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 – Modulo risposte prova scritta (diagramma delle classi UML)**

Dati dello studente e dell'esame

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

Data: .....

Corso di laurea e canale di appartenenza:

☐ Laurea in Informatica, canale 1 (Prof. G. Perelli)

☐ Laurea in Informatica, canale 2 (Prof.ssa M. De Marsico)

Firma di un membro della Commissione per avvenuta identificazione:

.....

Rinuncia alla prova

☐ Desidero rinunciare a questa prova d'esame. Firma: .....

[Versione 2024-06-05.UML]

Questo modulo è ottimizzato per la stampa fronte-retro

Istruzioni e regole d'esame

Prima dell'esame

- Stampare questo modulo, preferibilmente fronte-retro, e rilegarlo con un fermaglio rimovibile, come quello disegnato in alto
- Compilare il frontespizio con i propri dati, come richiesto
- Scrivere la propria matricola nello spazio apposito nella parte alta di tutte le pagine

Durante l'esame

- La prova è dimensionata per essere svolta in circa 3 ore. Tuttavia, data la sua natura fortemente progettuale, la Commissione offre agli studenti la più ampia disponibilità di tempo, al fine ovviare ad eventuali (e limitati) errori di analisi/progettazione rilevati più a valle del ciclo di vita.

  Il tempo massimo per la consegna è quindi rilassato a 5 ore (il massimo tempo compatibile con le disponibilità di aule).
- Scrivere le risposte negli spazi predisposti sotto le relative domande. Le ultime pagine sono vuote e possono essere usate come minute oppure, se puntate opportunamente, per contenere risposte in caso gli spazi appositi dovessero risultare insufficienti.
- Non è possibile usare alcun tipo di materiale didattico.
- In caso di necessità di ulteriori fogli (in proprio possesso), chiedere preventivamente alla Commissione una nuova procedura di controllo.
- La Commissione può rispondere solo a brevi domande inerenti al testo dei quesiti.
- Tra la seconda e la quarta ora d'esame, gli studenti possono effettuare **brevi pause** (uno studente alla volta) seguendo la seguente procedura:
  1. Alla lavagna è riportata una coda denominata 'Coda prenotazioni pause'. Sia n (un intero) l'elemento in fondo alla coda (si assuma $n = 0$ in caso di coda vuota).
  2. Recarsi alla lavagna ed aggiungere l'intero $n + 1$ come proprio contrassegno in fondo alla coda, seguito da una stringa a propria scelta (ad es., le proprie iniziali).
  3. Se il proprio contrassegno non è l'elemento affiorante della coda, tornare al lavoro in attesa che lo diventi.
  4. Consegnare tutti i fogli di lavoro e il testo d'esame alla Commissione ed uscire.
  5. Al rientro, cancellare il proprio contrassegno dalla coda di modo da permettere al successivo studente prenotato di uscire, e riprendere i fogli prima consegnati.

Al momento della consegna

- Ordinare tutti i fogli che si vuole far valutare e rilegarli con un fermaglio rimovibile. Non includere fogli che la Commissione non deve valutare (ad es., requisiti, minute), ma includere ovviamente il frontespizio.
- Consegnare i fogli ordinati **nelle mani** di un membro della Commissione. **Non lasciare l'aula senza la conferma, da parte della Commissione, del buon esito delle operazioni di consegna**.

In caso di rinuncia

- È possibile rinunciare alla consegna a partire dalla seconda ora d'esame. In caso di rinuncia, consegnare nelle mani della Commissione solo il frontespizio, dopo aver compilato e firmato la sezione dedicata.

[Versione 2024-06-05.UML]

Sommario delle domande

Si richiede di progettare l'applicazione descritta dalla specifica dei requisiti effettuando le fasi di Analisi concettuale dei requisiti e di Progettazione logica della base dati e delle funzionalità, utilizzando la metodologia vista nel corso.

In particolare (vengono indicati i tempi suggeriti per i diversi passi chiave):

Parte 1: Analisi concettuale dei requisiti Effettuare la fase di Analisi concettuale dei requisiti producendo lo schema concettuale per l'applicazione, che includa:

- Analisi dei dati (45 minuti; 75 minuti al massimo):
  - un diagramma UML concettuale delle classi (*)
  - (parte del)le specifiche formali delle classi e delle associazioni
  - le specifiche dei tipi di dato
  - la specifica formale dei vincoli esterni (*)
- Analisi delle funzionalità:
  - un diagramma UML degli use-case (5 minuti; 10 minuti al massimo)
  - la segnatura di tutte le operazioni di use-case (10 minuti)
  - (parti del)le specifiche formali degli use-case. (30 minuti; 60 minuti al massimo)

Si richiede *esplicitamente* di modellare le specifiche formali delle operazioni di clase e/o use-case necessarie a modellare i requisiti contrassegnati dalla barra laterale (come quella qui a sinistra), *incluse* tutte le eventuali operazioni ausiliarie, usando l'estensione della logica del primo ordine studiata nel corso. (*)

Parte 2: Progettazione della base dati e delle funzionalità Effettuare la progettazione della base dati e delle funzionalità a partire dallo schema concettuale prodotto nella Parte 1, ed in particolare eseguire i seguenti passi:

