

Esame Es.20230130 – Prova scritta del 30 gennaio 2023

Si vuole progettare Slimmy, una nuova applicazione che permetta agli utenti di monitorare la loro alimentazione, tracciando i cibi che ingeriscono giornalmente (con relative quantità e nutrienti) per tenere sotto controllo apporto calorico ed equilibrio della loro dieta.

A seguito di interviste con il committente, è stata stilata la seguente specifica dei requisiti.

Slimmy deve consentire agli utenti di registrarsi al servizio fornendo nome, cognome, nazionalità, email e password.

Slimmy deve fornire un archivio di schede alimenti (cibi e bevande). Tale archivio viene continuamente esteso, oltre che dalla redazione di Slimmy, anche dagli stessi utenti. Per ogni alimento presente nell'archivio, la relativa scheda deve permettere di rappresentare il nome, la marca (se disponibile), la grandezza della confezione (se disponibile), il codice identificativo (una stringa) codificato nel codice a barre della confezione (se disponibile), il relativo apporto calorico (in kCal) per unità di alimento, il valore di tale unità (ad es., per i cibi liquidi: litri; per i solidi, grammi o etogrammi o chilogrammi; per alimenti come le uova, unità), la quantità (in unità di alimento) di una porzione standard, oltre che la quantità (questa volta sempre in grammi, per garantire uniformità) di ogni nutriente per unità di alimento.

La lista dei nutrienti è tipicamente stabile ed uguale per tutti gli alimenti (ad es., carboidrati totali, zuccheri, proteine, grassi totali, grassi saturi, grassi insaturi, sodio, potassio, fibre, alcol, etc.), ma Slimmy deve fare in modo che tale lista sia estendibile dall'autore dell'inserimento di una scheda alimento. Viceversa, Slimmy deve permettere l'inserimento di schede alimenti che non hanno valori per i nutrienti definiti opzionali. La lista dei nutrienti opzionali è definita dalla redazione di Slimmy. Inoltre, tutti i nutrienti aggiunti dagli utenti alla lista dei nutrienti sono, per definizione, opzionali. Si mostrano di seguito alcuni esempi di schede alimento:

Fusilli Di Chicco (formato n. 34)

- Codice identificativo: "8033829843"
- Unità di alimento: 100 grammi
- kCal per unità di alimento (ovvero per 100 grammi): 351
- Porzione standard: 0.7 unità di alimento (ovvero 70 grammi)
- Nutrienti (in grammi per 100 grammi di prodotto; *: nutrienti obbligatori):

– Grassi totali*: 1.5 g	– Zuccheri*: 3.4 g	– Sale: 0 g
– Grassi saturi*: 0.3 g	– Fibre*: 2.9 g	
– Carboidrati*: 69 g	– Proteine*: 14 g	

Buzz Cola

- Codice identificativo: "11928322032"
- Unità di alimento: 100 ml
- kCal per unità di alimento (ovvero per 100 ml): 42
- Porzione standard: 3.3 unità di alimento (ovvero 330 ml)
- Nutrienti (in grammi per 100 ml di prodotto; *: nutrienti obbligatori):

- Grassi totali*: 0 g
- Grassi saturi*: 0 g
- Carboidrati*: 11 g
- Zuccheri*: 11 g

- Fibre*: 0 g
- Proteine*: 0 g
- Caffeina: 0.008 g
- Calcio: 0.002 g

- Fosforo: 0.01 g
- Potassio: 0.002 g
- Ferro: 0.00011 g
- Sale: 0.001 g

Come detto, schede relative a nuovi alimenti possono essere aggiunte in ogni momento dalla redazione di Slimmy, ma anche da parte degli utenti, che quindi possono contribuire in prima persona all'ampliamento dell'archivio degli alimenti di Slimmy. Mentre le schede di alimenti inserite dalla redazione diventano subito pubbliche e visibili a tutti gli utenti, le schede aggiunte da utenti devono attraversare un processo di validazione. Una volta validate diventano pubbliche esattamente come quelle inserite dalla redazione.

Una scheda alimento inserita da un utente viene resa pubblica quando almeno 10 utenti l'hanno dichiarata valida. Fintantoché una scheda non supera il processo di validazione, può ancora essere ottenuta come risultato di una ricerca, ma può essere utilizzata (ad es., inserita in un diario alimentare) solo da parte dell'utente che l'ha compilata e di quelli che l'hanno dichiarata valida fino a quel momento (in questo modo si vogliono incentivare gli utenti interessati ad inserire quell'alimento a contribuire alla validazione della relativa scheda).

Nel tempo, le schede alimenti (sia quelle inserite dalla redazione che quelle inserite dagli utenti e diventate pubbliche a seguito di validazione) possono essere invalidate (ad es., perché obsolete, si pensi al caso in cui un alimento confezionato cambi profilo nutrizionale perché prodotto con una nuova ricetta). Questo avviene quando 100 utenti le hanno dichiarate non più valide. Le schede invalidate diventano inaccessibili a tutti gli utenti (tranne che, per le schede inizialmente inserite da un utente, all'utente che le ha inserite).

Gli utenti durante la loro giornata, inseriscono in Slimmy la composizione dei loro pasti (diario alimentare), in termini di alimenti ingeriti e relative quantità. L'insieme dei pasti è attualmente definito dalla redazione di Slimmy come: colazione, pranzo, snack e cena, ma potrebbe essere oggetto di estensioni e perfino di personalizzazioni in futuro.

