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.

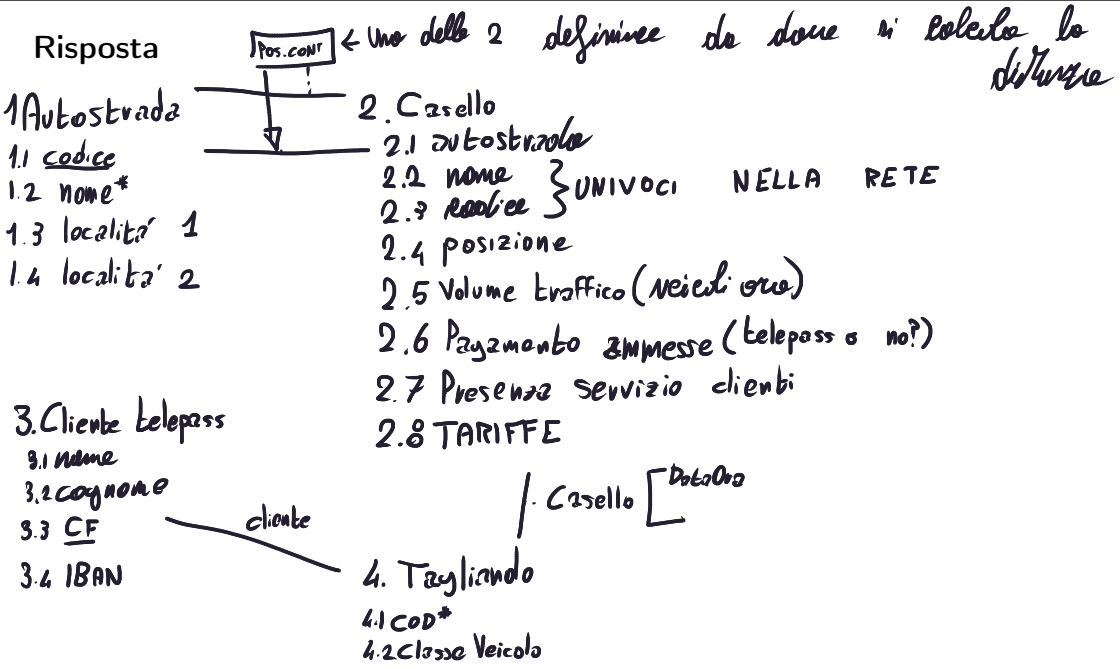
(*) Una risposta soddisfacente a questa domanda è condizione *necessaria* (ma non sufficiente) per superare la prova.

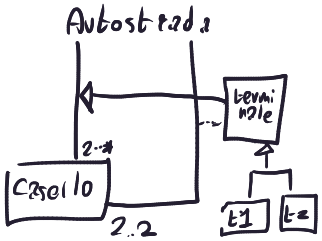
Questa pagina è stata intenzionalmente lasciata vuota

