

2

Specifiche dei Requisiti

Gli utenti di *TravelPlan* utilizzano il sistema per organizzare viaggi ed unirsi a viaggi organizzati da altri utenti.

Di ogni utente interessa conoscere il nome, il cognome, l'indirizzo email, la città di provenienza e la data di iscrizione.

Ogni utente può creare viaggi in qualità di organizzatore dei viaggi interessa il nome, il numero minimo e il numero massimo di partecipanti. Gli utenti di *TravelPlan* possono quindi prendere parte ai viaggi organizzati dagli altri utenti.

Un viaggio è associato ad un certo numero di attività quali visite, pasti, tour, trasporti e pernottamenti. Di ogni attività interessa conoscere il nome, l'istante di inizio, la durata, il prezzo, il luogo in cui si tiene, delle informazioni testuali ed eventualmente un insieme di codici identificativi di biglietti/prenotazioni relativi all'attività.

Di ogni luogo interessa conoscere l'indirizzo, la città, la regione e la nazione.

Per facilitare la pianificazione, il sistema deve permettere di creare attività composte, cioè che raggruppano più attività semplici. Ad esempio, l'attività "tour gastronomico del centro storico" includerà la visita di due piazze, un percorso a piedi, un pasto al ristorante e la visita di una cantina.

Il sistema deve poter rappresentare gli spostamenti (con qualsiasi mezzo: treno, aereo, nave, bicicletta, a piedi, ecc.) come attività, delle quali è di interesse conoscere sia il luogo di partenza che quello di arrivo.

I pernottamenti sono attività particolari, poiché il sistema non deve poter rappresentare più di un pernottamento al giorno per ogni utente che partecipa a un viaggio.

Poiché può succedere che gli utenti che partecipano a uno stesso viaggio non prendano tutti parte alle stesse attività (p.es., due sotto-gruppi possono alloggiare in hotel diversi), il sistema deve permettere di specificare, per ogni attività inclusa in un viaggio, gli utenti che vi partecipano. Se per una certa attività non è specificato nessun utente, si assume che tutti i partecipanti al viaggio vi prendano parte.

I partecipanti ad un viaggio possono dare un feedback all'organizzazione del viaggio, assegnando un voto da 1 a 5.

 $H \leq 3$ $p = 0$

else

 $H \geq 4$ $p = \lfloor 10\% \text{ del num. Feedback} \rfloor$

TravelPlan prevede un sistema di punteggi per i propri utenti, in base alle valutazioni ricevute sui viaggi da loro organizzati. Il punteggio p di un utente è così calcolato: p è uguale a 0 se la media aritmetica dei voti ricevuti è ≤ 3 , altrimenti p è uguale alla parte intera inferiore del 10% del numero dei feedback ricevuti con voto ≥ 4 .

Il sistema deve permettere agli utenti registrati di partecipare a viaggi e di creare viaggi in qualità di organizzatori, aggiungendo a essi attività di vario tipo.

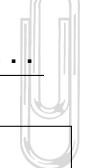
Infine, il sistema deve offrire le seguenti funzionalità avanzate:

1. Gli utenti registrati devono poter:
 - 1.1. creare viaggi in qualità di organizzatori e aggiungere a essi attività di vario tipo;
 - 1.2. trovare tutti i viaggi che includono una certa destinazione in un certo intervallo di tempo;
 - 1.3. trovare la/e città toccata/e dal maggior numero di viaggi in un certo intervallo di tempo;
 - 1.4. calcolare, per ogni regione di una data nazione, il numero di viaggi organizzati in un dato periodo di tempo che toccano quella regione;
 - 1.5. dato un budget minimo, un budget massimo, un insieme di regioni, un periodo di tempo e un punteggio, calcolare l'insieme dei viaggi che hanno un budget nell'intervallo richiesto, toccano almeno una delle regioni date, si svolgono completamente all'interno del periodo dato e sono organizzati da un utente con un punteggio pari o superiore a quello dato.
2. I responsabili del sistema devono poter calcolare, data una città di destinazione, per ognuno dei 12 mesi dell'anno, il numero di viaggi organizzati in quel mese nell'ultimo anno solare.

