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: Casu Marco Matricola: .....

Data: 13/06/2024

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: .....

Città Verde

Sulle 5 ore e 30 minuti

Questo modulo è ottimizzato per la stampa fronte-retro

1 Analisi concettuale

Domanda 1 (10 minuti) Raffinare la specifica dei requisiti eliminando inconsistenze, omissioni e ridondanze e producendo un elenco numerato di requisiti il meno ambiguo possibile. (La risposta a questa domanda non sarà valutata, ma si consiglia di svolgere accuratamente questo passo, in quanto può facilitare di molto le attività di progetto.)

Risposta

1. Soggetto
   - 1.1 data piantumazione
   - 1.2 posizione (lat-long)
   - 1.3 specie {nome scient. e comune}
   - 1.4 rischio
   - 1.5 dimensioni (da definire)
   - 1.6 Rimosso?
     - 1.6.1 data rimozione
     - 1.6.2 causa

V. rimozione ≥ piantagione

2. Area Verde
   - 2.1 denominazione
   - 2.2 soggetti
   - 2.3 fruibile/non fruibile
     - 2.3.1 Se Fruibile — {sensibile, non sensibile}

3. Operatore
   - 3.1 nome
   - 3.2 cognome
   - 3.3 data inizio servizio
   - 3.4 data fine // V. fine > inizio [0..1]
   - 3.5 CF

V. operatore della squadra in cui opera

4. Squadra
   - 4.1 Fondazione
   - 4.2 scioglimento [0..1]
   - 4.3 codice
   - 4.4 membri
   - 4.5 CAPO
   - 4.6 attrezzature (Req. 5)

Operatore — PARTECIPAZIONE (inizio, fine) — squadra (0..*)

V. i

Attrezzatura (0..*) — V. uso att. della squadra

5 Attrezzatura
- 5.1 tipologia
- 5.2 nome

6. Intervento
   - 6.1 area
   - 6.2 inizio
   - 6.3 durata stim.
   - 6.4 priorità
   - 6.5 attività
   - 6.6 piante interessate [0..*]
   - 6.7 attrezzature necessarie
   - 6.8 num operatori richiesti
   - 6.7 Squadra assegnata
   - 6.8 istante esecuzione
   - 6.9 ist completamento [0..1]

V. piante dell'intervento devono essere parte dell'area verde

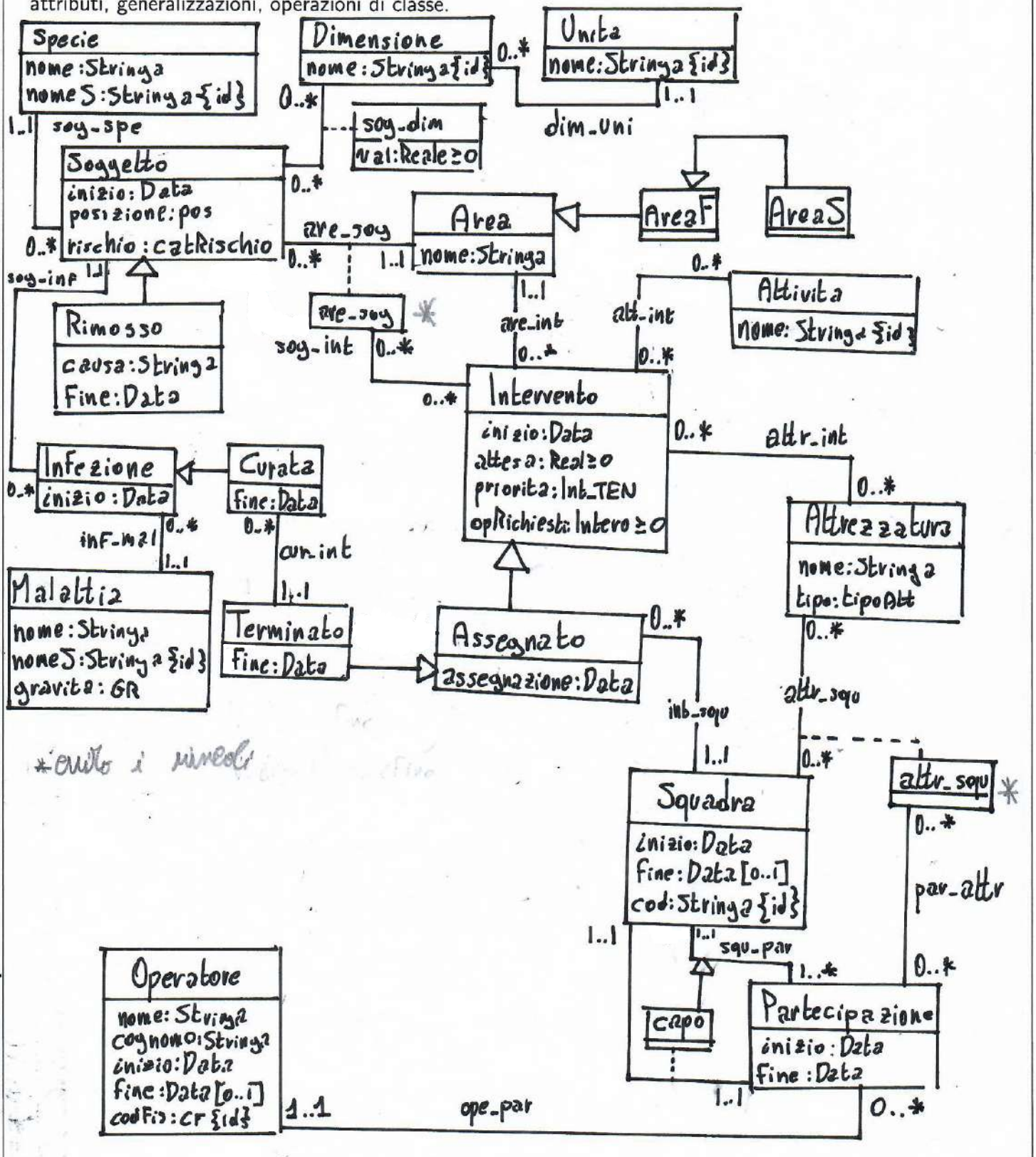
[continua alla pagina seguente]

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.