- Progettazione della base dati relazionale con vincoli:
  - Ristrutturazione del diagramma UML concttuale delle classi e delle specifiche (20 minuti; 30 minuti al massimo):
    - scelta del DBMS da utilizzare
    - progettazione della corrispondenza tra i tipi di dato concettuali ed opportuni domini SQL (domini base o utente, oppure realizzati mediante relazioni aggiuntive) supportati dal DBMS scelto
    - ristrutturazione del diagramma UML concttuale delle classi e delle specifiche dei vincoli esterni.
  - Produzione dello schema relazionale della base dati e dei relativi vincoli (*) (30 minuti; 60 minuti al massimo)
- Progettazione delle funzionalità (30 minuti; 45 minuti al massimo):
  - definizione della specifica realizzativa delle operazioni necessarie a modellare i requisiti contrassegnati dalla barra laterale, in modo conforme alla loro specifica concettuale prodotta nella fase di Analisi, in termini di algoritmi in pseudo-codice e comandi SQL immersi. (*)

Le pagine seguenti contengono le domande specifiche a cui è richiesto rispondere, ulteriori delucidazioni per ogni singolo punto, e spazi per le risposte.

Le pagine da 31 in poi possono essere utilizzate per scrivere minute che non verranno valutate.

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

Questa pagina è stata intenzionalmente lasciata vuota

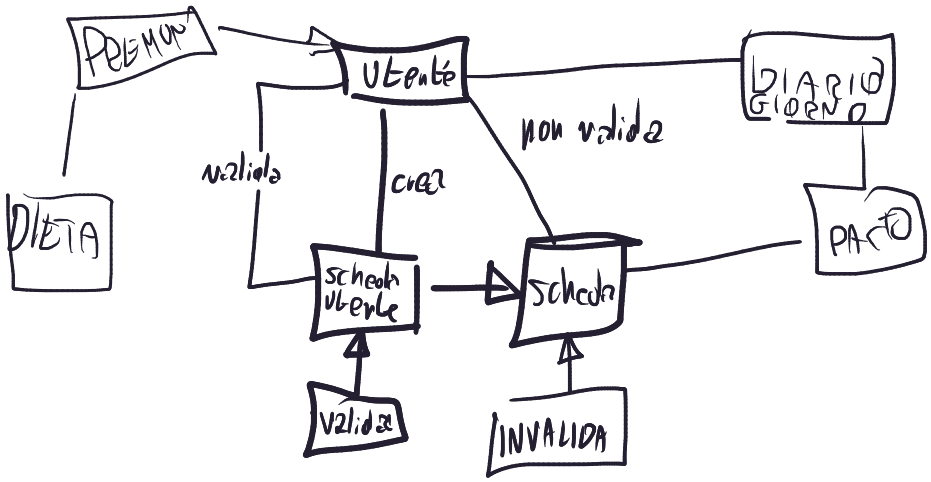
1 Analisi concettuale

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 username
1.4 email
1.5 password

2. alimento
2.1 nome
2.2 marca [0..1]
2.3 ingrediente conf [0..1]
2.4 cod identific.
2.5 cod barre [0..1]
2.6 [kcal/unità]
2.7 unità
2.8 porzione standard (quantità)
2.9 quantità nutriente per unità



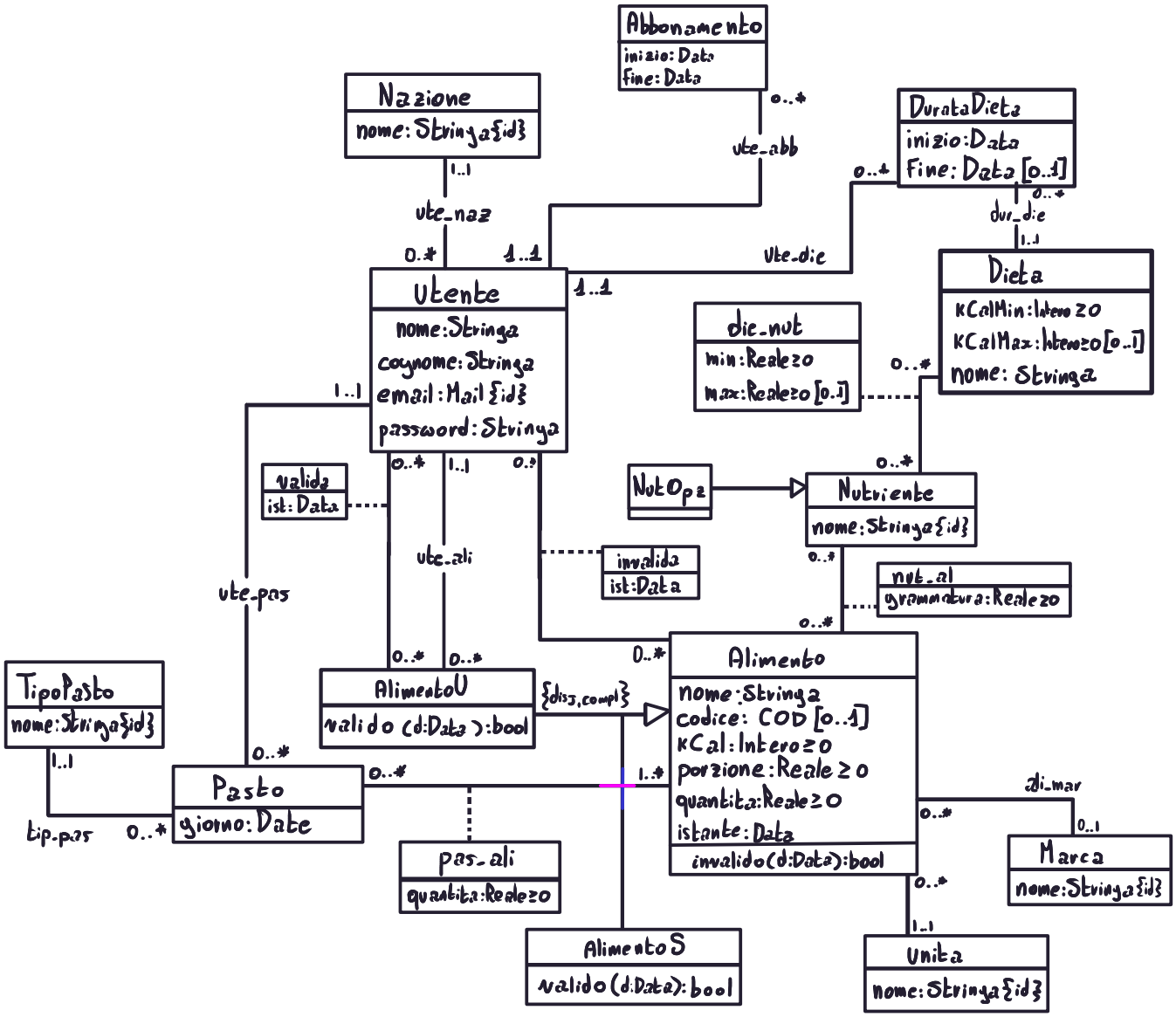
[continua alla pagina seguente]