Il sistema deve permettere di calcolare una serie di statistiche riguardo il diario alimentare di un utente in un certo giorno. In particolare: (i) Numero totale di kCal ingerite; (ii) grammi totali ingeriti di ogni nutriente.

Gli utenti di Slimmy possono acquistare abbonamenti Premium (la cui durata dipende dalla tipologia di abbonamento scelta tra quelle previste), durante i quali avranno accesso a servizi non offerti ai clienti Free. Il servizio principale offerto da Slimmy ai clienti Premium è relativo alla assistenza di coloro che desiderano seguire una dieta.

Le diete sono definite dai singoli utenti e sono private. Una dieta definisce delle regole, in termini di soglie minime e massime per alcune delle statistiche giornaliere di cui sopra. Ad esempio, un utente Premium che vuole perdere peso può definire una dieta, che decide di chiamare "ipocalorica ed iperproteica", con le seguenti regole circa le kCal totali e le quantità di nutrienti ingeriti al giorno:

Dieta: "ipocalorica ed iperproteica"

1. kCal totali: tra 1300 e 1500
2. Carboidrati totali: tra 150 e 180 g
3. Zuccheri: ≤ 30 g
4. Proteine totali: tra 80 e 110 g
5. Grassi saturi: ≤ 10 g
6. Sale: ≤ 3 g
7. Fibre: tra 35 e 40 g

Dato un pasto p , Slimmy deve permettere agli utenti di ottenere gli alimenti più comunemente aggiunti al diario per il pasto p da tutti gli altri utenti. Per ognuno, si vuole anche conoscere la quantità media con la quale viene aggiunta al diario.

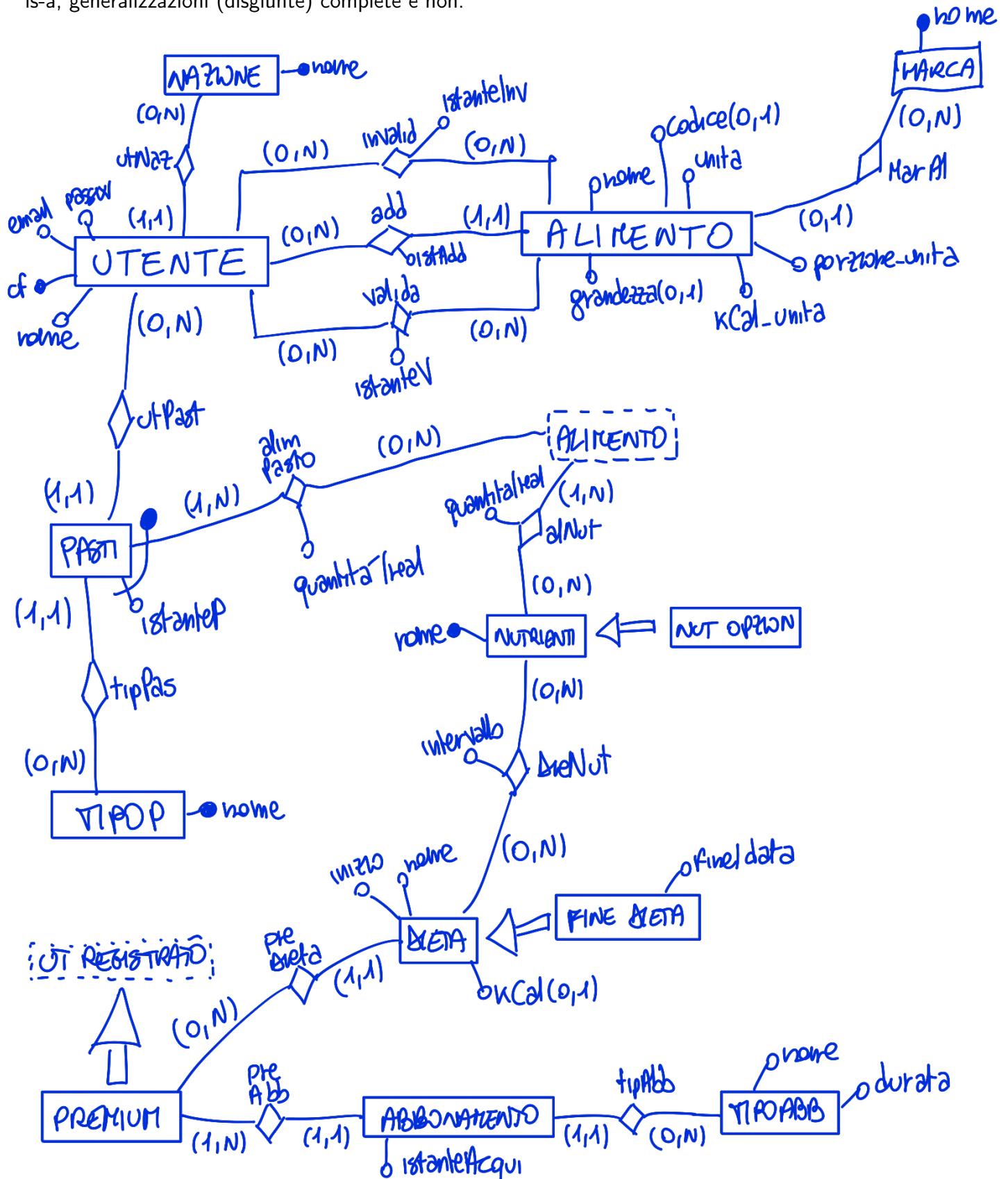
Infine, Slimmy deve restituire, all'utente Premium che sta seguendo una dieta, le quantità di ognuno dei nutrienti che può ancora assumere (in base alle regole della dieta) in un certo giorno.

