



Sapienza Università di Roma  
Facoltà di Ing. dell'Informazione, Informatica e Statistica, Laurea in Informatica  
Insegnamento di **Basi di Dati, Modulo 2**  
Prof. Toni Mancini  
Dipartimento di Informatica  
<http://tmancini.di.uniroma1.it>

Esame **BD2.Esame.Risposte.ER** – Modulo risposte prova scritta

**Dati dello studente e dell'esame**

Cognome e nome: ..... Matricola: .....

Data: .....

Corso di laurea e canale di appartenenza:

- ☐ Laurea in Informatica, canale 1 (A-L, Prof. G. Perelli)
- ☐ Laurea in Informatica, canale 2 (M-Z, Prof.ssa M. De Marsico)
- ☐ Laurea in Informatica in Modalità Teledidattica Unitelma Sapienza

Firma di un membro della Commissione per  
avvenuta identificazione:

.....

**Rinuncia alla prova**



Desidero rinunciare a questa prova d'esame.

Firma: .....



Questo modulo è ottimizzato per la stampa fronte-retro

# 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

- 1.1 nome
- 1.2 cognome
- 1.3 email
- 1.4 città di provenienza
- 1.5 data iscrizione

### 2. Viaggio

- 2.1 nome
- 2.2 creatore
- 2.3 min partecipanti
- 2.4 max partecipanti
- 2.5 partecipanti
- 2.6 attività
- 2.7 budget()
  - somma prezzi delle attività
- 2.8 inizio()
- 2.9 fine()

→ e Feedback (1..5)

### 3. Attività

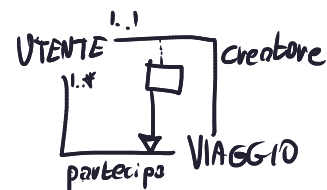
- 3.1 nome
- 3.2 istante inizio
- 3.3 durata in minuti
- 3.4 prezzo
- 3.5 info testuali
- 3.6 utente che vi partecipa (eventuale biglietto)
- 3.7 Luogo
  - 3.7.1 indirizzo
  - 3.7.2 città
  - 3.7.3 regione
  - 3.7.4 nazione
- 3.8 istante\_fine(): inizio + durata
- 3.9 tipo speciale? → {dis}

### 4. Attività composte

- 4.1 insieme di attività
- 4.2 prezzo\_tot()