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



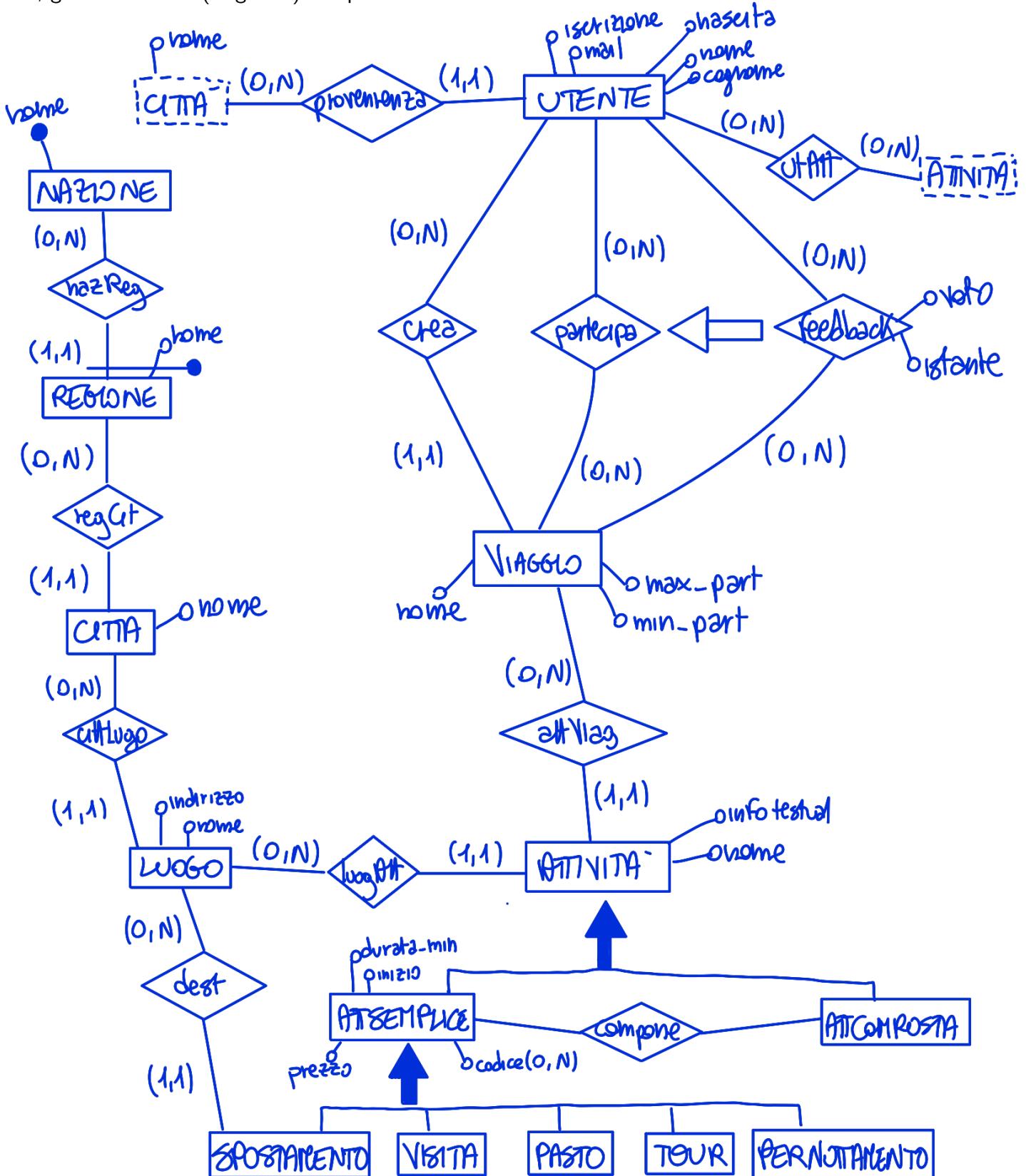
Risposta alla Domanda 1 (segue)

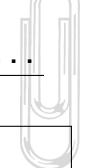
Domanda 2 (45 minuti; 75 minuti al massimo) Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma ER concettuale per l'applicazione, il dizionario dei dati ed eventuali vincoli esterni.

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.





Risposta alla Domanda 2 (segue)

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: Entità Relationship (cerchiare)</p> <p>Nome: <u>Nazione</u></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <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> <hr/> <p><u>Regno</u></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>attributo</th> <th>dominio</th> </tr> </thead> <tbody> <tr> <td>nome</td> <td>str</td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	nome	str		attributo	dominio	nome	str	<p>[2] Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>AttComposta</u></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> </table> <hr/> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> <p>$\exists N. \forall AttComp. Viaggio]$ $\forall ac, as, v. AttComp(ac) \wedge$ $composta(as, ac) \wedge attViaggio(v, ac)$ $\rightarrow attViaggio(v, as)$</p> <p>$[N. \forall AttComp. Utente]$ $\forall ac, as, u. AttComp(ac) \wedge$ $composta(as, ac) \wedge utAtt(u, ac)$ $\rightarrow utAtt(u, as)$</p>	attributo	dominio	moltep. (*)
attributo	dominio	moltep. (*)												
nome	str													
attributo	dominio													
nome	str													
attributo	dominio	moltep. (*)												

<p><input type="checkbox"/> Tipo: Entità Relationship (cerchiare)</p> <p>Nome: CITTÀ.....</p> <table border="1" data-bbox="92 224 794 291"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>str</td> <td></td> </tr> </tbody> </table> <hr/> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> <p>[V. Città. Spostamento]</p> $\forall s, l, l' \ Spostamento(s) \wedge \\ luogAtt(s, l) \rightarrow \neg dest(s, l')$	attributo	dominio	moltep. (*)	name	str		<p><input type="checkbox"/> Tipo: Entità Relationship (cerchiare)</p> <p>Nome: LUOGO.....</p> <table border="1" data-bbox="794 224 1511 291"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>str</td> <td></td> </tr> <tr> <td>indirizzo</td> <td>Indirizzo</td> <td></td> </tr> </tbody> </table> <hr/> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> <p>[V. Luogo. Spostamento]</p> $\forall s, l, l' \ Spostamento(s) \wedge \\ luogAtt(s, l) \rightarrow \neg dest(s, l')$	attributo	dominio	moltep. (*)	name	str		indirizzo	Indirizzo	
attributo	dominio	moltep. (*)														
name	str															
attributo	dominio	moltep. (*)														
name	str															
indirizzo	Indirizzo															