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: Entità Relationship (cerchiare)</p> <p>Nome: <u>UTENTE</u></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>name</td> <td>str</td> <td></td> </tr> <tr> <td>cf</td> <td>Codice</td> <td></td> </tr> <tr> <td>passw</td> <td>Passw</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. (*)	name	str		cf	Codice		passw	Passw		email	Email		<p>[2] 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>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>str</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)	name	str	
attributo	dominio	moltepl. (*)																				
name	str																					
cf	Codice																					
passw	Passw																					
email	Email																					
attributo	dominio	moltepl. (*)																				
name	str																					

<p>3] Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>ALIMENTO</u> </p> <table border="1"> <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>grandezza</td> <td>reale > 0</td> <td>(0,1)</td> </tr> <tr> <td>Kcal-unita</td> <td>reale > 0</td> <td></td> </tr> <tr> <td>codice</td> <td>str</td> <td>(0,1)</td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)	nome	str		grandezza	reale > 0	(0,1)	Kcal-unita	reale > 0		codice	str	(0,1)	<p>5] Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>ALIMENTO</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td>porzione</td> <td>reale > 0</td> <td></td> </tr> <tr> <td>unità</td> <td>UNIT</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)	porzione	reale > 0		unità	UNIT	
attributo	dominio	moltepl. (*)																							
nome	str																								
grandezza	reale > 0	(0,1)																							
Kcal-unita	reale > 0																								
codice	str	(0,1)																							
attributo	dominio	moltepl. (*)																							
porzione	reale > 0																								
unità	UNIT																								

<p>4] Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>MARCA</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</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	moltepl. (*)	nome	str		<p>6] Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>InValid</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td>istanteInv</td> <td>dataora</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)	istanteInv	dataora	
attributo	dominio	moltepl. (*)											
nome	str												
attributo	dominio	moltepl. (*)											
istanteInv	dataora												

<p><input type="checkbox"/> 7 Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>abb</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td><u>istanteInv</u></td> <td><u>dataora</u></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	<u>istanteInv</u>	<u>dataora</u>		<p><input type="checkbox"/> 9 Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>valid</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td><u>istanteInv</u></td> <td><u>dataora</u></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	<u>istanteInv</u>	<u>dataora</u>	
attributo	dominio	moltep. (*)											
<u>istanteInv</u>	<u>dataora</u>												
attributo	dominio	moltep. (*)											
<u>istanteInv</u>	<u>dataora</u>												

<p><input type="checkbox"/> 8 Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>PASTI</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td><u>istanteP</u></td> <td><u>dataora</u></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> <p><u>[V.Pasto.Valido] [V.Pasto.InValido]</u></p> <p><u>$\forall u, p, a \ alimPasto(u, p) \wedge$</u></p> <p><u>$istanteP(i, p) \wedge utPast(u, p)$</u></p> <p><u>$\wedge add(u, a) \rightarrow valida(u, a) \leftrightarrow$</u></p> <p><u>$\exists$</u></p>	attributo	dominio	moltep. (*)	<u>istanteP</u>	<u>dataora</u>		<p><input type="checkbox"/> 10 Tipo: Entità Relationship (cerchiare)</p> <p>Nome: <u>alimPasto</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td><u>quantità</u></td> <td><u>reale > 0</u></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	<u>quantità</u>	<u>reale > 0</u>	
attributo	dominio	moltep. (*)											
<u>istanteP</u>	<u>dataora</u>												
attributo	dominio	moltep. (*)											
<u>quantità</u>	<u>reale > 0</u>												

11 Tipo: **Entità** | Relationship (cerchiare)

Nome: TIPPOP

attributo	dominio	moltepl. (*)
<u>name</u>	<u>TIPO</u>	

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

Vincoli:

13 Tipo: **Entità** | Relationship (cerchiare)

Nome: alNut

attributo	dominio	moltepl. (*)
<u>quantita</u>	<u>reale>0</u>	

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

Vincoli:

12 Tipo: **Entità** | Relationship (cerchiare)

Nome: NUTRIENTI

attributo	dominio	moltepl. (*)
<u>name</u>	<u>str</u>	

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

Vincoli:

[V. Nutriente.noOpz]

$\forall a, n \text{ Alimento}(a) \wedge \text{Nutriente}(n)$

$\wedge \neg \text{NutOpz}(a) \rightarrow \exists \text{Nut}(a, n)$

14 Tipo: **Entità** | Relationship (cerchiare)

Nome: DieNut

attributo	dominio	moltepl. (*)
<u>intervallo</u>	<u>INTER</u>	

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

Vincoli:

15	Tipo: Entità Relationship (cerchiare)	
Nome:	METÀ	
attributo	dominio	moltep. (*)
name	str	
INIZIO	dataora	
K Cal	INTER	(0,1)

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

Vincoli:

[V. Metà.durata]

$$\forall d, p \text{ Metà}(d) \wedge \text{preDte}(p, d) \wedge$$

$$\text{inizio}(i, d) \rightarrow \exists a, t, d \text{ Abbonamento}(a) \wedge$$

$$\text{preAbb}(p, a) \wedge \text{tipAbb}(t, a) \wedge$$

$$\text{durata}(d, t) \wedge d.\text{da} \leq i \wedge i \leq d.\text{a}$$

17	Tipo: Entità Relationship (cerchiare)	
Nome:	ABBONAMENTO	
attributo	dominio	moltep. (*)
istAcqui	dataora	

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

Vincoli:

[V. Abbonamento.date]

$$\forall a, ist, t, d$$

$$\text{Abbonamento}(a) \wedge \text{ristanteAcq}(ist, a) \wedge$$

$$\text{tipAbb}(t, a) \wedge \text{durata}(d, t)$$

$$\rightarrow i \leq d.\text{da}$$

16	Tipo: Entità Relationship (cerchiare)	
Nome:	TIPOABB	
attributo	dominio	moltep. (*)
name	str	
durata	Periodo	

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

Vincoli:

18	Tipo: Entità Relationship (cerchiare)	
Nome:	FINE METÀ	
attributo	dominio	moltep. (*)
fine	dataora	

(*) 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 Passw: str secondo standard

Domino CdFisc: str di 16 char secondo standard

Domino Unità: { litri, unita, grammi }

Domino TIPO { colaz, pranzo, snack, cena }

Domino Periodo

da: dataora

a: dataora

$\forall p, x, y \text{ Periodo}(p) \wedge \text{da}(x, p) \wedge \text{a}(y, p) \rightarrow x < y$

Domino Intervallo

da: reale

a: reale

$\forall i, x, y \text{ Intervallo}(i) \wedge \text{da}(x, i) \wedge \text{a}(y, i) \rightarrow x < y$

[V.Utente.alimentodate]

$\forall u, a, i_a, i_v, i_i$

Utente(u) \wedge Alimento(a) \wedge valida(u, v) \wedge istanteVal(i_v, a, u)

\wedge invalida(u, v) \wedge istanteInv(i_i, a, u) \rightarrow i_i < i_v

[V.dieta.date]

$\forall d, i, f \quad \text{FineDieta}(d) \wedge \text{inizio}(i, d) \wedge \text{fine}(f, d) \rightarrow i < f$

Risposta alla Domanda 2 (segue)

[V. Dieta. disgiunto]

$$\begin{aligned} \forall u, d, d', i, i', f, f' \quad & \text{Premium}(u) \wedge \text{preDieta}(u, d) \wedge \text{preDieta}(u, d') \\ \wedge d = d' \wedge \text{inizio}(i, d) \wedge \text{inizio}(i', d') \wedge \text{fine}(f, d) \\ \wedge \text{fine}(f', d') \rightarrow \exists t \quad & \text{dataora}(t) \wedge \\ & (i \leq t \wedge t \leq f) \wedge (i' \leq t \wedge t \leq f') \end{aligned}$$

[V. Abbonamento. disgiunto]

$$\begin{aligned} \forall p, a, a', d, d', t \quad & \text{Premium}(p) \wedge \text{preAbb}(p, a) \wedge \text{preAbb}(p, a') \\ \wedge a \neq a' \wedge \text{tipAbb}(t, a) \wedge \text{tipAbb}(t', a') \wedge \text{durata}(d, t) \\ \wedge \text{durata}(d', t') \rightarrow \exists t' \quad & \text{dataora}(t') \wedge \\ & (d.a \leq t' \wedge t \leq d.a) \wedge \\ & (d'.a \leq t' \wedge t' \leq d'.a) \end{aligned}$$

`isValid(a: Alimento): boolean`

pre: /

post: $r = \{u \mid \text{valida}(u, a)\}$

result = $\begin{cases} \text{if } |v| \geq 10 \wedge \neg \text{isInvalid}(a) : \text{return true} \\ \text{else: return false} \end{cases}$

`isValid(a: Alimento): boolean`

pre: /

post: $r = \{u \mid \text{invalida}(u, a)\}$

result = $\begin{cases} \text{if } |v| \geq 100 : \\ \quad \text{return true} \\ \text{else: return false} \end{cases}$