### 3.9.2 Spostamento

- 3.9.2.1 luogo arrivo
- 3.9.2.2 mezzo
- 3.9.3 Prenotamento

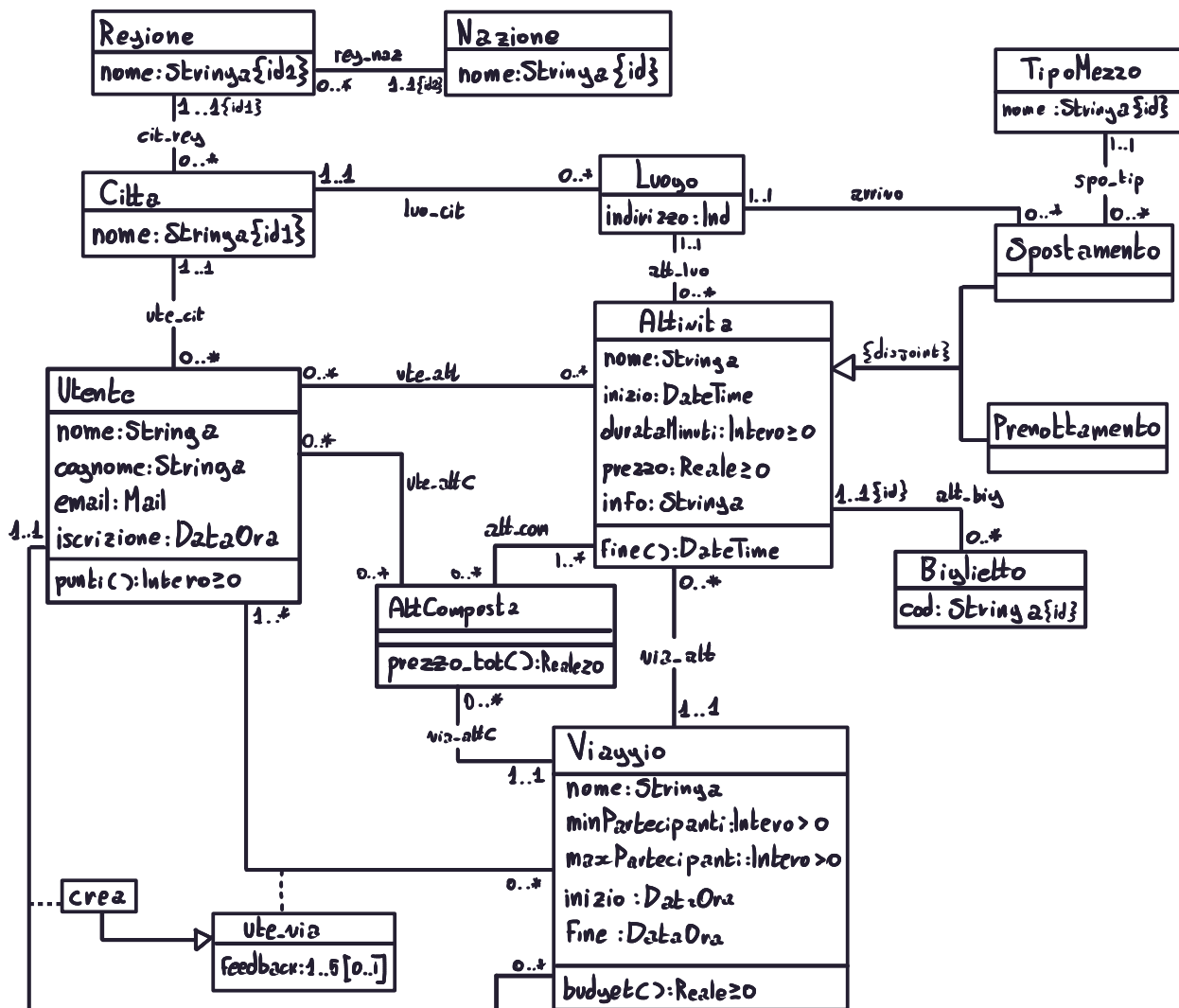


**Domanda 2 (45 minuti; 75 minuti al massimo)** Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma ER concettuale per l'applicazione, il dizionario dei dati ed eventuali vincoli esterni.

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

## Diagramma ER

Prodotto 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

<div>1</div> <div>Tipo: <u>Entità</u>   Relationship (cerchiare)</div> <div>Nome: <u>Utente</u> .....</div> <div><table><tr><td>attributo</td><td>dominio</td><td>moltepl. (*)</td></tr><tr><td colspan="3"></td></tr></table></div> <div>(*) solo se diversa da (1,1)</div> <div>Vincoli:</div> <div><div>[V.no-2-viaggi]</div><div><math display="block">\forall u, v_1, v_2, i_1, i_2, f_1, f_2</math><math display="block">[Utente(u) \wedge Viaggio(v_1) \wedge Viaggio(v_2) \wedge ute\_via(u, v_1) \wedge u \neq v_2</math><math display="block">ute\_via(u, v_2) \wedge inizio(v_1, i_1) \wedge inizio(v_2, i_2) \wedge Fine(v_1, f_1)</math><math display="block">\wedge Fine(v_2, f_2)] \rightarrow [f_1 &lt; i_2 \vee f_2 &lt; i_2]</math></div></div> <div><div>[V.iscritto-poc-viaggi]</div><div><math display="block">\forall u, v, i, n</math><math display="block">[Utente(u) \wedge Viaggio(v) \wedge iscrizione(u, n) \wedge inizio(v, i)</math><math display="block">\wedge ute\_via(u, v)] \rightarrow n &lt; i</math></div></div>	attributo	dominio	moltepl. (*)				<div>2</div> <div>Tipo: <u>Entità</u>   Relationship (cerchiare)</div> <div>Nome: <u>Utente</u> .....</div> <div><table><tr><td>attributo</td><td>dominio</td><td>moltepl. (*)</td></tr><tr><td colspan="3"></td></tr></table></div> <div>(*) solo se diversa da (1,1)</div> <div>Vincoli:</div> <div><div>[V.no-auto-valutazioni]</div><div><math display="block">\forall u, v</math><math display="block">[Utente(u) \wedge crea(u, v)] \rightarrow [\neg \exists f feedback(u, v, f)]</math></div></div> <div><div>[V.partecipa-att-del-viaggio]</div><div><math display="block">\forall u, a [Utente(u) \wedge ute\_att(z, u)] \rightarrow [ \exists v Viaggio(v)</math><math display="block">ute\_via(u, v) \wedge [via\_att(v, a) \vee [\exists ac via\_att(v, ac)</math><math display="block">\wedge att\_com(z, ac)]]]</math></div></div>	attributo	dominio	moltepl. (*)			
attributo	dominio	moltepl. (*)											
attributo	dominio	moltepl. (*)											