<p><input type="checkbox"/> Tipo: Entità Relationship (cerchiare)</p> <p>Nome: ATTIVITÀ.....</p> <table border="1" data-bbox="82 1280 794 1347"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>info test</td> <td>str</td> <td></td> </tr> <tr> <td>name</td> <td>str</td> <td></td> </tr> </tbody> </table> <hr/> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> <p>[V. Attività. utente]</p> $\forall u, v, a \ Attivita(a) \wedge utAtt(u, a) \\ \wedge attVtag(a, v) \rightarrow partecipa(u, v)$	attributo	dominio	moltep. (*)	info test	str		name	str		<p><input type="checkbox"/> Tipo: Entità Relationship (cerchiare)</p> <p>Nome: ATTEMPLICE.....</p> <table border="1" data-bbox="794 1280 1511 1347"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>durata_min</td> <td>int > 0</td> <td></td> </tr> <tr> <td>codice</td> <td>str</td> <td>(0,N)</td> </tr> <tr> <td>name</td> <td>str</td> <td></td> </tr> <tr> <td>prezzo</td> <td>Denaro</td> <td></td> </tr> </tbody> </table> <hr/> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	durata_min	int > 0		codice	str	(0,N)	name	str		prezzo	Denaro	
attributo	dominio	moltep. (*)																							
info test	str																								
name	str																								
attributo	dominio	moltep. (*)																							
durata_min	int > 0																								
codice	str	(0,N)																							
name	str																								
prezzo	Denaro																								

7	Tipo: Entità Relationship (cerchiare)	
Nome:	VIAGGIO	
attributo	dominio	moltep. (*)
name	str	
max-posti	int > 0	
min-posti	int > 0	

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

Vincoli:

[V.Viaggio.Posti]

 $\forall v, x, y$ $Viaggio(v) \wedge \text{max_part}(y, v)$ $\wedge \text{min_part}(x, v) \rightarrow$ $x \leq \left| \left\{ x \mid \text{Utente}(u) \wedge \text{partecipa}(u, p) \right\} \right| \leq y$

9	Tipo: Entità Relationship (cerchiare)	
Nome:	feedback	
attributo	dominio	moltep. (*)
voto	int [1,5]	
istante	dataora	

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

Vincoli:

[V.feedback.Continuità]

 $\forall u, v, i$ $\text{Utente}(u) \wedge \text{feedback}(u, v) \wedge$
 $\text{istante}(i, u, v) \rightarrow \text{fineViaggio}(v) < i$

8	Tipo: Entità Relationship (cerchiare)	
Nome:	UTENTE	
attributo	dominio	moltep. (*)
iscrizione	data	
mail	str	
nasuta	data	
name	str	
cognome	str	

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

Vincoli:

[V.Utente.Nasuta]

 $\forall u, i, n \quad \text{Utente}(u) \wedge \text{nasuta}(n, u)$
 $\wedge \text{iscrizione}(i, u) \rightarrow n < i$

[V.Utente.Continuità]

 $\forall u, v, i \quad \text{Utente}(u) \wedge \text{partecipa}(v, u)$
 $\text{iscrizione}(i, u) \rightarrow \text{inizioViaggio}(v) > i$

10	Tipo: Entità Relationship (cerchiare)	
Nome:	PERNOTTAMENTO	
attributo	dominio	moltep. (*)

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

Vincoli: [V.Pernottamento.unoAlgiorno]

 $\forall u, p, p', i, i', d, d' \quad \text{Utente}(u)$
 $\wedge \text{Pernottamento}(p) \wedge \text{Pernottamento}(p')$
 $\wedge p \neq p' \wedge u \in \text{Partecipanti}(p) \wedge$
 $u \in \text{Partecipanti}(p') \wedge \text{inizio}(i, p)$
 $\wedge \text{data}(d, i) \wedge \text{inizio}(i', p')$
 $\wedge \text{data}(d', i') \rightarrow d' \neq d$

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

Dominio Indirizzo

Via: str

Civico: int > 0 (0,1)

CAP: str di 5 cifre

Dominio Denaro:

Valuta: str di 3 caratteri

Importo: reale ≥ 0

[N. Utente. feedback]

$\forall u, v, i_u, if \quad Utente(u) \wedge feedback(f, v) \wedge inserzione(i_u, u)$
 $\wedge instantefed(if, u, v) \rightarrow i_u < if$

[N. Utente. noAutoFeedback]

$\forall u, v \quad Utente(u) \wedge crea(u, v) \rightarrow \neg feedback(u, v)$

[N. AttSempl. disgiunte]

$\forall u, a, a', i, i', d, d' \quad Utente(u) \wedge AttSemplice(a) \wedge AttSemplice(a') \wedge a \neq a'$
 $\wedge inizio(a, i) \wedge inizio(a', i') \wedge durata-min(a, d) \wedge durata-min(a', d')$
 $\rightarrow \#t \text{ dataora}(t) \wedge (i \leq t \wedge t \leq i + d \cdot '1 minuto')$
 $\wedge (i' \leq t \wedge t \leq i' + d' \cdot '1 minuto')$