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





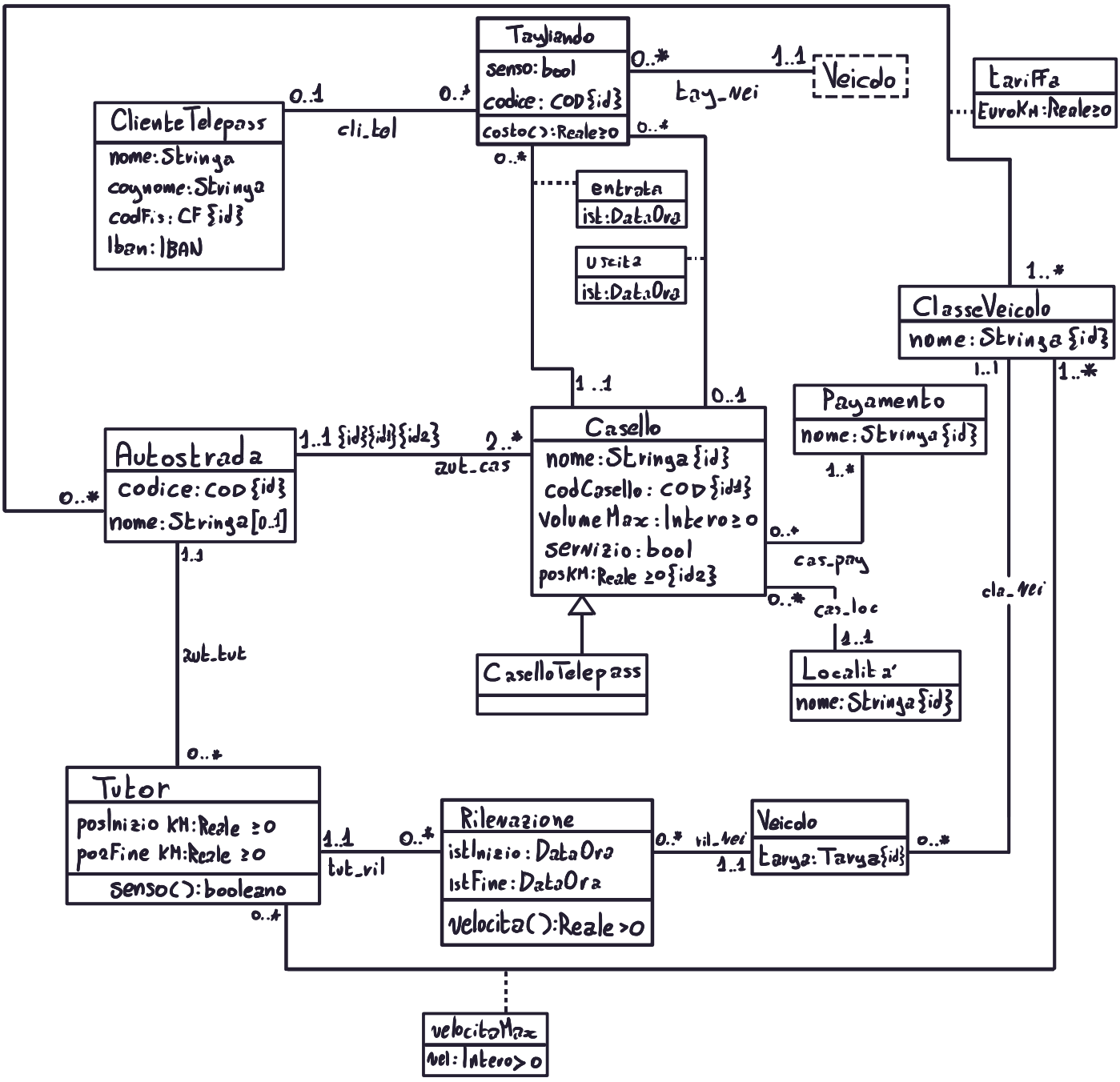
[continua alla pagina seguente]

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: Rilevazione .....

Operazioni, vincoli:

[V.coerenza_rilevazione] Senso-True

$\forall r, ir, Fr, v, t, it, Ft$ [Rilevazione(r) ∧ istInizio(ir) ∧ istFine(r,Fr) ∧ ril_vei(r,v) ∧ tut_ril(r,t) ∧ posInizioKM(t,it) ∧ posFineKM(t,Ft) ∧ senso(t,true)] →
[∃ tag, c, iec, pe
Tagliando(tag) ∧ senso(tag,true) ∧ tag_vei(tag,v) ∧ Casello(c) ∧ entrata(tag,c) ∧ iec ≤ ir ∧ ist(tag,c,iec) ∧ posKM(c,pe) ∧ pe ≤ it ∧ [∃ c2, iuc, pu Casello(c2) ∧ posKM(c2,pu) ∧ uscita(tag,c2) ∧ ist(tag,c2,iuc)] →
[iuc ≥ Fr ∧ pu ≥ Ft]]]

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

Nome: Rilevazione .....

Operazioni, vincoli:

[V.rilevato_veicolo_classe]

$\forall r, v, t, c$
[Rilevazione(r) ∧ ril_vei(r,v) ∧ cla_vei(v,c) ∧ tut_ril(t,r)] → velocitaMax(t,c)

[V.inizio_poi_fine]

$\forall r, i, f$ [Rilevazione(r) ∧ inizio(r,i) ∧ fine(r,f)] → i < f

3 Tipo: Classe | Associazione (cerchiare)

Nome: Tagliando

Operazioni, vincoli:

[V.senso_di_marcia]

$\forall t, ce, cu, pe, pu$
$[Tagliando(t) \wedge entrata(t, ce) \wedge uscita(t, cu) \wedge posKM(ce, pe) \wedge posKM(cu, pu)] \rightarrow$
$[[senso(t, True)] \leftrightarrow [pe < pu]]$

[V.tagliando_classe_veicolo]

$\forall t, v, c, a, cl [Tagliando(t) \wedge tag\_vei(t, v) \wedge [entrata(t, c) \vee uscita(t, c)] \wedge cla\_vei(v, cl) \wedge aut\_cas(a, c)] \rightarrow tariffa(a, cl)$