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

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

Diagramma UML concettuale delle classi

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



[Versione 2024-06-05.UML]

[continua alla pagina seguente]

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

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

Risposta

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

Nome: AlimentoS .....

Operazioni, vincoli:

valido(d:Data):bool
- pre: $d \leq$ adesso
- post: $V = \{(u, dv) \mid invalida(u, this) \wedge ist(u, this, dv) \wedge dv \leq d\}$

$|V| \geq 100 \rightarrow$ result=False $\wedge$
$|V| < 100 \rightarrow$ result=True

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

Nome: AlimentoU .....

Operazioni, vincoli:

[V.invalidato_se_valido]
$\forall a, d_{inv}$
$[\text{AlimentoU}(a) \wedge \exists u \ \ invalida(u, a) \wedge ist(u, a, d_{inv})]$
$\rightarrow [|\{(u, dv) \mid valida(u, a) \wedge ist(u, a, dv) \wedge dv \leq d_{inv}\}| \geq 10]$

valido(d:Data):bool
- pre: $d \leq$ adesso
- post: $V = \{(u, dv) \mid invalida(u, this) \wedge ist(u, this, dv) \wedge dv \leq d\}$
  $V_v = \{(u, dv) \mid valida(u, this) \wedge ist(u, this, dv) \wedge dv \leq d\}$
  $|V_v| \geq 10 \wedge |V| \leq 100 \rightarrow$ result=True
  $\wedge [|V_v| \leq 10 \vee (|V_v| \geq 10 \wedge |V| \geq 100)] \rightarrow$ result=False

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

Nome: Alimento

Operazioni, vincoli:

[V.validato_dopo_creato]

$\forall a, da, u, dv\ [Alimento U(a) \wedge valida(u,a) \wedge ist(u,a,dv) \wedge istante(a,da)] \rightarrow da < dv$

[V.invalidato_dopo_creato]

$\forall a, da, u, dv\ [Alimento(a) \wedge invalida(u,a) \wedge ist(u,a,dv) \wedge istante(a,da)] \rightarrow da < dv$

[V.nutrienti_obbligatori]

$\forall a, n\ [Alimento(a) \wedge Nutriente(n) \wedge \neg NutOpz(n)] \rightarrow not\_ali(a,n)$

[V.cod_unico]

$\forall a1,a2\ c1,c2\ [Alimento(a1) \wedge codice(a1,c1) \wedge Alimento(a2) \wedge codice(a2,c2) \wedge a1 \neq a2] \rightarrow c1 \neq c2$

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

Nome: Pasto

Operazioni, vincoli:

[V.nel_pasto_se_valido_e_creato]

$\forall p, dp, a, da, u\ [Pasto(p) \wedge giorno(p,dp) \wedge pas\_ali(p,a) \wedge Utente(u) \wedge ute\_pas(u,p) \wedge Alimento(a) \wedge istante(a,da)] \rightarrow da \leq dp \wedge [valido(a,dp,True) \vee [\neg invalido(a,dp,False) \wedge \exists k\ valida(u,a) \wedge ist(u,a,k) \wedge k \leq dp]\ \forall ute\_ali(u,a)]$

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

Nome: Dieta

Operazioni, vincoli:

[V.intervallo_calorie]

$\forall d, m, M\ [Dieta(d) \wedge KCalMin(d,m) \wedge KCalMax(d,M)] \rightarrow m \leq M$

[V.durata_dieta]

$\forall d, i, f\ [DurataDieta(d) \wedge inizio(d,i) \wedge Fine(d,f)] \rightarrow i \leq F$

[V.intervallo_nutrienti]

$\forall n, d, m, M\ [Dieta(d) \wedge min(d,n,m) \wedge max(d,n,M)] \rightarrow m \leq M$