FineAttComposta (a: AttComp): dataora

pre:

post: $A = \{f \mid \exists a, i, d \quad AttSemplice(a) \wedge inizio(a, i) \wedge durataMin(a, d)$
 $\wedge f = i + d \cdot '1 minuto' \wedge compone(a, a')\}$
 result = max(a)

Risposta alla Domanda 2 (segue)

inizioViaggio(v:Viaggio): dataora

pre:

post:

$$A = \{ i \mid \exists a \quad \text{ATTSEMPICE}(a) \wedge \text{inizio}(a,i) \wedge \text{attViag}(a,v) \}$$

$$\text{result} = \min(A)$$

fineViaggio(v:Viaggio): dataora

pre:

post:

$$A = \{ f \mid \exists a, i, d \quad \text{ATTSEMPICE}(a) \wedge \text{inizio}(a,i) \wedge \text{durataMin}(a,d) \\ \wedge f = i + d \cdot '1\text{minuto}' \wedge \text{attViag}(a,f) \}$$

$$\text{result} = \max(A)$$

inizioAttComposta(a:AttComposta): dataora

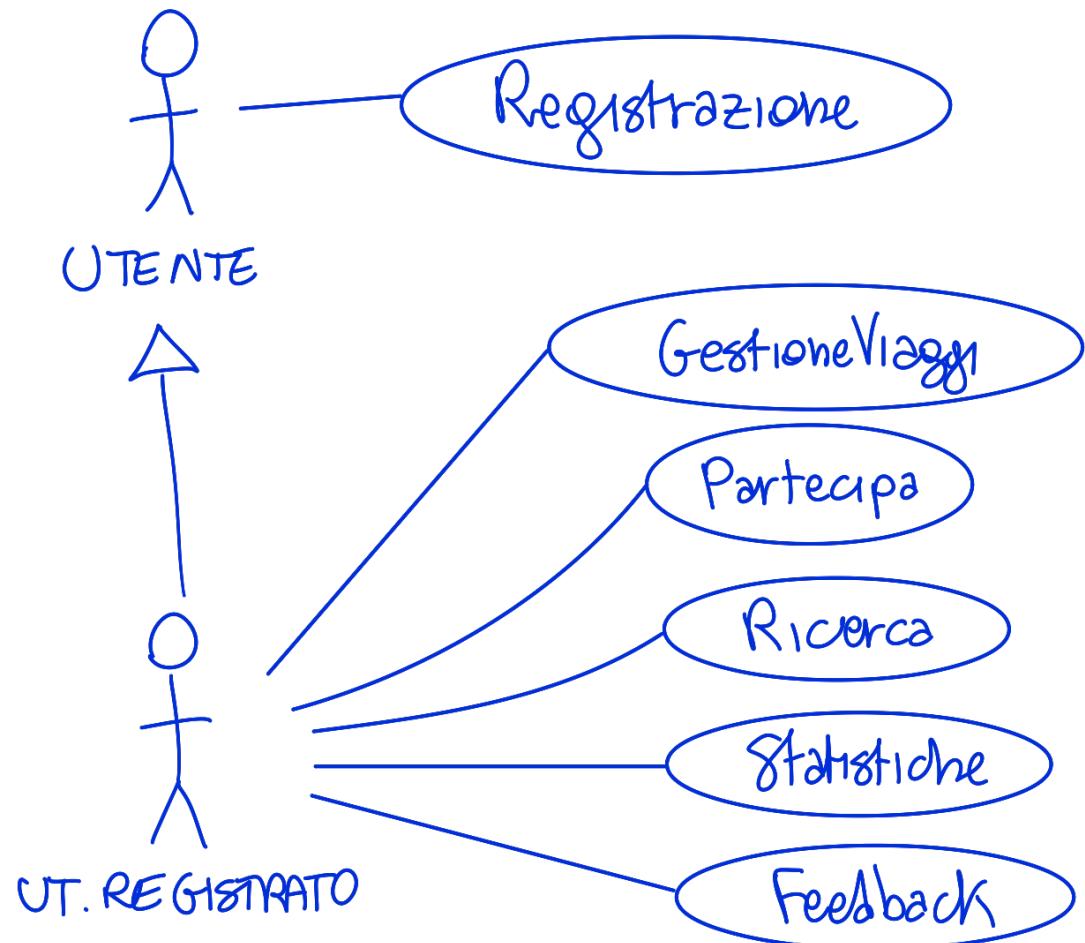
pre:

post:

$$A = \{ i \mid \exists a' \quad \text{ATTSEMPICE}(a') \wedge \text{inizio}(a',i) \wedge \text{composta}(a,a') \}$$

$$\text{result} = \min(A)$$

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:

IscrivzioneUtente(ho:str, co:str, na:data, mail>Email, d:data, c:Utente):Utente

2 Specifica use-case: GestioneViaggi (nome use-case)

Operazioni dello use-case:

creaViaggio(ho:str, u:Utente):Viaggio

creaAttComp(v:Viaggio, ho:str, i:str, as:AttSemp(1,N)):AttComp

creaAttSempl(v:Viaggio, ho:str, i:dataora, d:intenz>0, i':stringa, p:Denaro):AttSemplice