6 Tipo: Classe | Associazione (cerchiare)

Nome: Tagliando

Operazioni, vincoli:

[V.entra_poi_esce]

$\forall t, c1, c2, e, u [Tagliando(t) \wedge entrata(t, c1) \wedge uscita(t, c2) \wedge ist(t, c2, u) \wedge ist(t, c1, e)] \rightarrow e < u$

[V.casello_telepass]

$\forall t, c$
$[Tagliando(t) \wedge [entrata(t, c) \vee uscita(t, c)] \wedge \exists cl \; cli\_tal(cl, t)] \rightarrow CaselloTelepass(c)$

4 Tipo: Classe | Associazione (cerchiare)

Nome: Tagliando

Operazioni, vincoli:

costo( ): Reale ≥ 0

- pre-cond: $\exists u$ uscita(this, u)
- post-cond: Sia tar :=

$\exists c, a, v, cl$ tag_vei(this, v) $\wedge$ [entrata(this, c) $\vee$ uscita(this, c)] $\wedge$ aut_cas(c, a) $\wedge$ tariffa(a, cl) $\wedge$ cla_vei(cl, v) $\wedge$ euroKM(a, cl, tar)

Siano p1, p2 :=

$\exists c1, c2$ entrata(this, c1) $\wedge$ posKM(c1, p1) $\wedge$ uscita(this, c2) $\wedge$ posKM(c2, p2)

Result = |p1 − p2| · tar

7 Tipo: Classe | Associazione (cerchiare)

Nome: Tutor

Operazioni, vincoli:

senso( ): bool

- pre-cond: nessuna
- post-cond: Siano i, F :=

posInizioKM(this, i) $\wedge$ posFineKM(this, F)

i < F → Result = True $\wedge$

i > F → Result = False

[V.compreso_in_autostrada]

$\forall t, a, et, ut [Tutor(t) \wedge posInizioKM(t, et) \wedge posFineKM(t, ut) \wedge aut\_tut(a, t)] \rightarrow [\exists c1, c2, e, u \; aut\_cas(a, c1) \wedge aut\_cas(a, c2) \wedge posKM(c1, e) \wedge posKM(c2, e) \wedge e \leq et \wedge u \geq ut]$

5 Tipo: Classe | Associazione (cerchiare)

Nome: Tutor

Operazioni, vincoli:

[V.no_intersezioni_tutor]

$\forall t1, t2, e1, u1, e2, u2, a [Tutor(t1) \wedge Tutor(t2) \wedge aut\_tut(a, t1) \wedge aut\_tut(a, t2) \wedge posInizioKM(t1, e1) \wedge posFineKM(t1, u1) \wedge posInizioKM(t2, e2) \wedge posFineKM(t2, u2) \wedge u1 > e1 \wedge u2 > e2 \wedge senso(t1, true) \wedge senso(t2, true)] \rightarrow [u2 < e1 \vee u1 < e2]$

$\forall t1, t2, e1, u1, e2, u2, a [Tutor(t1) \wedge Tutor(t2) \wedge aut\_tut(a, t1) \wedge aut\_tut(a, t2) \wedge posInizioKM(t1, e1) \wedge posFineKM(t1, u1) \wedge posInizioKM(t2, e2) \wedge posFineKM(t2, u2) \wedge u1 < e1 \wedge u2 < e2 \wedge senso(t1, false) \wedge senso(t2, false)]$
$\rightarrow [u2 > e1 \vee u1 > e2]$

8 Tipo: Classe | Associazione (cerchiare)

Nome: Rilevazione

Operazioni, vincoli:

velocita( ): Reale ≥ 0

- pre-cond: nessuna
- post-cond:

Siano pi, pF, ii, iF tali da soddisfare:

istInizio(this, ii) $\wedge$ istFine(this, iF) $\wedge \exists t$ tut_ril(this, t) $\wedge$ posInizioKM(t, pi) $\wedge$ posFineKM(t, pF)

Sia σ : oreTrascorse(ii, iF, 0) (semantica mondo reale)

Result = $\frac{|pi - pF|}{\sigma}$

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

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

[V.no_tagliandi_che_si_intersecano]