[V.no_intersezione_dieta]

$\forall u, d1, d2, i1, i2\ [DurataDieta(d1) \wedge DurataDieta(d2) \wedge inizio(d1,i1) \wedge inizio(d2,i2) \wedge ute\_die(u,d1) \wedge ute\_die(u,d2) \wedge d1 \neq d2] \rightarrow [\neg \exists t\ t \geq i1 \wedge t \geq i2 \wedge [\exists F\ Fine(d1,f) \rightarrow t \leq f] \wedge [\exists F\ Fine(d2,f) \rightarrow t \leq f]]$

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

Nome: Abbonamento

Operazioni, vincoli:

[V.durata_abbonamento]

$\forall a, i, f\ [Abbonamento(a) \wedge inizio(a,i) \wedge Fine(a,f)] \rightarrow i \leq f$

[V.no_intersezione_abbonamento]

$\forall a1,a2,u,i1,i2,f1,f2\ [Abbonamento(a1) \wedge Abbonamento(a2) \wedge inizio(a1,i1) \wedge inizio(a2,i2) \wedge Fine(a1,f1) \wedge Fine(a2,f2) \wedge a1 \neq a2 \wedge ute\_abb(a1,u) \wedge ute\_abb(a2,u)] \rightarrow [[f2 < i2] \vee [f2 < i1]]$

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

Nome: Alimento

Operazioni, vincoli:

invalido(d:Data):bool

- pre: $d \leq adesso$
- post: $V = \{(u, dv) \mid invalida(u, this) \wedge ist(u, this, dv) \wedge dv \leq d\}$

$|V| \geq 100 \rightarrow result = True \wedge$

$|V| < 100 \rightarrow result = False$

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

Nome: AlimentoU

Operazioni, vincoli:

[V.no_auto_validazione]

$\forall a, u\ [Alimento(a) \wedge valida(u,a)] \rightarrow \neg ute\_ali(u,a)$

[continua alla pagina seguente]

[Versione 2024-06-05.UML]

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

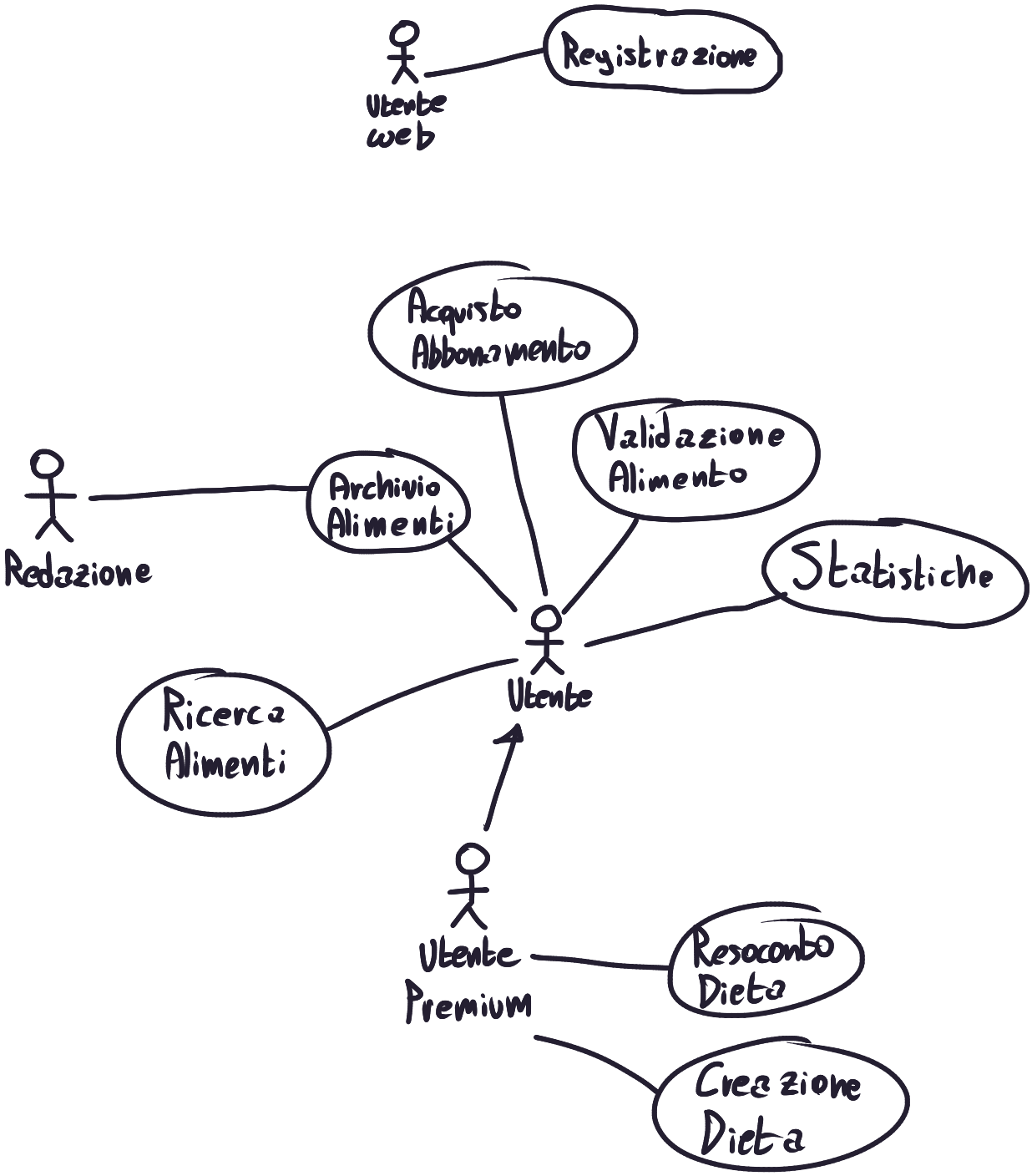
COD = [0-9]{10}

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

[continua alla pagina seguente]