3 Specifica use-case: Partecipa (nome use-case)

Operazioni dello use-case:

partecipaViaggio(u:Utente, v:Viaggio)

partecipaAttivita'(u:Utente, a:Attivita')

partecipanti(a:Attivita'):Utente(0,N)

4 Specifica use-case: Ricerca (nome use-case)

Operazioni dello use-case:

ricercaCittà (i: dataora, f: dataora): Città(0,N)

ricercaViaggi (i: dataora, f: dataora, l: luogo): Viaggio(0,N)

ricercaViaggiBudget (b_m: Denaro, b_M: Denaro, r: Regione(0,N),
i: data, f: data, punt: intero ≥ 0): Viaggio(0,N)

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

Operazioni dello use-case:

~~X~~ calcolaPunteggio (u: Utente): intero ≥ 0

~~■~~ numViaggiRegioneNazione (n: Nazione, i: dataora, f: dataora):
(r: Regione, n': int ≥ 0)(0,N)

~~X~~ calcolaCostoViaggio (v: Viaggio): Denaro

~~■~~ numViaggiUltimoAnno (c: Città): (m: int [1,12], n: int ≥ 0)(0,N)

6 Specifica use-case: Feedback (nome use-case)

Operazioni dello use-case:

feedbackRicevuti (v: [1,5], u: Utente, v: Viaggio): intero ≥ 0

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

calcolaPunteggio($u: \text{Utente}$) : $\text{Intero} \geq 0$

precondizioni: $\exists v \text{ crea}(u, v)$

postcondizioni:

Modifica del Inv. estens. dei dati: nessuna

Valore di ritorno :

$$V = \{(u, v', v) \mid \text{Utente}(u') \wedge \text{Viaggio}(v) \wedge \text{crea}(u, v) \wedge \\ \text{partecipa}(u', v) \wedge \text{feedback}(u', v) \wedge \text{voto}(u, v', v)\}$$

$$M = \frac{\sum_{(u, v', v) \in V} v'}{|V|}$$

$$\text{result} = \begin{cases} P=0 & \text{se } M \leq 3 \\ P=\lfloor M \cdot 0,1 \rfloor & \text{se } M \geq 4 \end{cases}$$

numViaggiRegioniNazione($n: \text{Nazione}, i: \text{dataora}, f: \text{dataora}$):

precond: $i < f$

: ($r: \text{Regione}, n': \text{Int} \geq 0$) (O, N)

postcond:

$$\text{result} = \left\{ \begin{array}{l} (r, k) \mid \text{Regione}(r) \wedge \text{nazReg}(n, r) \\ \wedge K = \{v \mid \text{Viaggio}(v) \wedge i \leq \text{inizioViaggio}(v) \wedge \\ \text{fineViaggio}(v) \leq f \wedge \\ (\exists a, l, c \text{ attViag}(a, v) \wedge \text{luogoA}(a, l) \wedge \\ \text{dest}(l, a) \wedge \text{attluogo}(c, l) \wedge \text{regCit}(r, c)\} \end{array} \right\}$$

Risposta alla Domanda 5 (segue)

$\text{calcolaCostoViaggio}(v: \text{Viaggio}): \text{Denaro}$

precond: nessuna

postcond:

Rit. del lv. estens. dei dati: nessuna

Valore di ritorno:

$$A = \{(p, a) \mid \text{AttSemplice}(a) \wedge \text{prezzo}(p, a) \wedge \text{attViaggio}(v, a)\}$$

$$\text{result} = \sum_{(p, a) \in A} p$$

$\text{ricercaViaggiParametri}(\min: \text{Denaro}, \max: \text{Denaro}, r: \text{Regione}(0, N), i: \text{data}, f: \text{data}, \text{punt}: \text{intero} \geq 0): \text{Viaggio}(0, N)$

precond: $i < f$

postcond: No side effect

Valore di ritorno:

$$V = \{v \mid \text{Viaggio}(v) \wedge i \leq \text{inizioViaggio}(v) \wedge \text{fineViaggio}(v) \leq f \wedge \text{attViag}(a, v) \wedge \dots\}$$

$$\begin{aligned} K = & \text{CalcolaCostoViaggio}(v) \wedge \min \leq pr \wedge pr \leq \max \\ & \wedge \text{attViaggio}(a, v) \wedge \text{dest}(l, a) \wedge \text{crea}(u, v) \\ & \wedge \text{calcolaPunt}(u) \geq p \wedge \text{luogAtt}(l, a) \\ & \wedge \text{attLuogo}(c, l) \wedge \text{regGT}(r, c) \end{aligned}$$

$$\text{result} = V$$

Risposta alla Domanda 5 (segue)

`numViaggUltimoAnno(c: Città): (m: int [1, 12], n: int ≥ 0) (0, N)`

precond: nessuna

postcond:

No side-effect

Valore di ritorno:

$$M = \left\{ (m, n) \mid \begin{array}{l} \text{Intero}(m) \wedge m \geq 1 \wedge m \leq 12 \\ n = \{ v \mid \text{Viaggio}(v) \wedge (\exists i, f, a, d_i \\ i = \text{inizioViaggio}(v) \wedge f = \text{fineViaggio}(v) \\ \wedge (\text{adesso} - '1 anno') \leq i \wedge f \leq \text{adesso} \wedge \\ \text{data}(i, d_i) \wedge \text{data}(f, d_f) \wedge \text{mese}(d_i, m) \\ \wedge \text{attViag}(v, a) \wedge \text{luogofit}(a, l) \wedge \text{dest}(a, l) \\ \wedge \text{città}(c, l) \} \end{array} \right\}$$

result = M

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.