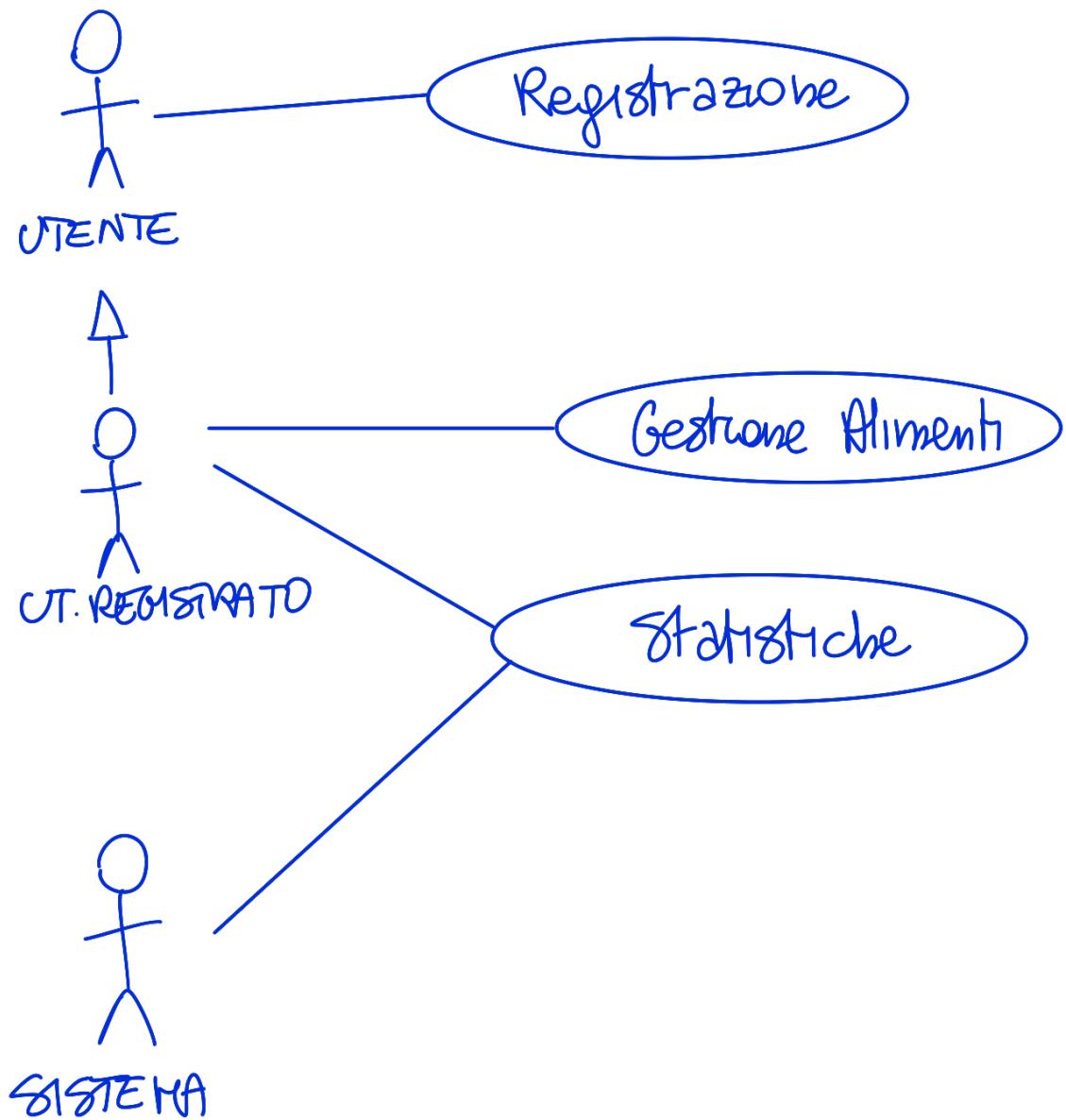
`isUsable(a: Alimento, u: Utente): boolean`

pre: /

post:

$r = \begin{cases} \text{if } [\text{isValid}(a) \vee \text{add}(a, u)] : \\ \quad \text{return true} \\ \text{else: return false} \end{cases}$

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:

iscrizione(r: str, cf: CodFis, e: Email, p: Passar, n: Nazione): Utente

2 Specifica use-case: *Gestione Alimenti* (nome use-case)

Operazioni dello use-case:

*inserisciAlim(u: Utente, n: str, m: Marca(0,1), g: reale > 0(0,1), k: reale > 0
m: UNIT, p: reale > 0, n: Nutrienti(1,N), c: str(0,1)): Alimento*

isValid(a: Alimento): boolean

isUsable(a: Alimento, u: Utente): boolean

isValid(a: Alimento): boolean

3 Specifica use-case: *Statistiche* (nome use-case)

Operazioni dello use-case:

calcolaCalorierite(d: data): reale > 0

calcoloTotogrammi(d: data): reale > 0

alimPiùAdd(t: TipoPasto): (a: Alimento, n: int > 0, m: reale > 0)(1,N)

calcolaRimanenze(g: data, d: Data, p: Premium): (n: Nutriente, m: reale > 0)(0,N)

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

Operazioni dello use-case:

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

Operazioni dello use-case:

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

Operazioni dello use-case:

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

calcolaTotGrammi (d: data): reale > 0

pre: /

post: No side-effect

Val. ritorno:

$$\begin{aligned} P = \{ (t, q) \mid & \text{TipoPasto}(t) \wedge \exists p, a \text{ Pasto}(p) \wedge \text{ristanteP}(d, p) \\ & \wedge \text{Nutrienti}(n) \wedge \text{Alimento}(a) \wedge \text{alimPasto}(a, p) \\ & \wedge \text{alNut}(a, n) \wedge \text{quantita'}(a, n, q) \} \end{aligned}$$

$$\text{result} = \sum_{(p, q) \in P} q$$

calcolaKCalIngerite (d: data): reale > 0

pre: /

post: No side-effect

Val. ritorno:

$$\begin{aligned} K = \{ (t, K) \mid & \text{TipoPasto}(t) \wedge \exists p, a \text{ Pasto}(p) \wedge \text{ristanteP}(d, p) \\ & \wedge \text{Alimento}(a) \wedge \text{KCal-unita'}(K, a) \\ & \wedge \text{alimPasto}(a, p) \} \end{aligned}$$

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

Risposta alla Domanda 5 (segue)

$\text{alimPiùAdd}(t : \text{TipoPasto}) : (\alpha : \text{Alimento}, n : \text{int} > 0, m : \text{reale} > 0)(1, N)$

pre: /

post: No side-effect

Val ritorno:

$$A = \left\{ (\alpha, n, m) \mid \text{Alimento}(\alpha) \wedge \right. \\ \left. \times \left\{ (p, q) \mid \text{Pasti}(p) \wedge \text{tipPas}(t, p) \wedge \right. \right. \\ \left. \left. \text{alimPasto}(\alpha, p) \wedge \text{quantità}(p, \alpha, q) \right\} \right\}$$

$$\wedge n = |X| \wedge m = \left[\frac{\sum_{(p, q) \in X} q}{|n|} \right]$$

$$\text{result} = \left\{ (\alpha, \alpha_n, m) \in A \mid \begin{array}{l} \alpha_n = \arg\max(n) \\ (\alpha, n, m) \in A \wedge n = \alpha_n \end{array} \right\}$$