3 Tipo: Entità | Relationship (cerchiare)Nome: Viaggio .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.inizio-poi-fine] $\forall v, i, f [Viaggio(v) \wedge inizio(v, i) \wedge fine(v, f)] \rightarrow i < f$ [V.max-maggiore-min] $\forall v, max, min [Viaggio(v) \wedge maxPartecipanti(v, max) \wedge minPartecipanti(v, min)] \rightarrow max \geq min$ [V.limite-partecipanti] $\forall v, m [Viaggio(v) \wedge maxPartecipanti(v, m)] \rightarrow |\{u | ute.via(u, v)\}| \leq m$ 5 Tipo: Entità | Relationship (cerchiare)Nome: Prenotamento .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.no-stessa-data] $\forall u, p_1, p_2, i_1, i_2, d_1, d_2$  $[p_1 \neq p_2 \wedge Utente(u) \wedge Prenotamento(p_1) \wedge Prenotamento(p_2) \wedge ute\_att(u, p_1) \wedge ute\_att(u, p_2) \wedge inizio(p_1, i_1) \wedge inizio(p_2, i_2) \wedge Data(i_1, d_1) \wedge Data(i_2, d_2)] \rightarrow d_1 \neq d_2$ 4 Tipo: Entità | Relationship (cerchiare)Nome: Viaggio .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.attività\_dentro\_tempi\_viaggio] $\forall v, iv, fv, z, i_2, f_2$  $[Viaggio(v) \wedge inizio(v, iv) \wedge fine(v, fv) \wedge Attività(z) \wedge inizio(z, i_2) \wedge fine(z, f_2) \wedge [via\_att(v, z) \vee [\exists ac via\_att(v, ac) \wedge att\_com(z, ac)]]] \rightarrow [iv \leq i_2 \wedge fv \geq f_2]$ 6 Tipo: Entità | Relationship (cerchiare)Nome: AttComposta .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.inclusione-in-viaggio] \* $\forall z, ac, v$  $[Viaggio(v) \wedge AttComposta(z) \wedge via\_att(v, z) \wedge att\_com(z, ac)] \rightarrow via\_att(v, ac)$

7 Tipo: ~~Entità~~ | Relationship (cerchiare)Nome: Viaggio .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.no\_intersezione\_attivita] $\forall V, z1, z2, i1, i2, f1, f2$  $[Viaggio(V) \wedge Attivita(z1) \wedge Attivita(z2) \wedge z1 \neq z2$  $\wedge inizio(z1, i1) \wedge inizio(z2, i2) \wedge Fine(z1, f1) \wedge Fine(z2, f2)]$  $\rightarrow [f1 < i2 \vee f2 < i1]$ 9 Tipo: ~~Entità~~ | Relationship (cerchiare)Nome: Biglietto .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.biglietti\_utenti] ✖ $\forall z [ |\{u | ute\_att(u, z) \vee \exists ac \text{ att.com}(z, ac) \wedge ute\_att(u, ac)\}|$  $\neq 0] \rightarrow [ |\{b | att\_big(z, b)\}| = 0 \vee$  $|\{b | att\_big(z, b)\}| =$  $|\{u | ute\_att(u, z) \vee \exists ac \text{ att.com}(z, ac) \wedge ute\_att(u, ac)\}| ]$ 8 Tipo: ~~Entità~~ | Relationship (cerchiare)Nome: Biglietto .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.biglietti\_utenti] ✖ $\forall z$  $[ |\{u | ute\_att(u, z) \vee \exists ac \text{ att.com}(z, ac) \wedge ute\_att(u, ac)\}| = 0 ]$  $\rightarrow [ |\{b | att\_big(z, b)\}| = 0 \vee$  $|\{b | att\_big(z, b)\}| =$  $|\{u | ute\_att(u, z) \vee \exists ac \text{ att.com}(z, ac) \wedge ute\_att(u, ac)\}| ]$ 

10 Tipo: Entità | Relationship (cerchiare)

Nome: .....

attributo	dominio	moltepl. (*)

(\*) 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

## OPERAZIONI DI CLASSE

### AttComposta

prezzo\_tot(): Reale  $\geq 0$

•pre-cond: nessuna

•post-cond:

$$A = \{(a, b) \mid \text{att\_com}(a, \text{this}) \wedge \text{prezzo}(a, b)\}$$

$$\text{Result} = \sum_{(a, b) \in A} b$$

### Attivita

fine(): DateTime

•pre-cond: nessuna

•post-cond: siano:

i tale che inizio(this, i)

d tale che durataMinuti(this, d)

Result = i + d //somma DataOra + Minuti(intero) data dalla semantica del mondo reale

### Viaggio

budget(): Reale  $\geq 0$

•pre-cond: nessuna

•post-cond:

$$A = \{(a, b, i) \mid \left[ \left[ \text{via\_att}(\text{this}, a) \right] \vee \left[ \exists zc \text{ AttComposta}(zc) \wedge \text{via\_att}(zc, \text{this}) \right] \right] \wedge \text{att\_com}(a, zc) \wedge \text{prezzo}(a, b) \wedge \text{inizio}(a, i) \}$$

$$\text{Result} = \sum_{(a, b, i) \in A} b$$

## Tipi di Dato

Mai = [a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,6}

Ind = (via: Stringa, civico: [0-9]+[a-z]{1,3})