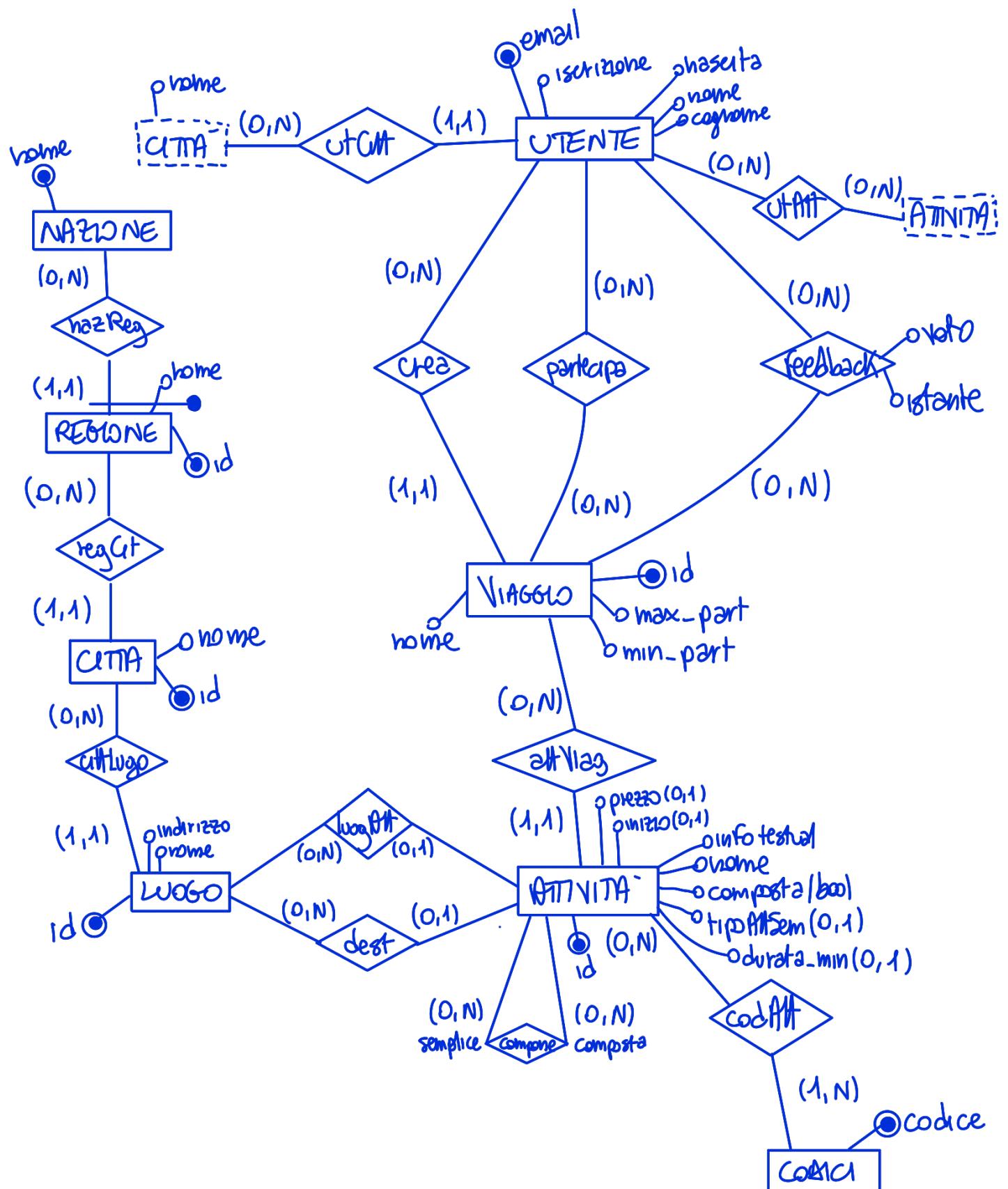
DBMS da utilizzare PostgreSQL.....
Corrispondenza tra domini concettuali e domini supportati dal DBMS

```

create domain StringS as varchar(50);
create domain StringM as varchar(200);
create domain StringL as varchar(1000);
create domain Email as StringM check( isEmail (value) );
create domain IntegerGEZ as integer check(value >= 0)
create domain IntegerGZ as integer check(value > 0)
create domain Voto as integer check( value >= 1 and value <= 5 )
create type Indirizzo as( via:StringM, civico:IntegerGEZ (0,1),
                           CAP: char(5) );
create domain RealGEZ as real check(value > 0)
create type Denaro as( valuta:char(3), importo:RealGEZ );
create type TipoAttSempl as
    enum ('SPOSTAMENTO', 'VISITA', 'PASTO', 'TOUR', 'PERNOTTAMENTO')