Risposta alla Domanda 5 (segue)

`calcolaRimanenze(g: data, d: dieta, p: Premium): (n: Nutriente, m: reale > 0)(O, N)`

pre: $\exists i, f \text{ Premium}(p) \wedge \text{preDieta}(p, d) \wedge \text{inizio}(i, d) \wedge i \leq d$
 $\wedge \text{dietaFine}(d) \wedge \text{fine}(f, d) \wedge f \geq d$

post: No side-effect.

Val. ritorno:

$$X = \{(nu, q) \mid \text{Nutriente}(nu) \wedge \text{dielNut}(d, n) \wedge \\ \exists i \text{ intervallo}(i, d, n) \wedge \text{max} = i \cdot a \wedge \\ P = \{ (p, a, q_n) \mid \text{Pasto}(p) \wedge \text{alimPasto}(a, p) \wedge \\ \text{alNut}(a, nu) \wedge \text{quantita}(a, n, q_n) \wedge \\ \text{data}(p, g) \} \wedge \\ Q = \text{max} - \sum_{(p, a, q_n) \in P} q_n \}$$

$$B = \{ (a, q) \mid \exists p \text{ Pasto}(p) \wedge \text{data}(g, p) \wedge \text{alimPasto}(a, p) \\ \wedge \text{KCal}(a, q) \}$$

$$K = \{ K \mid \text{KCal}(c, d) \wedge K = c - \sum_{(a, q) \in B} q \}$$

$$\text{result} = X \cup K$$

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(500);
create domain IntegerGZ as integer check (value > 0)
create domain IntegerGEZ as integer check (value ≥ 0)
create domain RealGZ as real check (value > 0)
create domain RealGEZ as real check (value > 0)

create type Periodo(
    d2: timestamp
    a: timestamp)

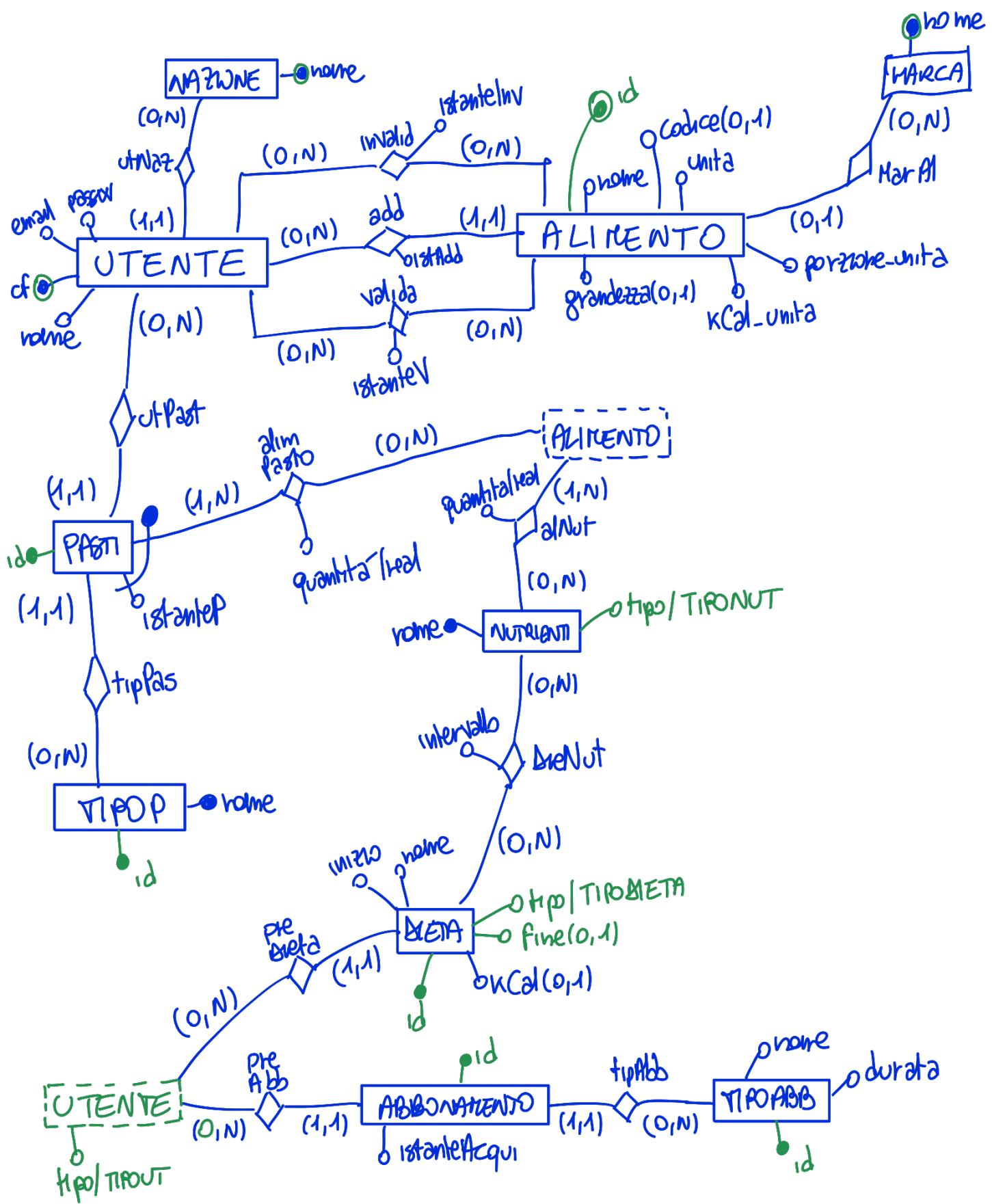
create type Intervallo(
    da: RealGZ
    a: RealGZ)