Risposta alla Domanda 2 (segue)

## Utente

punti(): intero  $\geq 0$

• pre\_cond:  $\exists k, v, u_2 \text{ crea}(\text{this}, v) \wedge \text{feedback}(u_2, v, k)$

• post\_cond:

$$V = \{(u, k, v) \mid \text{crea}(\text{this}, v) \wedge \text{ute\_via}(u, v) \wedge \text{Feedback}(u, v, k)\}$$

$$\text{media} = \sum_{(u, k, v) \in V} k \cdot \frac{1}{|V|}$$

$$V' = \{(u, k, v) \in V \mid k \geq 4\}$$

$$\text{point} = \lfloor |V'| \cdot 0.1 \rfloor$$

Result e' tale che:

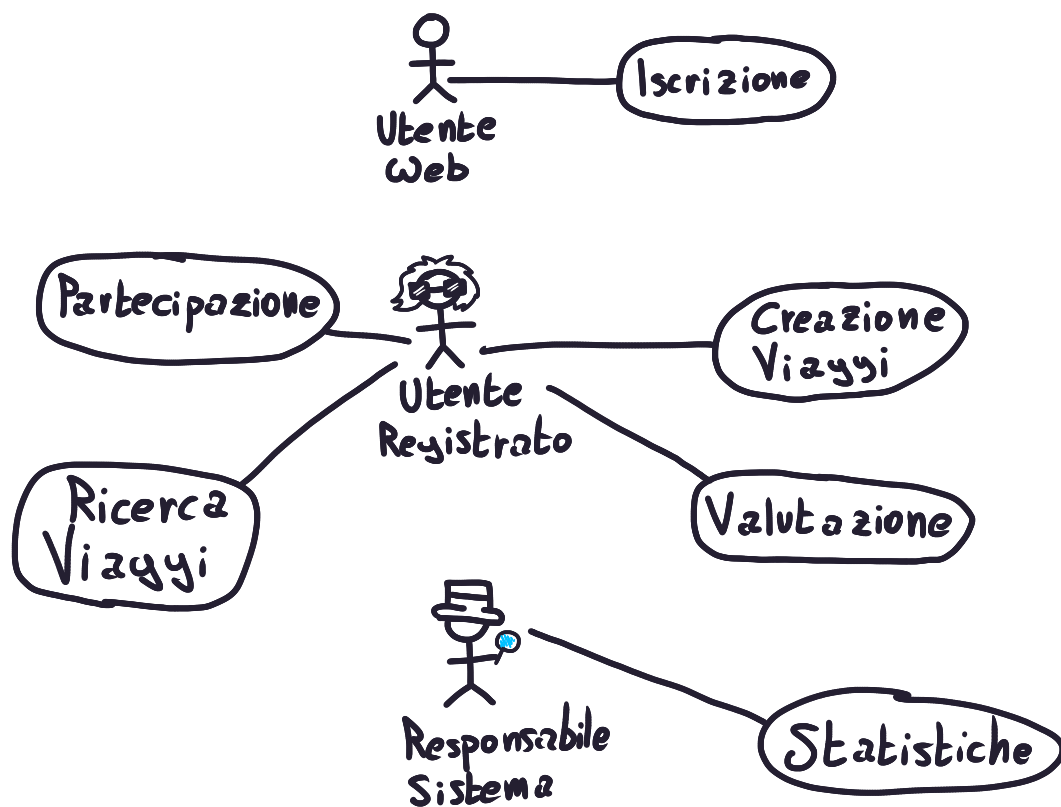
$$\text{media} \leq 3 \rightarrow \text{Result} = 0 \wedge$$

$$\text{media} > 3 \rightarrow \text{Result} = \text{point}$$



**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: Iscrizione ..... (nome use-case)

Operazioni dello use-case:

*iscriviti(nome:String, cognome:String, m:Mail):Utente*

**2** Specifica use-case: Creazione Viaggi ..... (nome use-case)

Operazioni dello use-case:

*crea(n:String, min:Intero>0, max:Intero>0, i:DataOra, f:DataOra, a:Attivita[0..\*], ac:AttComposta[0..\*]):Viaggio*

*aggiungi(v:Viaggio, a:Attivita[1..\*])*

**3** Specifica use-case: Valutazione ..... (nome use-case)

Operazioni dello use-case:

*valuta(v:Viaggio, i:1..5)*

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

Operazioni dello use-case:

*partecipa-viaggio* (*n*:Viaggio)

*partecipa-attivita* (*n*:Viaggio, *a*:Attivita)