```

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. feedback.152]

$\forall u, t \text{ feedback}(u, t) \rightarrow \text{partecipa}(u, v)$

[V. Attività: semplice]

$\forall a \text{ Attività} \rightarrow \exists p, i, t, l, d \text{ prezzo}(a, p) \wedge \text{luogoAtt}(l, a) \wedge$
 $\text{tip}(a, t) \wedge \text{inizio}(i, a) \wedge$
 $\text{durat_min}(a, d) \Leftrightarrow \text{composta}(a, \text{false})$

[V. Attività: Composta]

$\forall a, a' \text{ Attività}(a) \wedge \text{Attività}(a') \wedge \text{componete}(\text{simp}: a, \text{comp}: a')$
 $\rightarrow \text{composta}(a, \text{true}) \wedge \text{composta}(a, \text{false})$

[V. Att. Spostamento]

$\forall a \text{ Attività}(a) \wedge \text{tip}(a, 'SPOST') \rightarrow \exists l \text{ dest}(a, l)$

Risposta alla Domanda 6 (segue)

[V. Pernottamento.unoAlgiorno]

$$\begin{aligned} \forall u, p, p', i, i', d, d' \quad & \text{Utente}(u) \wedge \text{Attivita}(p) \wedge \text{Attivita}(p') \wedge \\ & \wedge \text{tipo}(p, 'Pern') \wedge \text{tipo}(p', 'Pern') \wedge \\ & \wedge p \neq p' \wedge u \in \text{Partecipanti}(p) \wedge u \in \text{Partecipanti}(p') \wedge \text{inizio}(i, p) \\ & \wedge \text{data}(d, i) \wedge \text{inizio}(i', p') \wedge \text{data}(d', i') \rightarrow d' \neq d \end{aligned}$$

[V. luogo.Spostamento]

$$\forall s, l', l \quad \text{Attivita}(s) \wedge \text{tipo}(s, 'spost') \wedge \text{luogAtt}(l, s) \rightarrow \neg \text{dest}(l, s)$$

[V. Attivita.Codice]

$$\forall a \quad \text{Attivita}(a) \rightarrow (\exists c \quad \text{codAtt}(a, c) \rightarrow \text{composta}(a, \text{false}))$$

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.

1	Relazione <u>NAZIONE</u> ... (nome)	Derivante da: entità relationship (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:

2	Relazione <u>REGIONE</u> ... (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> <u>nazionale</u> <u>name</u>	
Domini	<u>integer</u> <u>StringM</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

chiavi: (nazionale, nome)

PK: nazionale refer Nazione(name)

La relazione accorda le relazioni che implementano le seguenti relationship:

3	Relazione <u>CITTA</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> <u>name</u> <u>regione</u>	
Domini	<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: regione refer Regione(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

4	Relazione <u>LUGO</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> <u>name</u> <u>indirizzo</u> <u>citta</u>	
Domini	<u>integer</u> <u>StringM</u> <u>Indirizzo</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: citta references Citta(id)

seriale: id

La relazione accorda le relazioni che implementano le seguenti relationship:

5	Relazione <u>ATTIVITA</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> <u>infoText</u> <u>name</u> <u>prezzo*</u> <u>inizio*</u> <u>composta</u> <u>durat-min*</u> <u>tipAtt</u>	
Domini	<u>integer</u> <u>StringL</u> <u>StringM</u> <u>Denaro</u> <u>timestamp</u> <u>boolean</u> <u>Integer62</u> <u>TipAtt</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

ennupla: TIPOATT='SPOST' → BEST≠NULL

ennupla: BEST≠NULL → BEST≠LUOGO

ennupla: COMPOSTA = FALSE ↳ PREZZO ≠ NULL ∧ INIZIO ≠ NULL ∧ TIPOATT ≠ NULL ∧ LUOGO ≠ NULL ∧ DURAT-MIN ≠ NULL

La relazione accorda le relazioni che implementano le seguenti relationship:

6 Relazione	<u>ATTIVITA'</u> ... (nome)	Derivante da:	entità	relationship (cerchiare)
Attributi	<u>luogPnt</u> * <u>dest</u> * <u>ViAGGIO</u>			
Domini	<u>integer</u> <u>integer</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: viaggio refer Viaggio(id)

PK: dest refer Luogo(id)

FK: (luogPnt) refer Luogo(id)

La relazione accorda le relazioni che implementano le seguenti relationship: luogPnt, attViag, dest.....

7 Relazione	<u>CodAtt</u> (nome)	Derivante da:	entità	relationship (cerchiare)
Attributi	<u>ATTIVITA'</u> <u>CODICE</u>			
Domini	<u>integer</u> <u>stringS</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: attivita' refer Attivita'(id)

FK: codice refer Codice(stringS)

La relazione accorda le relazioni che implementano le seguenti relationship:

8 Relazione	<u>CODICE</u> (nome)	Derivante da:	entità	relationship (cerchiare)
Attributi	<u>codice</u>			
Domini	<u>stringS</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

Inclusione: codice ⊆ CodAtt(codice)

La relazione accorda le relazioni che implementano le seguenti relationship:

9 Relazione	<u>VIAGGIO</u> (nome)	Derivante da:	entità	relationship (cerchiare)
Attributi	<u>id</u> nome max-part min-part utente			
Domini	<u>integer</u> <u>stringM</u> <u>integerGZ</u> <u>integerGZ</u> <u>Email</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: utente refer Utente(email)

seriale: id

La relazione accorda le relazioni che implementano le seguenti relationship:

10 Relazione	<u>UTENTE</u> (nome)	Derivante da:	entità	relationship (cerchiare)
Attributi	<u>email</u> iscrizione nascita nome cognome citta			
Domini	<u>Email</u> <u>date</u> <u>date</u> <u>stringM</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):

FK: citta refer Citta(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

11 Relazione <u>Partecipa</u> ... (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>utente</u> <u>viaggio</u>	
Domini <u>integer</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: viaggio refer Viaggio(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

12 Relazione <u>feedback</u> ... (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>utente</u> <u>viaggio</u> <u>voto</u> <u>istante</u>	
Domini <u>integer</u> <u>integer</u> <u>Voto</u> <u>timestamp</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: (utente, viaggio) refer Partecipa(utente, viaggio)

La relazione accorda le relazioni che implementano le seguenti relationship:

13 Relazione <u>attività</u> ... (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>utente</u> <u>attività</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: attività refer Attività(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

14 Relazione <u>Compon</u> ... (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>composta</u> <u>semplice</u>	
Domini <u>integer</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):

FK: composta refer Attività(id)

PK: semplice refer Attività(id)

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 ennupple); (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.

- revolve update on Attivita' from User

[T. Pernottamento.choAlGiorno]

- inserimento in Attivita'
- post-operazione

```
isError = exists (select u.email
                  from Attivita a, Utente u
                  where new.tipo = 'Pern' and a.tipo = 'Pern'
                    and u.email in DB.partecipanti(new.id)
                    and u.email in DB.partecipanti(a.id)
                    and extract(date from new.inizio) =
                      = extract(date from a.inizio))
```

if isError: revert
genera errore

- Revolve update on Componete from Utente

[T. attivita'.composta]

- inserimento in Componete
- pre-operazione

```
isValid = (exists (select * from Attivita a, Attivita a'
                   where new.composta = a.id
                     and new.semplice = a'.id
                     and a.composta = true and a'.composta = false))
```

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

$\text{DB.calcolaPunteggio}(\text{v: email}) : \text{IntegerGEZ}$

$Q = (\text{select avg(f.voto) as m,}$
 $\text{floor}(0.1 * \text{sum(case when f.voto} \geq 4 \text{ then 1 else 0 end)} \text{ as r}$
 $\text{from Viaggio v, feedBack f}$
 $\text{where v.utente} = :v \text{ and f.viaggio} = v.\text{id}}$

if $Q \leq 3$: return 0
else return $Q.r$

$\text{DB.mizioViaggio}(\text{v: integer}) : \text{timestamp}$

$Q = (\text{select min(a.mizio) as i}$
 from Attivita a
 $\text{where a.composta} = \text{false and a.viaggio} = :v$

return $Q.i$

$\text{DB.mizioViaggio}(\text{v: integer}) : \text{timestamp}$

$Q = (\text{select min(a.mizio + a.durata * '1 minute::interval') as f}$
 from Attivita a
 $\text{where a.composta} = \text{false and a.viaggio} = :v$

return $Q.f$

Risposta alla Domanda 8 (segue)

ricercaViaggi Parametri ($R: \text{Insieme}(<r:\text{integer}>)$, $i: \text{timestamp}$, $f: \text{timestamp}$,
 $b_{\min}: \text{Numero}$, $b_{\max}: \text{Numero}$,
 $p: \text{IntegerGEZ} : \text{Insieme}(<v:\text{integer}>)$)

if ($:f \geq :i \wedge :b_{\min} \geq :b_{\max}$): return error

TmpViaggi(v:integer)

PK: v refer Viaggio(id)

Insert into Viaggi(v)

(select v.id

from Viaggi v, Attivita a, Luogo l, Citta c

where :i ≤ DB.inizioViaggio(v.id) and DB.fineViaggio(v.id) ≤ :f

and a.viaggio=v.id and a.luogo=l.id and a.dest=l.id

and l.citta' = c.id and c.regione in (:R))

$Q = (\text{select } v$

from (select v.v as v,

sum(a.prezzo) as b,

DB.calcolaPunteggio(v.utente) as p'

from ViaggiRegioni v, Attivita a

group by v.v

having count :min ≤ b and b <= :max and p' ≥ :p)

return Q

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.]

numViaggiRegioneNazione (n:StringM, i:Timestamp, f:Timestamp)
:InSteme(<r:integer, n:IntegerGEZ>)

if :f > :i :return errore

Q = select r.id, count(v.id)

from Regione r, Attivita a, Luogo l, Citta c, Viaggio v

where :i ≤ DB.inizioViaggio(v.id) and :f ≥ DB.inizioViaggio(v.id)

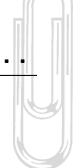
and r.nazione = :n and a.Viaggio = v.id

and l.citta = c.id and (a.dest = l.id ∨ a.luogo = l.id)

and c.regione = r.id

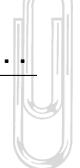
group by r.id

return Q



[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.]