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



Questa pagina è stata intenzionalmente lasciata vuota

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

Risposta

Registrazione

registra(n:Stringa, c:Stringa, naz:Nazionalita, e:Mail, p:Stringa):Utente

---

Validazione Alimento

valida(a:AlimentoU)

invalida(a:Alimento)

Acquisto Abbonamento

compra(i:Data, f:Data):Abbonamento

Archivio Alimenti

aggiungi_nuovo(n:Stringa, c:COD[0..1], cal:Reale≥0, q:Reale≥0, u:Unita, m:Marca[0..1], nut:(Reale≥0, Nutriente)[0..*]):Alimento

---

Ricerca Alimenti

cerca(n:Stringa, cal:Reale≥0[0..1]):Alimento[0..*]

Statistiche

alimenti_comuni(p:TipoPasto):(Alimento, Intero, Reale)[0..*]

cal_giornaliere(u:Utente, d:Data):Reale≥0

nut_giornalieri(u:Utente, d:Data):(Reale≥0, Nutriente):[0..*] //numero minimo: nut obbligatori

---

Creazione Dieta

crea_dieta(calmin:Reale≥0, calmax:Reale≥0[0..1], n:(m:Reale≥0, M:Reale≥0[0..1], Nutriente)):Dieta

inizia(d:Dieta):DuratoDieta

termina(d:Dieta):DuratoDieta

Resoconto Dieta

nutrienti_rimanenti(u:Utente, d:Data)

Questa pagina è stata intenzionalmente lasciata vuota

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

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

Risposta

alimenti_comuni(p:TipoPasto): (Alimento, Intero, Reale) [0..*]

- pre: nessuna
- post-cond:

$$A = \{(a, n, q) \mid Alimento(a) \wedge alimento\_in\_pasto(a, p, n, q)\}$$

$$Result = \underset{(a,n,q) \in A}{Argmax}(n)$$

---

alimento_in_pasto(tp:TipoPasto, a:Alimento): (Intero ≥0, Reale ≥0) // dato 1 tipo di pasto ed un alimento, restituisce il numero di pasti di quel tipo che includono tale alimento, con la quantità media.

- pre-cond: nessuna
- post-con: $P = \{(p, q) \mid tip\_pas(p, tp) \wedge pas\_ali(p, a) \wedge quantita(p, a, q)\}$

$$Result = \left(|P|, \sum_{(p,q) \in P} q \cdot \frac{1}{|P|} \right)$$

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

Risposta alla Domanda 5 (segue)

nutrienti_rimanenti(u:Utente, d:Data): (Nutriente, Reale≥0)[0..*]

- pre-cond: $[\exists a,i,f[\text{Abbonamento}(a)\wedge\text{inizio}(a,i)\wedge\text{fine}(a,f)\wedge\text{ute\_abb}(a,u)\wedge i\le d\le f)]$

  $\wedge[\exists D,i,f\ [\text{DurataDieta}(D)\wedge\text{inizio}(a,i)\wedge\text{fine}(a,f)\wedge i\le d\le f\wedge\text{ute\_die}(u,D)]$

- post-cond: $N=\{(n,r)\mid \text{Nutriente}\wedge\text{nutriente\_rimanente}(u,d,n,r)\}$

  Result = N

---

nutriente_rimanente(u:Utente, d:Data, n:Nutriente): Reale≥0

- pre-cond: $[\exists a,i,f[\text{Abbonamento}(a)\wedge\text{inizio}(a,i)\wedge\text{fine}(a,f)\wedge\text{ute\_abb}(a,u)\wedge i\le d\le f)]$

  $\wedge[\exists D,i,f\ [\text{DurataDieta}(D)\wedge\text{inizio}(a,i)\wedge\text{fine}(a,f)\wedge i\le d\le f\wedge\text{ute\_die}(u,D)]$

- post-cond:

$$Q=\left\{(p,a,g,q)\ \middle|\ \begin{array}{l}\text{Alimento}(a)\wedge\text{nut\_al}(a,n)\wedge\text{grammatura}(a,n,g)\wedge\quad\text{Pasto}(p)\wedge\text{giorno}(p,d)\\ \wedge\text{pas\_ali}(a,p)\wedge\text{quantita}(a,p,q)\wedge\text{ute\_pas}(p,u)\wedge\exists dD,i\quad\text{DurataDieta}(dD)\wedge\\ \text{ute\_die}(u,dD)\wedge\text{inizio}(dD,i)\wedge d\ge i\wedge[\exists f\ \text{fine}(dD,f)\to d\le f]\wedge\exists D\ \text{Dieta}(D)\wedge\\ \text{dur\_die}(dD,D)\wedge\text{die\_nut}(D,n)\wedge\exists M\ \max(D,n,M)\end{array}\right\}$$

$$Q_{ingerito}=\sum_{(p,a,g,q)\in Q} g\cdot q$$

Sia M tale che : $\exists dD,i$ DurataDieta(dD) $\wedge$
ute_die(u,dD) $\wedge$ inizio(dD,i) $\wedge$ d≥i $\wedge$ [$\exists f$ fine(dD,f) → d≤f] $\wedge$
$\exists D$ Dieta(D) $\wedge$ dur_die(dD,D) $\wedge$ die_nut(D,n) $\wedge$ max(D,n,M)