*partecipa-attivitaCon* (*n*:Viaggio, *a*:AttComposta)

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

Operazioni dello use-case:

*viaggi\_per\_citta* (*c*:Citta): (1..12, Intero ≥ 0) [12]

6 Specifica use-case: Ricerca Viaggi ..... (nome use-case)

Operazioni dello use-case:

*cerca\_per\_dest* (*l*:luogo, *i*:DataOra, *f*:DataOra): Viaggio [0..\*]

*citta\_viaggi* (*i*:DataOra, *f*:DataOra): Citta [0..\*]

*ricerca-regione* (*i*:DataOra, *f*:DataOra, *n*:Nazione): (Regione, Intero ≥ 0) [0..\*]

*cerca-budget-regioni* (*bm*:Reale ≥ 0, *bH*:Reale ≥ 0, *R*:Regione [1..\*], *i*:DataOra, *f*:DataOra, *p*:Intero ≥ 0): Viaggio [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

**verranno considerati gli usecase degni di nota**

**cerca-budget-regioni**(bm:Reale $\geq 0$ , bH:Reale $\geq 0$ , R:Regione[1..\*], i:DataOra, f:DataOra, p:Intero $\geq 0$ ):Viaggio[0..\*]

• pre-cond:  $bm \leq bH \wedge i < f$

• post-cond: 
$$V = \left\{ v \mid \begin{array}{l} \text{Viaggio}(v) \wedge \exists b \text{ budget}(v, b) \wedge bm \leq b \leq bH \wedge \exists i, f, v \\ \text{inizio}(v, i) \wedge \text{fine}(v, f) \wedge i \geq i \wedge f \leq f \wedge \exists u \text{ crea}(u, v) \\ \wedge \exists k \text{ punti}(u, k) \wedge k \geq p \wedge \exists a [\text{via\_att}(v, a) \vee [\exists zc \\ \text{via\_attC}(v, zc) \wedge \text{att\_com}(z, zc)]] \wedge \exists l, c, v \text{ att\_lvo}(z, l) \\ \wedge \text{lvo\_cit}(l, c) \wedge \text{cit\_reg}(c, v) \wedge v \in R \end{array} \right\}$$

Result = V

**viaggi\_per\_citta**(c:Citta): (1..12, Interio $\geq 0$ )[12]

• pre-cond: nessuna

• post-cond:  $V = \{ (m, n) \mid \text{viaggi\_per\_citta\_mese}(c, m, n) \wedge 1 \leq m \leq 12 \wedge \text{Interio}(n) \}$

**viaggi\_per\_citta\_mese**(c:Citta, m:1..12): Interio $\geq 0$

• pre-cond: nessuna

• post-cond: 
$$V = \left\{ v \mid \begin{array}{l} \text{Viaggio}(v) \wedge \exists i, an, mes \text{ inizio}(v, i) \wedge \text{Anno}(i, an) \\ \wedge \exists k \text{ Anno}(\text{ADESSO}, k) \wedge k = an \wedge \text{Mese}(i, mes) \wedge \\ mes = m \wedge \exists a, l \text{ Attivita}(a) \wedge \text{att\_lvo}(a, l) \wedge \text{lvo\_cit}(l, c) \\ \wedge [\text{via\_att}(v, a) \vee [\exists zc \text{ via\_attC}(v, zc) \wedge \text{att\_com}(z, zc)]] \end{array} \right\}$$

Result = |V|

**ADESSO** è un simbolo di costante dato dalla semantica del mondo reale

**Anno** e **Mese** sono predicati dati dalla semantica del mondo reale

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 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 multivalore 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.