$\forall v,t1,t2,e1,e2\,[Tagliand(t1)\wedge Tagliando(t2)\wedge tag\_vei(t1,v)\wedge tag\_vei(t2,v)\wedge \exists c\ entrata(t1,c)\wedge t1\neq t2 \wedge ist(t1,c,e1)\wedge \exists k\ entrata(t2,k)\wedge ist(t2,k,e2)] \rightarrow \neg[t\ \ t\geq e1\wedge[\forall u,tu\ uscita(t1,u)\wedge ist(t1,u,tu) \rightarrow tu\geq t]\wedge t\geq e2\wedge[\forall u,tu\ uscita(t2,u)\wedge ist(t1,u,tu)\rightarrow tu\geq t]]$

[V.no_rilevazioni_che_si_intersecano]

$\forall r1,r2,i1,i2,f1,f2,v\ \ [Rilevazione(r1)\wedge Rilevazione(r2)\wedge istInizio(r1,i1)\wedge istInizio(r2,i2)\wedge istFine(r1,f1)\wedge istFine(r2,f2)\wedge ril\_vei(r1,v)\wedge ril\_vei(r2,v)\wedge r1\neq r2\ \ ]\rightarrow[f1<i2\vee f2<i1]$

[V.no_tagliandi_che_si_intersecano_cli]

$\forall c,t1,t2,e1,e2\,[Tagliand(t1)\wedge Tagliando(t2)\wedge cli\_tal(t1,c)\wedge cli\_tal(t2,c)\wedge \exists c\ entrata(t1,c)\wedge t1\neq t2 \wedge ist(t1,c,e1)\wedge \exists k\ entrata(t2,k)\wedge ist(t2,k,e2)] \rightarrow \neg[t\ \ t\geq e1\wedge[\forall u,tu\ uscita(t1,u)\wedge ist(t1,u,tu) \rightarrow tu\geq t]\wedge t\geq e2\wedge[\forall u,tu\ uscita(t2,u)\wedge ist(t1,u,tu)\rightarrow tu\geq t]]$

[V.tagliando_caselli_stessa_autostrada]

$\forall t,e,u\ [Tagliando(t)\wedge entrata(t,e)\wedge uscita(t,u)]\rightarrow \exists a\ aut\_cas(a,e)\wedge aut\_cas(a,u)$

[V.coerenza_rilevazione] Senso - False

$\forall r,ir,fr,v,t,it,Ft\ \ [Rilevazione(r)\wedge istInizio(ir)\wedge istFine(r,Fr)\wedge ril\_vei(r,v)\wedge tut\_ril(r,t)\wedge posInizioKM(t,it)\wedge posFineK(t,Ft)\wedge senso(t,False)]\rightarrow$
$[\exists tag,c,iec,pe\ Tagliando(tag)\wedge senso(tag,False)\wedge tag\_vei(tag,v)\wedge Casello(c)\wedge entrata(tag,c)\ \ \wedge iec\leq ir\wedge \wedge ist(tag,c,iec)\wedge posKM(c,pe)\wedge pe\geq it \wedge[[\exists c2,iuc,pu\ \ Casello(c2)\wedge posKM(c2,pu)\wedge uscita(tag,c2)\wedge ist(tag,c2,ivc)]\rightarrow [iuc\geq Fr\wedge pu\leq Ft]]]$

Tipi di Dato

COD = [A-Z0-9]+

IBAN = [A-Z]{2}[0-9]{2}[A-Z][0-9]{22}

CF = [A-Z]{6}[0-9]{2}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]

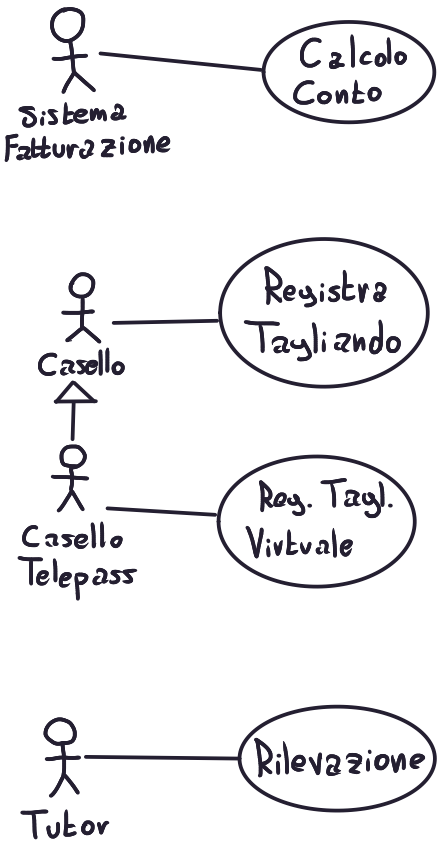
Targa = [A-Z]{2}[0-9]{3}[A-Z]{2}

[continua alla pagina seguente]