$M\ge Q_{ingerito}\to \text{Result}=M-Q_{ingerito}\wedge$

$M<Q_{ingerito}\to \text{Result}=0$

2 Progettazione della base dati e delle funzionalità

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

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

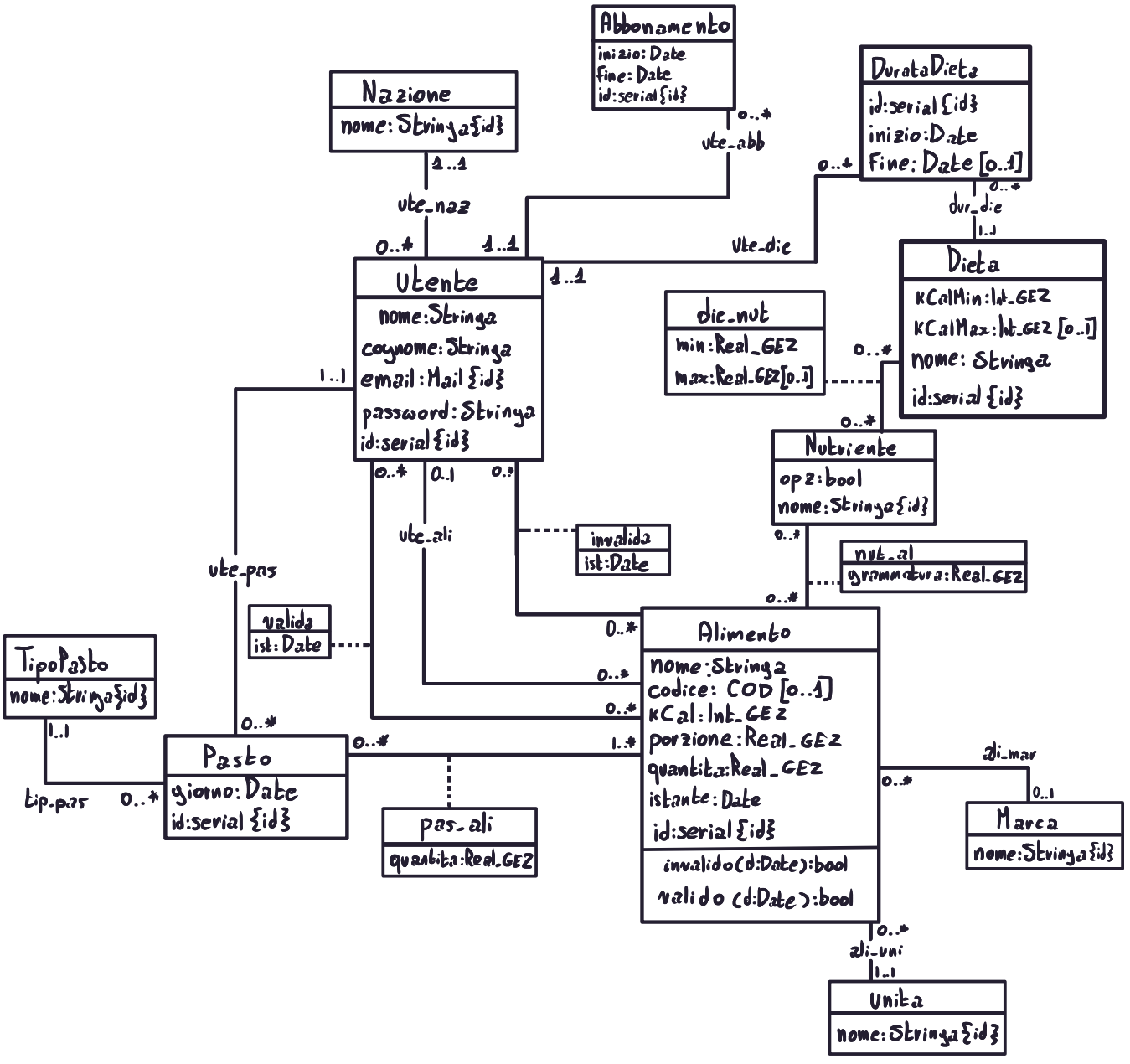
Descrivere brevemente le principali scelte effettuate.

DBMS da utilizzare PostgreSQL.....