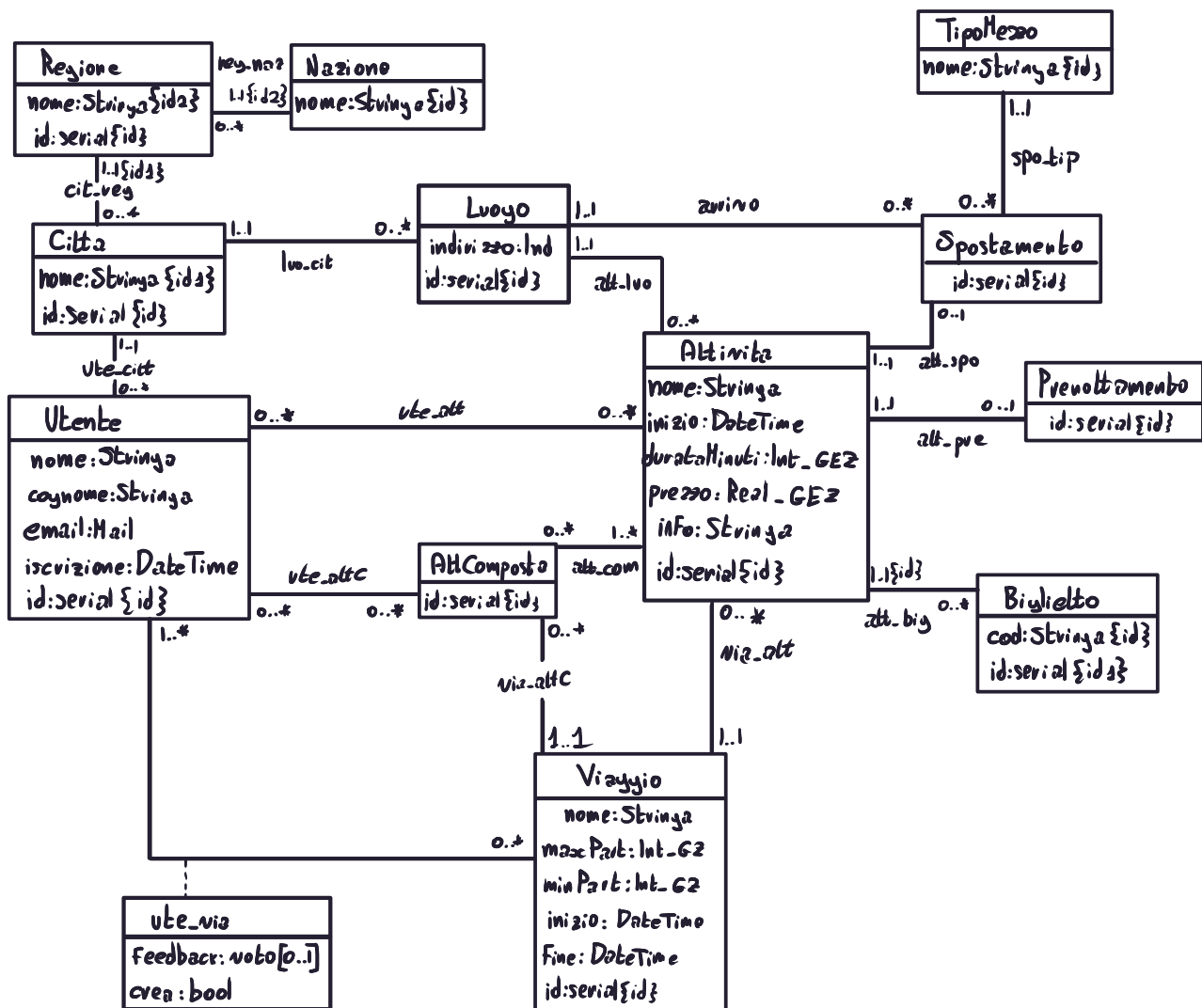
Descrivere brevemente le principali scelte effettuate.

DBMS da utilizzare PostgreSQL .....

Corrispondenza tra domini concettuali e domini supportati dal DBMS

```
create domain Mail as varchar ~ '[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,6}';
create domain String2 as varchar NOT NULL;
create domain Civ as varchar ~ '[0-9]+[a-z]{1,3}';
create type ind as (
    via: String2
    civico: Civ);
create domain Int_G2 as Integer check(value > 0);
create domain Int_GE2 as Integer check(value >= 0);
create domain Real_GE2 as Real check(value >= 0);
create domain NO60 as Integer check(value >= 1 and value <= 5);
```

## 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)

$$\forall k, u, v \ [feedback(u, v, k)] \rightarrow [\exists f \ creat(u, v, f) \wedge f = 'False']$$

$$\forall a \ [\exists p \ alt-pre(a, p) \wedge Attivita(a)] \rightarrow [\neg \exists s \ alt-spo(a, s)]$$

$$\forall a \ [\exists s \ alt-spo(a, s) \wedge Spostamento(s)] \rightarrow [\neg \exists p \ alt-pre(a, p)]$$



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 ER ristrutturato.

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

**1 Relazione Nazione..... (nome)** Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>							
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 relationship: .....

**2 Relazione Regione..... (nome)** Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>	<u>id</u>	<u>nazione</u>					
Domini	Stringa	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 nazione ref Nazione(nome);

unique(nome, nazione);

La relazione accorpa le relazioni che implementano le seguenti relationship: reg\_naz .....

**3 Relazione ...Città..... (nome)** Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>	<u>id</u>	<u>regione</u>					
Domini	Stringa	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 regione ref Regione(id);

unique(nome, regione);

La relazione accorpa le relazioni che implementano le seguenti relationship: citt-reg .....

**4 Relazione Utente..... (nome)** Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>	<u>cognome</u>	<u>email</u>	<u>iscrizione</u>	<u>id</u>	<u>città</u>		
Domini	Stringa	Stringa	Mail	DateTime	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 città ref Città(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: ute-cit .....