Risposta alla Domanda 2 (segue)

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

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

calcola_conto(c: ClienteTelepass, m: 1..12, anno: Intero≥0): (Stringa, Stringa, DataOra, DataOra, Reale≥0) [0..*]

- pre-cond: nessuna
- post-cond:

$$C = \left\{ (ne, nu, de, du, im) \;\middle|\; \begin{array}{l} \exists t\ \text{Taglicando}(t) \wedge \text{cli\_tal}(c,t) \wedge \exists ce, cu\ \ \text{entrata}(t, ce) \\ \wedge \text{uscita}(t, cu) \wedge \text{nome}(ce, ne) \wedge \text{nome}(cu, nu) \wedge \\ \text{ist}(t, ce, de) \wedge \text{ist}(t, cu, du) \wedge \text{Mese}(de, m) \wedge \text{Mese}(du, m) \\ \wedge \text{Anno}(de, anno) \wedge \text{Anno}(du, anno) \wedge \text{costo}(t, im) \end{array} \right\}$$

Result = C

---

rileva_passaggio(t: Tutor, i: DataOra, f: DataOra, v: Veicolo): Rilevazione

- pre-cond: $\exists cl$ ClasseVeicolo(cl) $\wedge$ cla_vei(v, cl) $\wedge$ velocitaMax(t, cl)
- post-cond: Sia α un nuovo oggetto del dominio tale che:

  Rilevazione(α) $\wedge$ istInizio(α, i) $\wedge$ istFine(α, f) $\wedge$ ril_vei(α, v) $\wedge$ tut_ril(α, t)

  Sia vMax tale che

  $\exists cl$ ClasseVeicolo(cl) $\wedge$ cla_vei(v, cl) $\wedge$ velocitaMax(t, cl) $\wedge$ vel(t, cl, vMax)

  Sia vel tale che : velocita(α, vel)

  Se vel $\leq$ vMax : Termina operazione ed α viene cancellato.

  Se vel > vMax : α e' un nuovo oggetto del dominio: $M_{out} = M_{in} \cup \{\alpha\}$

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

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 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.

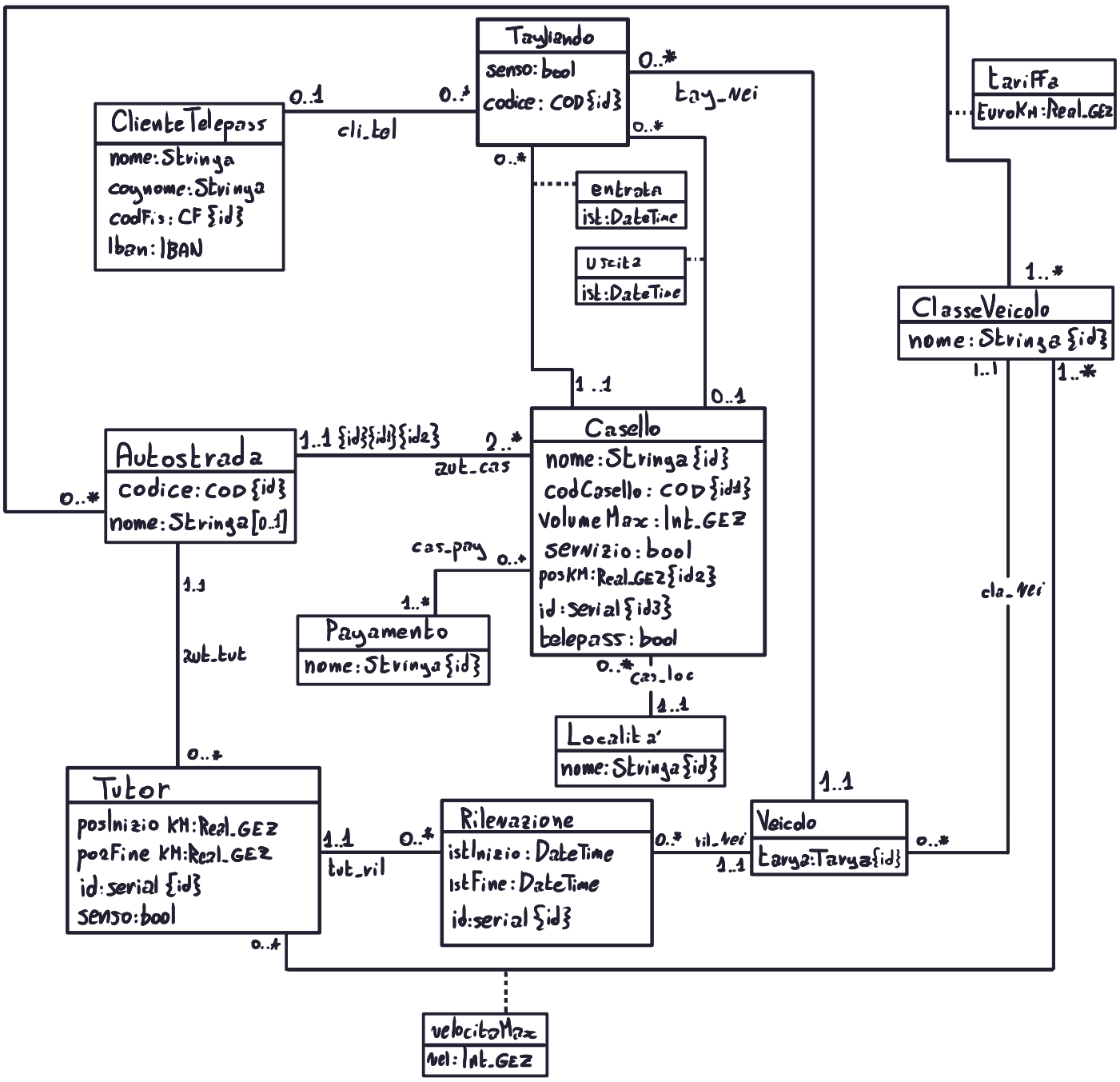
DBMS da utilizzare ..PostgreSQL.....

