPren</u> .... (nome)	Derivante da: entità   relationship (cerchiare)
Attributi   <u>prenotaz</u>   <u>sistemaz</u>   <u>nocc</u>	
Domini   <u>integer</u>   <u>integer</u>   <u>IntegerGZ</u>	

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):

PK: prenotaz. refer Prenotazione(id)

FK:sistemaz. refer Sistemat(id)

La relazione accorda le relazioni che implementano le seguenti relationship: .....

13 Relazione <u>TARIFFE</u> .... (nome)	Derivante da: entità   relationship (cerchiare)
Attributi   <u>id</u>   <u>prezzo</u>   <u>nocc</u>   <u>data</u>   <u>sistemaz</u>	
Domini   <u>integer</u>   <u>decimal</u>   <u>IntegerGZ</u>   <u>date</u>   <u>integer</u>	

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): seriale:id

PK:sistemaz. refer Sistemat(id)

chiavi:(nOcc, data, sistemazione)

La relazione accorda le relazioni che implementano le seguenti relationship: stafTar .....

14 Relazione <u>UTENTE</u> .... (nome)	Derivante da: entità   relationship (cerchiare)
Attributi   <u>email</u>   <u>cf*</u>   <u>name</u>   <u>cognome</u>   <u>gestore</u>   <u>identific*</u>   <u>telefono</u>	
Domini   <u>Email</u>   <u>CodFisc</u>   <u>StringM</u>   <u>StringM</u>   <u>boolean</u>   <u>boolean</u>   <u>Telefono</u>	

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):

ennupla: GESTORE = TRUE  $\leftrightarrow$  CF ≠ NULL

ennupla: IDENTIFICATO ≠ NULL  $\rightarrow$  GESTORE = TRUE

La relazione accorda le relazioni che implementano le seguenti relationship: .....

15 Relazione ..... (nome)	Derivante da: entità   relationship (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 accorda le relazioni che implementano le seguenti relationship: .....

### 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. SistPren. disgiunte]

- inserimento in SistPren
- pre-operazione

`isValid = (not exist (select *`

```
from Prenotazione p, Prenotazione p1, SistPren sp
where sp.pren = p.id and p1.id = new.pren
and p.istAcc is not null
and p1.istAcc is not null
and p.istCanc is null and p1.istCanc is null
and sp.sist = new.sist
and (p.inizio, p.fine) overlaps (p1.inizio, p1.fine)
```

`if isValid: commit`

`else: genera errore`

### [T. Utente.cfUnico]

- inserimento in Utente
- pre-operazione

`isValid = (not exists (select *`

```
from Utente u
where u.cf = new.cf
and new.cf is not null
and u.cf is not null))
```

`if isValid: commit`

`else: genera errore`

## Risposta alla Domanda 7 (segue)

## [T. Struttura.gestore]

- inserimento in GestStrut
- pre-operazione

isValid = (exists (select \*

from Utente u

where u.email = newv.utente and u.gestore = TRUE

if isValid: commit

else: genera errore

## [T. Sistemazione.postiLotto]

- inserimento in sistPren

- pre-operazione

isValid = (exists (select \*

from Sistemazione s

where newv.strutt = s.id and newv.nOcc <= s.postiLotto

if isValid: commit

else: genera errore

## [T. Sistemazione.Struttura]

- inserimento in sistPren

- pre-operazione

isValid = (exists (select \*

from sistPren sp, Sistemazione s, Sistemazione s'

where sp.pren = newv.pren and s.id = sp.sist

and s'.id = newv.sist and s.strutt <> s'.strutt

if isValid: commit

else: genera errore

**Domanda 8 (30 minuti; 45 minuti al massimo)** Proseguire la fase di progettazione dell'applicazione producendo le specifiche realizzative delle operazioni di 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.

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

### Risposta

numGiorniPieni (s: integer) : (<m: IntegerGZ, n: IntegerGZ>)

TmpDate(d: date)

```
with recursive genDate(d) as(
    select date_trunc('year', 'CURRENT_TIMESTAMP')
    union all
    select d + '1 day' :: interval
    from genDate
    where d < date_trunc('year', 'CURRENT_TIMESTAMP' + '1 year' :: interval) +
        - '1 day' :: interval)
```

insert into TmpDate(n)

select d from genDate

Q = (select extract ('MONTH', t.d) as m, count(t.d)

from TmpDate t

where not exists (select \* from Sistematizzazione s' where s'.strutt = :s  
except

select \*

from Sistematizzazione s', sistPren sp, Prenotazione p

where s'.strutt = :s and sp.strutt = s'.id and

sp.pren = p.id and p.istAcc is not null and

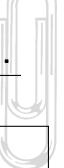
p.istCanc is null and p.inizio <= t.d

and t.d <= p.fine

group by m

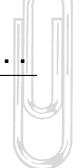
Return Q

[continua alla pagina seguente]



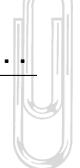
**Risposta alla Domanda 8 (segue)**

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.]