Corrispondenza tra tipi di dato concettuali e domini supportati dal DBMS

```
create domain Stringa as varchar NOT NULL;

create domain Mail as varchar ~'[a-zA-Z0-9]+@[a-zA-Z0-9]+.[a-z]{2,6}';

create domain COD as varchar~'[0-9]{10}';

create domain Int_GEZ as Integer check(value >=0);

create domain Real_GEZ as Real check(value >= 0);
```

Diagramma UML delle classi ristrutturato



[Versione 2024-06-05.UML]

Breve descrizione delle scelte effettuate durante la ristrutturazione

Fusione su alimento

Fusione su nutriente

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.nutrienti_obbligatori]

$\forall a,n\,[\text{Alimento}(a) \wedge \text{Nutriente}(n) \wedge \text{opz}(n,\text{False})] \rightarrow \text{nut\_ali}(a,n)$

[V.validato_se_utente]

$\forall a,u\,[\text{Alimento}(a) \wedge \text{valida}(u,a)] \rightarrow [\exists u'\ \ \text{ute\_ali}(u',a)]$

[V.invalidato_se_valido]

$\forall a, d_{max}$
$[\text{Alimento}(a) \wedge \exists u\ \ \text{invalida}(u,a) \wedge \text{ist}(u,a,d_{max})]$
$\rightarrow [\,|\{(u,dv) \mid \text{valida}(u,a) \wedge \text{ist}(u,a,dv) \wedge dv \leq d_{max}\}| \geq 10\}]$

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

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

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

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

Attributi	nome						
Domini	Stringa						

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

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

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

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

Attributi	nome	cognome	email	password	id	nazione		
Domini	Stringa	Stringa	Mail	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);

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

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

Attributi	inizio	Fine	id	utente			
Domini	Date	Date	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):

check(inizio <= Fine);

Fk utente ref Utente(id);

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

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

Attributi	kCalMin	kCalMax*	nome	id			
Domini	Real_GE2	Real_GE2	Stringa	serial			

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

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

check(kCalMax >= kCalMin OR kCalMax is NULL);

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

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

Attributi	id	inizio	Fine*	utente	dieta		
Domini	serial	Date	Date	Integer	Integer		

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

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

check(inizio <= Fine);

Fk dieta reF Dieta(id);

Fk utente rcF Utente(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: ute_die, dur_die

[Versione 2024-06-05.UML]

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

Attributi	nome						
Domini	Stringa						

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

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

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

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

Attributi	nome						
Domini	Stringa						

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

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

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

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

Attributi	nome	codice*	kCal	porzione	quantita	istante	id	unita
Domini	Stringa	COD	Int_GEZ	Real_GEZ	Real_GEZ	Date	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):

unique(codice);

FK unita ref Unita(nome);

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

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

Attributi	marca*	utente*						
Domini	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 ennupla, di dominio):

FK marca ref Marca(id); FK utente ref Utente(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: ali_mar, ute_ali .....

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

Attributi	utente	alimento	ist					
Domini	Integer	Integer	Date					

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 ref Utente(id);

FK alimento ref Alimento(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: ali, ute_ali .....

[Versione 2024-06-05.UML]

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

Attributi	utente	alimento	ist					
Domini	Integer	Integer	Data					

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 ref Utente(id);

Fk alimento ref Alimento(id);

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

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

Attributi	op2	nome						
Domini	bool	Stringa						

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

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

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

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

Attributi	nutriente	alimento	grammatura					
Domini	Stringa	Integer	Real_GE2					

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 nutriente ref Nutriente(nome);

Fk alimento ref Alimento(id);

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

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

Attributi	nutriente	dieta	min	max				
Domini	Stringa	Integer	Real_GE2	Real_GE2				

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 nutriente ref Nutriente(nome);

Fk dieta ref Dieta(id);

check(max IS NULL OR max >= min);

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

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

Attributi	nome							
Domini	Stringa							

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

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

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

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

Attributi	utente	tipo	id	giorno				
Domini	Integer	Stringa	serial	Date				

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 ref Utente(id)

Fk tipo ref TipoPasto(nome);

La relazione accorpa le relazioni che implementano le seguenti associazioni: tip_pas, utc_pas .....

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

Attributi	pasto	alimento	quantita					
Domini	Integer	Integer	Real_GEZ					

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

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

Fk pasto ref Pasto(id);

Fk alimento ref Alimento(id);

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

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

Attributi								
Domini								

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

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

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

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

Attributi								
Domini								

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

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

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

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

Attributi								
Domini								

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

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

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

[Versione 2024-06-05.UML]

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

Ulteriori vincoli esterni

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

T.nutrienti_obbligatori
op: Insert o Update Alimento

```
Error = EXISTS ((SELECT nome FROM Nutriente WHERE opz='False')
                 EXCEPT
                (SELECT n.nome
                 FROM Nutriente n, die_nut dn
                 WHERE dn.alimento=new.id AND dn.nutriente=n.nome
                 AND n.opz='False'));
```

if Error='True': errore e rollback
else: permetti op

T.no_intersezione_abbonamento
op: Insert o Update Abbonamento

```
Error= EXISTS (SELECT *
               FROM Abbonamento a
               WHERE a.utente=new.utente
               AND (a.inizio, a.fine) OVERLAPS
                   (new.inizio, new.fine));
```

if Error='True': errore e rollback
else: permetti op

T.validato_o_invalidato_dopo_creato
Insert o Update valida o invalida

```
Error=EXISTS(SELECT *
             FROM Alimento a
             WHERE new.alimento=a.id AND a.ist > new.ist);
```

if Error='True': errore e rollback
else: permetti op

T.no_intersezione_dieta
op: Insert o Update DurataDieta

nota : OVERLAPS può essere usato anche se una delle due date è NULL

```
Error= EXISTS (SELECT *
               FROM DurataDieta d WHERE d.utente=new.utente
               AND (d.inizio,d.fine) OVERLAPS (new.inizio,new.Fine));
```

if Error='True': errore e rollback
else: permetti op

[continua alla pagina seguente]