Corrispondenza tra tipi di dato concettuali e domini supportati dal DBMS

```
create domain COD as varchar ~ '[A-Z0-9]+'
create domain IBAN as varchar ~ '[A-Z]{2}[0-9]{2}[A-Z][0-9]{22}'
create domain Real_GEZ as Real check(value >= 0);
create domain Int_GEZ as Integer check(value>=0);
create domain CF as varchar ~ '[A-Z]{6}[0-9]{2}[A-Z][0-9]{12}[A-Z][0-9]{3}[A-Z]'

create domain Stringa as varchar NOT NULL;

create domain Targa as varchar ~ '[A-Z]{2}[0-9]{3}[A-Z]{2}'

create domain Mese as Integer check(value>=1 AND value<=12);
```

[Versione 2024-06-05.UML]

Diagramma UML delle classi ristrutturato



[Versione 2024-06-05.UML]

Breve descrizione delle scelte effettuate durante la ristrutturazione

Fusione su Casello

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.casello_telepass]
$\forall t, c$
$[\text{Tagliando}(t) \wedge [\text{entrata}(t.c) \vee \text{uscita}(t.c)]] \wedge \text{Casello}(c) \wedge \exists cl \ \ cli\_tal(cl,t)] \rightarrow$
$[\text{telepass}(c, \text{True})]$

[continua alla pagina seguente]

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 UML delle classi ristrutturato.

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

1 **Relazione** ClasseVeicolo (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** Veicolo (nome) Derivante da: (classe) | **associazione** (cerchiare)

Attributi	Targa	classe					
Domini	Targa	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 classe ref ClasseVeicolo(nome);

La relazione accorpa le relazioni che implementano le seguenti associazioni: cla-vei

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

Attributi	codice	nome*					
Domini	COD	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: .....

4 **Relazione** Localita (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: .....

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

Attributi	nome	codCasello	volumeMax	servizio	posKM	id	telepass	localita
Domini	Stringa	COD	Int-GEZ	bool	Real-GEZ	serial	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): Fk localita ref Localita(nome);

V.inclusione:= Casello(id) ⊆ cas_pay(casello);

La relazione accorpa le relazioni che implementano le seguenti associazioni: cas-loc

[Versione 2024-06-05.UML]

6 **Relazione** aut_cas (nome) Derivante da: **classe** | **associazione** (cerchiare)

Attributi	autostrada	casello						
Domini	COD	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):

V.Inclusione : Casello(id) ⊆ aut_cas(casello);

V.inclusione: Autostrada(codice) ⊆ aut_cas(autostrada); 2 VOLTE MINIMO NON SO COME DIRLO

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

7 **Relazione** Pagamento (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** cas_pag (nome) Derivante da: **classe** | ~~**associazione**~~ (cerchiare)

Attributi	casello	pagamento						
Domini	Integer	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 casello ref Casello(id);

Fk pagamento ref Pagamento(nome);

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

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

Attributi	nome	cognome	codFis	iban				
Domini	Stringa	Stringa	CF	IBAN				

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: .....