**5 Relazione Vizaggio..... (nome)** Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>	<u>maxPart</u>	<u>minPart</u>	<u>inizio</u>	<u>fine</u>	<u>id</u>		
Domini	Stringa	Int-GZ	Int-GZ	DateTime	DateTime	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):

check(minPart <= maxPart); check(inizio < fine);

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

6 Relazione Ute\_via ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>utente</u>	<u>viaggio</u>	<u>Feedback*</u>	<u>crea</u>				
Domini	Integer	Integer	NOLO	bool				

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

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

check((Feedback IS NOT NULL AND crea='False') OR (crea='True'));

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

7 Relazione Lvogo ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>indirizzo</u>	<u>id</u>	<u>citta</u>					
Domini	ind	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 ennuola, di dominio):

fk citta ref Citta(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: lvgo\_cit .....

8 Relazione Attivita ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>	<u>inizio</u>	<u>durataMinuti</u>	<u>prezzo</u>	<u>info</u>	<u>id</u>	<u>luogo</u>	<u>viaggio</u>
Domini	Stringa	DateTime	Int-GEZ	Real-GEZ	Stringa	serial	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 ennuola, di dominio):

fk luogo ref Luogo(id);

fk viaggio ref Viaggio(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: vie-att, att-luo .....

9 Relazione AttComposta ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>id</u>	<u>viaggio</u>						
Domini	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 ennuola, di dominio):

fk viaggio ref Viaggio(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: vie-attC .....

10 Relazione att-com ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>attC</u>	<u>att</u>						
Domini	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 ennuola, di dominio):

fk attC ref AttComposta(id);

fk att ref Attivite(id);

V.inclusione AttComposta(id) incluso in att-com(attC);

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

11 Relazione ute-alt ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>utente</u>	<u>alt</u>						
Domini	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 enupla, di dominio):

Fk alt ref Altivita(id);

Fk utente ref Utente(id);

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

12 Relazione ute-altC ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>utente</u>	<u>altC</u>						
Domini	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 enupla, di dominio):

Fk utente ref Utente(id);

Fk altC ref AltComposta(id);

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

13 Relazione Biglietto ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>cod</u>	<u>id</u>	<u>alt</u>					
Domini	Stringa	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 enupla, di dominio):

Fk alt ref Altivita(id);

Unique(alt, cod);

La relazione accorpa le relazioni che implementano le seguenti relationship: ... alt-big .....14 Relazione TipoMezzo ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>nome</u>							
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 enupla, di dominio):

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

15 Relazione Spostamento ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>id</u>	<u>luogo</u>	<u>mezzo</u>	<u>alt</u>				
Domini	serial	Integer	Stringa	Integer				

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

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

Fk luogo ref Luogo(id);

Fk mezzo ref TipoMezzo(nome); Fk alt ref Altivita(id); Unique(alt);

La relazione accorpa le relazioni che implementano le seguenti relationship: alt-spo, spo-big, arrivo

16 Relazione Prenotazione... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>alt</u>	<u>id</u>						
Domini	Integer	Serial						

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

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

unique(alt); fr alt reF Altinibo(id);

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

17 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 enupla, di dominio):

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

18 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 enupla, di dominio):

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

19 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 enupla, di dominio):

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

20 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 enupla, di dominio):

La relazione accorpa 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, enunpla, dominio, inclusione), progettare un trigger che lo implementi, definendo: (a) gli eventi da intercettare (inserimento, modifica, eliminazione di enunple); (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.

da fare i vincoli \*

Trigger.parteicipa\_att\_viaggio  
Insert o Update ute\_att

```
OK = EXISTS (SELECT *
              FROM Viaggio v, Attivita a, Utente u, ute_via uv
              WHERE new.att=a.id
              AND a.viaggio=v.id
              AND new.utente=u.id
              AND uv.utente=u.id
              AND uv.viaggio=v.id);
```

IF OK='true':permetti  
else:errore e rollback

Trigger.limite\_partecipanti  
Insert o Update ute\_via

```
WITH MP AS(SELECT maxPart FROM Viaggio v where v.id=new.viaggio),
     NMP AS(SELECT count(*) as n FROM ute_viaggio uv WHERE uv.viaggio=new.viaggio)

Error = EXISTS(SELECT * FROM NMP, MC where n > maxPart);

IF Error: rollback
else: permetti op.
```

T. attivita\_dentro\_tempi\_del\_viaggio  
Insert o Update via\_att

```
OK = EXISTS (SELECT
              FROM Attivita at, Viaggio v
              WHERE at.id=via_att.att AND v.id=via_att.viaggio
              AND at.inizio>=v.inizio
              AND fine_att(at.id)<=v.fine);
```

IF OK='true':permetti  
else:errore e rollback

Risposta alla **Domanda 7** (segue) ogni trigger sarà controllato POST operazione

T.no\_2\_viaggi

Insert o Update ute\_via

```
Error = EXISTS ( SELECT *
                  FROM ute_via u1, Viaggio v1, Viaggio v2
                  WHERE u1.utente = new.utente
                  AND v1 = u1.viaggio
                  AND v2 = new.viaggio AND v1 <> v2
                  AND (v1.inizio, v1.fine) INTERSECT (v2.inizio, v2.fine));
```

if Error: rollback

else: permetti op.