create domain CF as char(16) check (isValidCodiceFiscale(value))
create domain Email as StringS check (isValidEmail(value))
create domain Passwr as StringS check (isValidPasswr(value))
create type Tipop as enum('colazione', 'pranzo', 'snack', 'cena')
create type Unit as enum('litri', 'unità', 'grammi')

```

[continua a PAG 23]

Diagramma ER ristrutturato



Breve descrizione delle scelte effettuate durante la ristrutturazione

fusione 15-a Nutriente e Opzionale

fusione 15-a Dieta e FineDieta

creazione relen per Utente e Premium

create type TIRODIETA as enum ('Dieta', 'Dieta Fine')

create type TIRONUIT as enum('Nutriente', 'Nutriente Opzionale')

create type TIPOUT as enum('Utente', 'Premium')

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. Dieta.Fine]

$\forall d \text{ Dieta}(d) \rightarrow$

$[\text{tipo}(d, 'DietaFinita')] \leftrightarrow [\exists f \text{ fine}(f, d)]$

[V. Alimento.noOpzio]

$\forall a, n \text{ Alimento}(a) \wedge \text{Nutriente}(n) \wedge \text{NutOpzio}(n, \text{false}) \rightarrow \text{alNut}(a, n)$

Risposta alla Domanda 6 (segue)

[N.Utente.tip]

$\forall u \text{ Utente}(u) \rightarrow$

$[tip(u, 'Premium')] \leftrightarrow [\exists d \text{ prenota}(d, u)] \wedge$

$[tip(u, 'Premium')] \leftrightarrow [\exists a \text{ prefAbbo}(a, u)] \wedge$

$\forall u, u', c, c' [utente(u) \wedge utente(u') \wedge u \neq u'$

$cf(u, c) \wedge cf(u', c) \rightarrow u = u'$

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>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):

La relazione accorda le relazioni che implementano le seguenti relationship:

2	Relazione . <u>UTENTE</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>cf</u> email <u>passw</u> name <u>nazione</u> <u>tip</u>	
Domini	<u>CodFis</u> Email <u>Passw</u> <u>StringS</u> <u>StringS</u> <u>TipUT</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 : nazione refer Nazione(name)

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

3	Relazione . <u>PASTI</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> <u>istanteP</u> <u>alimento</u> <u>utente</u> <u>tipop</u>	
Domini	<u>integer</u> <u>timestamp</u> <u>StringS</u> <u>CodFis</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 : utente refer utente(cf) FK : alimento refer Alimento(codice)

chiavi(istanteP,alimento) PK : tipop refer Tipop(id) inclusione: id ⊆ alimPasto(Pasto)

La relazione accorda le relazioni che implementano le seguenti relationship: utPast,tipPast

4	Relazione . <u>TIPOP</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> <u>name</u>	
Domini	<u>integer</u> <u>Tipop</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:

5	Relazione . <u>MARCA</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>name</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):

La relazione accorda le relazioni che implementano le seguenti relationship:

6 Relazione	<u>AZIMENTO</u>	(nome)	Derivante da:	entità	relationship (cerchiare)
--------------------	-----------------	--------	---------------	---------------	---------------------------------

Attributi	<u>name</u>	<u>codice*</u>	<u>unità</u>	<u>porzione</u>	<u>Kcal</u>	<u>id</u>	<u>grandezza*</u>	<u>marca*</u>
Domini	<u>StringS</u>	<u>StringS</u>	<u>UNIT</u>	<u>RealGZ</u>	<u>RealGZ</u>	<u>integer</u>	<u>RealGZ</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): seriale: id

PK: marca refer Marca(name)

La relazione accorda le relazioni che implementano le seguenti relationship: marA!

7 Relazione	<u>AZIMENTO</u>	(nome)	Derivante da:	entità	relationship (cerchiare)
--------------------	-----------------	--------	---------------	---------------	---------------------------------

Attributi	<u>utente</u>	<u>isfAdd</u>						
Domini	<u>CodFisc</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):

PK: utente refer Utente(cf)

inclusione: id ⊆ alNut(alimento)

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

8 Relazione	<u>alNut</u>	(nome)	Derivante da:	entità	relationship (cerchiare)
--------------------	--------------	--------	---------------	---------------	---------------------------------

Attributi	<u>quantità</u>	<u>alimento</u>	<u>nutrienti</u>					
Domini	<u>RealGZ</u>	<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: alimento refer Alimento(id)

PK: nutrienti refer Nutrienti(name)

La relazione accorda le relazioni che implementano le seguenti relationship:

9 Relazione	<u>AlimPasto</u>	(nome)	Derivante da:	entità	relationship (cerchiare)
--------------------	------------------	--------	---------------	---------------	---------------------------------

Attributi	<u>quantità</u>	<u>alimento</u>	<u>pasti</u>					
Domini	<u>RealGZ</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):

FK: alimento refer Alimento(id)

PK: pasti refer Pasti(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

10 Relazione	<u>invalida</u>	(nome)	Derivante da:	entità	relationship (cerchiare)
---------------------	-----------------	--------	---------------	---------------	---------------------------------

Attributi	<u>isInv</u>	<u>utente</u>	<u>alimento</u>					
Domini	<u>timestamp</u>	<u>CodFisc</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(cf)

FK: alimento refer Alimento(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

11 Relazione <u>Valida</u>(nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>idVal</u> <u>utente</u> <u>alimento</u>	
Domini <u>timestamp</u> <u>Codfisc</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(cf)

FK: alimento refer Alimento(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

12 Relazione <u>NUTRIENTI</u>(nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>name</u> <u>tipo</u>	
Domini <u>StringS</u> <u>TIPONUT</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: NUTOPZZO ≠ NULL \leftrightarrow NUTRIENTI = NULL

La relazione accorda le relazioni che implementano le seguenti relationship:

13 Relazione <u>bieNut</u>(nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>intervallo</u> <u>nutrienti</u> <u>dieta</u>	
Domini <u>Intervallo</u> <u>StringS</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: nutrienti refer Nutrienti(name)

PK: dieta refer Dieta(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

14 Relazione <u>BIETA</u>(nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>inizio</u> <u> nome</u> <u> tipo</u> <u> fine*</u> <u> KCal*</u> <u> id</u> <u> utente</u>	
Domini <u>timestamp</u> <u>StringS</u> <u>TipBiet</u> <u>timestamp</u> <u>Intervallo</u> <u>integer</u> <u>Codfisc</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): **setiale:id**

PK: utente refer Utente(cf)

ennupla = [tipo = 'BIETA FINITA' \leftrightarrow fine ≠ NULL] \rightarrow (inizio < fine) \rightarrow preBiet

La relazione accorda le relazioni che implementano le seguenti relationship:

15 Relazione <u>ABBONAMENTO</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>id</u> <u>istAcqui</u> <u>utente</u> <u>TIPOABB</u>	
Domini <u>integer</u> <u>timestamp</u> <u>Codfisc</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): **setiale:id**

PK: utente refer Utente(cf)

PK: tipAbb refer TipAbb(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

preAbb, tipAbb

16	Relazione <u>TROBB</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi	<u>id</u> nome durata	
Domini	integer strings periodo	

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

La relazione accorda 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 ennupla, di dominio):

La relazione accorda 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 ennupla, di dominio):

La relazione accorda 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 ennupla, di dominio):

La relazione accorda 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 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 ennupla); (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.

[V. Utente.tipo]

Operaz: inserimento e modifica su Abbonamento

Istante: prima dell'operaz. intercettata

Funz:

```
isError = exists (select *
                  from Utente u
                  where u.cf = new.utente and u.tipo = 'Premium')
```

if isError: genera errore

else: permetti l'operazione

[V. Alimento.noOpzio]

Operaz: inserimento e modifica su Abbonamento

Istante: prima dell'operaz. intercettata

Funz:

```
isError = exists (select *
                  from Nutriente n
                  where n.tip = 'Nutriente'
                  and not exists (select *
                                  from alNut an
                                  where an.alimento = new.id
                                  and an.nutriente = n))
```

if isError: genera errore

else: permetti l'operazione

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

$\text{alimPiùAlta}(\text{tp: integer}) : (<\text{a: integer}, \text{n: IntegerGz}, \text{m: RealGz}>)$

pre: nessuna

post: $A = \text{select count(*)}$
 from alimPast ap
 $\text{where ap.tipo = :tp}$
 $\text{group by ap.alimento}$

$B = \text{select ap.alimento, count(*)}$
 from alimPast ap
 $\text{where ap.tipo = :tp}$
 $\text{group by ap.alimento}$
 $\text{having count(p) = (select max(q.count)}$
 from :A as q)

$C = \text{select ap.alimento, sum(ap.quantità)}$
 from alimPast ap
 $\text{where ap.alimento = :B.alimento}$
 $\text{group by ap.alimento}$

return: $\text{select :C.alimento, :C.sum / :B.count}$
 $\text{from C,B where :B.alimento = :C.alimento}$

Risposta alla Domanda 8 (segue)

`calcolaRimanenze(g:date, d:integer, u:integer):(<n:Strings, m:RealGZ>)`

`Pre:` `isError = exists(select *`

`from Dieta d, Utente u`

`where u.tipo = 'Premium' and d.id = :d`

`and (d.inizio <= :g and d.fine = NULL)`

`or (d.tipo = 'DeltaFine' and d.inizio <= :g
and d.fine >= :g)`

`and u.cf = :u)`

`if isError: block`

`else: commit`

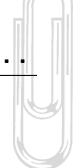
`Post:`

`A = select dn.nutrienti, dn.intervallo.a as max
from DietaNut dn
where dn.dieta = :d`

`B = select an.nutriente, sum(an.quantita')
from alNut an, alimPasto ap, Pasto p
where an.alimento = ap.alimento and ap.pasto = p.id
and p.utente = :u and p.istanteP = :g
group by an.nutriente`

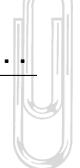
`return = select :A.nutriente, :A.max - :B.sum
From A, B
where :A.nutriente = :B.nutriente`

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.]