Risposta alla Domanda 7 (segue)

T.nel_pasto_se_valido_e_creato
op: Insert o Update pas_ali

```
OK=EXISTS(SELECT *
          FROM Pasto p, Utente u, Alimento a,
          valida left outer JOIN Utente u1 on valida.utente=u1.id
          WHERE new.pasto=p.id AND a.istante <= p.giorno
          AND p.utente=u.id AND new.alimento= a.id
          AND((valido(a.id, pasto.giorno)='True') OR
          (a.utente=u.id) OR
          (invalido(a.id,pasto.giorno)='False' AND u1.id = u.id)));
```

if OK=True : permetti
else: errore e rollback

T.no_auto_validazione
op: Insert o Update valida

```
Error=EXISTS (SELECT *
              FROM Alimento a
              WHERE new.alimento= a.id
              AND a.utente=new.utente);
```

if Error='True': errore e rollback
else: permetti op

T.validato_se_utente
op: Insert o Update valida

```
OK=EXISTS(SELECT *
          FROM Alimento a
          WHERE a.id=new.alimento AND a.utente IS NOT NULL);
```

if OK=True : permetti
else: errore e rollback

T.invalidato_dopo_validazione
op: Insert o Update invalida

```
Q= SELECT count(*);
   FROM valida v
   WHERE new.alimento=v.alimento
   AND v.ist <= new.ist
```

if Q>=10 : permetti
else: errore e rollback

[Versione 2024-06-05.UML]

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

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

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

Risposta

```
create function  invalido(d:Date, a:Integer):bool

    Q = SELECT count(*) as n
        FROM invalida
        WHERE ist <= d AND alimento = a;

    if Q.n >= 100 : result = True
    else : result = False
```

```
create function valido(d:Date, a:Integer):bool

    Q = EXISTS(SELECT * FROM Alimento WHERE id = a AND utente is not null);

    V = not invalido(d, a);

    nValid = SELECT count(*)
             FROM valida
             WHERE alimento = a AND ist <= d;

    if Q and nValid >= 10 and V : result = true
    else if not Q and V : result = true
    else result = False
```

USE CASE

alimento_in_pasto(tp:Stringa, a:Integer): <Int_GEZ, Real_GEZ>

- pre-cond: nessuna
- post-cond:

```
Result = SELECT count(*) AS num, AVG(pa.quantita)
         FROM pas_ali pa, Pasto p
         WHERE p.tipo = tp AND pa.pasto = p.id
         AND pa.alimento = a;
```

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

Risposta alla Domanda 8 (segue)

alimenti_comuni(tp:Stringa):Insieme(<Integer, Int_GE2, Real_GE2>)

- pre-cond: nessuna
- post-cond:

```
WITH massimo AS (SELECT max(num) as M FROM (SELECT alimento_in_pasto(tp,id)
                                            FROM Alimento a)),
     alimenti_occorrenze as (SELECT id,alimento_in_pasto(tp,id) FROM Alimento a)
SELECT * FROM alimenti_occorrenze oc, massimo m
WHERE oc.num = m.Mj
```

---

nutriente_rimanente(u:Integer, d:Date, n:Stringa):Real_GE2

- pre-cond:

```
EXISTS( SELECT * FROM Abbonamento WHERE utente=u
                                    AND inizio >= d
                                    AND Fine <= d)  AND
       EXISTS(SELECT * FROM DurataDieta WHERE utente = u
                                    AND inizio >= d
                                    AND ( Fine is NULL
                                    OR (Fine is NOT NULL AND Fine >= d)))
```

- post-cond:

```
M = ( SELECT dn.max as m
      FROM die_nut dn, DurataDieta DD
      WHERE dn.dieta = DD.dieta
      AND DD.utente = u AND dn.nutriente = n
      AND DD.inizio <= d AND (DD.fine IS NULL OR DD.fine >= d)
      AND dn.max IS NOT NULL)

T = (SELECT SUM(a.quantita * pa.quantita) as tot
     FROM Pasto p, Alimento a, pas_ali pa, nut_al na, DurataDieta DD
          die_nut dn
     WHERE pa.alimento = a.id AND pa.pasto = p.id
     AND na.alimento = a.id AND na.nutriente = n
     AND p.utente = u AND p.giorno = d
     AND dn.nutriente = n AND dn.dieta = DD.dieta
     AND DD.utente = u AND DD.inizio <= d
     AND (DD.fine IS NULL OR DD.fine >= d)
     AND dn.max IS NOT NULL)

Q = SELECT M.m - T.tot FROM M,T

if Q < 0 : result = 0
else : result = Q
```

Continua a minute 31

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.]

nutrienti_rimanenti(u:Integer, d:Date) : Insieme(<Stringa,Real_GE2>)

•pre-cond: EXISTS(SELECT * FROM Abbonamento WHERE utente=u
AND inizio >= d
AND Fine <= d) AND
EXISTS(SELECT * FROM DurataDieta WHERE utente=u
AND inizio>=d
AND (Fine is NULL
OR (Fine is NOT NULL AND Fine >= d)))

• post-cond: Result = SELECT nutriente_rimanente(u,d,n)
FROM Nutriente n;