T.iscritto\_poi\_viaggia

Insert o Update ute\_via

```
Error = EXISTS ( SELECT *
                  FROM Utente u, viaggio v
                  WHERE new.utente = u.id
                  AND new.viaggio = v.id
                  AND v.inizio <= u.iscrizione);
```

if Error: rollback

else: permetti op.

T.no\_intersezione\_attivita

insert o Update Attività

```
Error = EXISTS ( SELECT *
                  FROM Attività a WHERE a.viaggio = new.viaggio
                  AND (a.inizio, fine_att(a.id)) INTERSECT (new.inizio, fine_att(new.id)));
```

if Error: rollback

else: permetti op.

T.no\_stessa\_data

Insert o Update ute\_att

```
Error = EXISTS ( SELECT *
                  FROM ute_att at, Attività a1, Prenotamento p, Attività a2, Prenotamento p2
                  WHERE at <> new AND a2.id = new.att AND p2.att = a2.id
                  AND at.utente = new.utente AND at.att = a1.id AND a1.id = p.att
                  AND (a1.inizio, fine_att(a1.id)) INTERSECT (a2.inizio, fine_att(a2.id)));
```

if Error: rollback

else: permetti op.

**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

Tabella di Regioni

*cerca\_budget\_voyoni* (bm:Real\_GEZ, bM:Real\_GEZ, R:Insieme(<Integer>), i:DateTime, f:DateTime, p:Int\_GEZ)  
:Insieme(<Integer>)

• pre-cond:  $bm \leq bM$  AND  $i < f$

• post-cond:  $V = \text{SELECT } v.id$   
 FROM Viaggio  $v$ , Attività  $a$ , Luogo  $l$ , Città  $c$ , R, ute\_via  
 WHERE  $a.viaggio = v.id$  AND  $a.luogo = l.id \wedge l.città = c.id$  AND  $citta.regione = R.id$   
 AND  $v.inizio \geq i$  AND  $v.fine \leq f$   
 AND  $ute_via.viaggio = v.id$  AND  $ute_via.crea = 'True'$   
 AND  $punti(ute_via.utenbe) \geq p$   
 AND  $budget(v.id) \geq bm$  AND  $budget(v.id) \leq bM$ ;

result = V

*viaggi\_per\_città* (C: Integer)

$V = \text{SELECT EXTRACT(MONTH FROM } v.inizio), \text{count(DISTINCT } v.id)$   
 FROM Viaggio  $v$ , Attività  $a$ , Luogo  $l$   
 WHERE  $a.viaggio = v.id$  AND  $a.luogo = l.id$   
 AND  $l.città = C$  AND  
 $\text{EXTRACT(YEAR FROM } v.inizio) = \text{EXTRACT(YEAR FROM NOW())}$   
 GROUP BY  $\text{EXTRACT(MONTH FROM } v.inizio)$ ;

Result = V



Risposta alla **Domanda 8** (segue)

create function

fine\_att( at:Integer):DateTime

```
f = (SELECT inizio + INTERVAL durataInuti::Interval
      FROM Attività WHERE id = at.id);
```

return f

Create function

budget( v: id):Real\_GE2

```
b = SELECT sum(at.prezzo)
      FROM Attività at
      WHERE at.viaggio = v;
```

return b

Create function

punti( u:Integer):Int\_GE2

```
media = WITH Q AS (SELECT v.id
                      FROM ute_via JOIN Viaggio v on v.id = ute_via.viaggio
                      AND crea='True' AND ute_via.utente = u)
          SELECT avg(feedback)
          FROM ute_via JOIN Q on Q.v.id = ute_via.viaggio
          AND crea='False' AND feedback is not null;
```

```
punti = WITH Q AS (SELECT v.id
                      FROM ute_via JOIN Viaggio v on v.id = ute_via.viaggio
                      AND crea='True' AND ute_via.utente = u)
          SELECT count(*)
          FROM ute_via JOIN Q on Q.v.id = ute_via.viaggio
          AND crea='False' AND feedback > 3;
```

```
if media <= 3 :
    return 0
```

```
else
    return floor( punti * 0.1)
```

Matricola: .....

*Minute*

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.]





Matricola: .....

*Minute*

[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.]