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

Attributi	senso	codice	cliente *	veicolo	entrata	istE	uscita *	istU *
Domini	bool	COD	CF	Targa	Integer	DateTime	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):

Fk cliente ref ClienteTelepass(codFis); Fk veicolo ref Veicolo(targa); Fk uscita ref Casello(id);

check((istU is NULL AND uscita is NULL) OR (istU is NOT NULL AND uscita is NOT NULL)); Fk entrata ref Casello(id); check(istE < istU);

La relazione accorpa le relazioni che implementano le seguenti associazioni: cli_tel, tag_vei, entrata, uscita

[Versione 2024-06-05.UML]

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

Attributi	autostrada	classe	EuroKM					
Domini	COD	Stringa	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 autostrada ref Autostrada(codice);

FK classe ref ClasseVeicolo(nome);

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

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

Attributi	posInizioKM	posFineKM	id	autostrada	senso			
Domini	Real-GEZ	Real-GEZ	serial	COD	bool			

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 autostrada ref Autostrada(codice); check(posInizioKM <> posFineKM);

check((senso=True AND posInizioKM < posFineKM) OR ((senso=False AND posInizioKM > posFineKM);

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

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

Attributi	id	istInizio	istFine	tutor	veicolo			
Domini	serial	DateTime	DateTime	Integer	Targa			

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(istInizio < istFine); FK tutor ref Tutor(id); FK veicolo ref Veicolo(targa);

La relazione accorpa le relazioni che implementano le seguenti associazioni: tut_ril, ril_vei .....

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

Attributi	tutor	classe	vel					
Domini	Integer	Stringa	Int-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 tutor ref Tutor(id);

FK classe ref ClasseVeicolo(nome);

V.Inclusione:= Tutor(id) ⊆ velocitaMax(tutor);

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

15 **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: .....

16 **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: .....

17 **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: .....

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: .....

[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. coerenza_rilevazione
op: Insert o Update su Rilevazione

OK = EXISTS( SELECT *
             FROM Tagliando t, Casello c_ent, aut_cas ac, Tutor tut, Casello c_usc
             WHERE new.veicolo = t.veicolo AND t.entrata = c_ent.id  AND ac.casello = c_ent.id
             AND  ac.autostrada = tut.autostrada  AND tut.id = new.tutor
             AND  t.senso = tut.senso  AND new.istInizio >= t.istE  AND
             (( t.senso = True AND  c_en.posKM < tut.posInizioKM)  OR
             (t.senso = False AND  c_en.posKM > tut.posInizioKM))
             AND( t.uscita  IS  NULL  OR  ( t.uscita = c_usc.id AND new.istFine <= t.istU
             AND((t.senso=True  AND  c_usc.posKM > tut.posFineKM) OR
                  (t.senso=False  AND  c_usc.posKM < tut.posFineKM)))));

if OK = True: commit
else: errore e rollback
```

```
T. senso_di_marcia
Insert o Update Tagliando

Error = EXISTS (SELECT  *
                FROM  Casello e, Casello u
                WHERE  new.entrata = e.id AND  new.uscita = u.id
                AND ((new.senso = True  AND  u.posKM < e.posKM)
                 OR ( new.senso = False AND  u.posKM > e.posKM));

if Error: rollback
else: commit
```

```
T. no_tagliandi_che_si_intersecano
Insert o Update Tagliando

Error = EXISTS( SELECT * FROM Tagliando t
                WHERE (t.cliente = new.cliente OR t.veicolo = new.veicolo)
                AND (new.istE, new.istU) OVERLAPS (t.istE, t.istU));

if Error: rollback
else: commit
```


[continua alla pagina seguente]

Risposta alla Domanda 7 (segue)

T. Casello_telepass
Insert o Update Tagliando

```
OK = EXISTS (SELECT *
             FROM Casello ce, Casello cu
             WHERE (new.entrata = ce.id AND
             ce.telepass=True AND (new.uscita is False
             OR (new.uscita=cu.id AND cu.telepass=True)))
             OR new.cliente IS NULL);

if OK=True: commit
else: errore e rollback
```

T. Tagliando_classe_veicolo
Insert o update Tagliando

```
OK= EXISTS (SELECT *
            FROM Casello c, tariffa t, aut_cas ac, Veicolo v
            WHERE c.id=new.entrata
            AND ac.casello = c.id AND ac.autostrada=t.autostrada
            AND t.classe =v.classe AND v.targa = new.veicolo);

if OK=True: commit
else: errore e rollback
```

T. no_intersezioni_tutor
Insert or Update Tutor

```
Error= EXISTS (SELECT * FROM Tutor t WHERE t.autostrada = new.autostrada AND t.senso=new.senso
                AND (new.posInizioKM, new.posFineKM) OVERLAPS
                    (t.posInizioKM, t.posFineKM));

if Error: vollback
else: commit
```

T. no_rilevazioni_che_si_intersecano
Insert o Update Rilevazione

```
Error = EXISTS(SELECT * FROM Rilevazione r WHERE r.veicolo = new.veicolo AND
          (new.istInizio, new.istFine) OVERLAPS (r.istInizio, r.istFine));

if Error: vollback
else: commit
```

[Versione 2024-06-05.UML]

Risposta alla Domanda 7 (segue)

T. tagliando_caselli_stessa_autostrada
Insert o Update Tagliando

```
Error = EXISTS( SELECT
            FROM Casello e, Casello u, aut_cas ace, aut_cas acu
            WHERE e.id=new.entrata AND
            u.id = new.uscita AND ace.casello = e.id AND
            acu.casello = u.id AND ace.autostrada <> acu.autostrada);
```

if Error: vollback
else: commit

T. compreso_in_autostrada
Insert o Update Tutor

```
OK=EXISTS(SELECT
          FROM Casello e, Casello u, aut_cas ce, aut_cas cu
          WHERE ce.autostrada = cu. autostrada = new. autostrada
          AND ce.casello=e.id AND cu.casello = u.id AND
          AND ((new.senso = True AND e.posKM <= new.posInizioKM AND u.posKM >= new.posFineKM)
          OR (new.senso = False AND e.posKM >=new.posInizio KM AND u.posKM <= new.posFineKM)));
```

if OK=True: commit
else: errore e vollback

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 costo(ty: COD): Real_GEZ

    Error = EXISTS (SELECT * FROM Tagliando WHERE id=ty AND uscita IS NULL);

    IF (Error = true): termina operazione

    Q = SELECT (tariffa.euroKM * ABS(e.posKM - u.posKM)
        FROM Casello e, Casello u, Tagliando t, Autostrada a, Veicolo V, ClasseVeicolo Vc, tariffa, aut_cas
        WHERE t.codice = ty AND t.entrata = e.id AND t.uscita = u.id
        AND V.targa = t.veicolo AND Vc.nome = V.classe AND tariffa.classe = V.classe
        AND e.id = aut_cas.casello AND aut_cas.autostrada = a.codice;

    result = Q
```

```
create function velocita(ril: Integer): Real_GEZ

    V = SELECT ABS(t.posKMInizio - posKMFine)/(EXTRACT(EPOCH FROM(r.istFine - r.istInizio))/3600)
        FROM Rilevazione r, Tutor t
        WHERE r.id = ril AND t.id = r.tutor

    result = V
```

[Versione 2024-06-05.UML]

[continua alla pagina seguente]

Risposta alla Domanda 8 (segue)

```
calcola_conto(cl:CF, m:Mese, anno:Intero>=0):Insieme(<Stringa,Stringa,DateTime,DateTime,Real_GEZ>)

    Q = SELECT e.nome,u.nome,istE,istU,costo(t.cod)
        FROM Tagliando t, Casello e, Casello u
        WHERE t.cliente = cl AND
        EXTRACT('month' FROM istE) = m AND EXTRACT('month' FROM istU) = m AND
        EXTRACT('year' FROM istE) = anno AND EXTRACT('year' FROM istU) = anno AND
        AND t.entrata=e.id AND t.uscita=u.id;

    result = Q
```

```
rileva_passaggio(tut:Integer, i:DateTime, f:DateTime, vt:Targa):Integer[0..1]

    OK = EXISTS (SELECT *
                 FROM velocitaMax vm, Veicolo v
                 WHERE vm.tutor=tut AND vm.classe=v.classe AND v.targa=vt)

    if(OK=False):termina operazione

    ril = Insert INTO Rilevazione(i, f, tut, vt) returning id;

    D = EXISTS( WITH V as (SELECT vel FROM velocitaMax vm, Veicolo v
                           WHERE v.targa=vt AND v.tutor=tut)

                SELECT *
                FROM Rilevazione r, V WHERE V.vel >= velocita(r.id)
                AND r.id=ril);

    iF (D==True): DELETE FROM Rilevazione WHERE id=ril;
    else: result=ril
```

[Versione 2024-06-05.UML]