[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: Partecipazione

Operazioni, vincoli:

[V.partecipa_se_opera]

$\forall p, ip, Fp, \sigma, io$ [Partecipazione$(p) \wedge$ inizio(p, ip) $\wedge$ fine$(p, Fp) \wedge$ ope_par$(\sigma, p) \wedge$ inizio(σ, io)] $\rightarrow io \leq ip \wedge$ [$\forall Fo$ fine$(\sigma, Fo) \rightarrow Fo \geq Fp$]

[V.partecipa_se_squadra]

Simile a [V.partecipa_se_opera] ma con la partecipazione compresa nell'esistenza della squadra.

2 Tipo: (Classe) | Associazione (cerchiare)

Nome: Infezione

Operazioni, vincoli: [V.infezione_curata] (R)

$\forall s, inf, int$

[Curata$(inf) \wedge$ cur_int$(inf, int) \wedge$ sog_inf(inf, s)] $\rightarrow \exists a$ ave_sog$(a, s) \wedge$ sog_int(a, s, int)

[V.infetto_se_esistente]

$\forall s, inf, is, ii$ [Soggetto$(s) \wedge$ sog_inf$(inf, s) \wedge$ inizio$(inf, ii) \wedge$ inizio(s, is)] $\rightarrow ii \geq is \wedge$ [$\forall Fs, Fi$ fine$(inf, Fi) \wedge$ fine$(s, Fs) \rightarrow Fs \geq Fi$]

SULLA FINE, NO VINCOLI, UN INTERVENTO PUO' RIMUOVERE

3 Tipo: (Classe) | Associazione (cerchiare)
Nome: Intervento
Operazioni, vincoli:
[V. tempo_assegnazione]
$\forall in, i, a$ [Intervento(in) $\wedge$ inizio(in, i) $\wedge$ assegnazione(in, a)] $\rightarrow a \geq i \wedge$ [$\forall F$ Fine(in, F) $\rightarrow F \geq a$]
[V. soggetti_esistenti_intervento]
$\forall in, s, ii, is, a$ [Intervento(in) $\wedge$ Soggetto(s) $\wedge$ are_sog(a, s) $\wedge$ sog-int(a, s, in) $\wedge$ inizio(s, is) $\wedge$ inizio(in, ii)] $\rightarrow is \leq ii$

6 Tipo: (Classe) | Associazione (cerchiare)
Nome: Intervento
Operazioni, vincoli:
[V. prende_squadra_esistente]
$\forall in, s, as, is$ [$\wedge$ Squadra(s) $\wedge$ inizio(s, is) $\wedge$ assegnazione(in, as) $\wedge$ int_squ(in, s)] $\rightarrow as \geq is \wedge$ [$\forall Fs, Fi$ Fine(in, Fi) $\wedge$ Fine(s, Fs) $\rightarrow Fs \geq Fi$]

4 Tipo: (Classe) | Associazione (cerchiare)
Nome: Squadra
Operazioni, vincoli:
[V. no_part_intersecate_stesso_op]
$\forall p1, p2, s, o, i1, i2, F1, F2$
[Partecipazione(p1) $\wedge$ Partecipazione(p2) $\wedge$ ope_par(o, p1) $\wedge$ ope_par(o, p2) $\wedge$ $p1 \neq p2$ squ_par(s, p1) $\wedge$ squ_par(s, p2) $\wedge$ inizio(p1, i1) $\wedge$ inizio(p2, i2) $\wedge$ fine(p1, F1) $\wedge$ fine(p2, F2)] $\rightarrow F1 < i2 \vee F2 < i1$

7 Tipo: (Classe) | Associazione (cerchiare)
Nome: Squadra
Operazioni, vincoli:
[V. squadra_qualificata]
$\forall in, s, at, ii$ [Assegnato(in) $\wedge$ int_squ(in, s) $\wedge$ assegnazione(in, ii) $\wedge$ altr_int(at, in)] $\rightarrow$ [$\exists p, ip, fp$ Partecipazione(p) $\wedge$ inizio(p, ip) $\wedge$ fine(p, fp) $\wedge$ squ_par(s, p) $\wedge ii \geq ip \wedge$ [$\forall F$ fine(in, F) $\rightarrow F \leq Fp$] $\wedge$ altr_squ(at, s) $\wedge$ par_attr(p, at, s)

5 Tipo: (Classe) | Associazione (cerchiare)
Nome: Squadra
Operazioni, vincoli:
[V. squadra_membri_minimi]
identico a Use case
cerca_squadre_qualificate

8 Tipo: (Classe) | Associazione (cerchiare)
Nome: Malattia
Operazioni, vincoli:
[V. no_inf_cont_stessa_malattia]
$\forall inF1, inF2, s, m, i1, i2$ [Infezione(inF1) $\wedge$ Infezione(inF2) $\wedge$ inF_mal(inF1, m) $\wedge$ inF_mal(inF2, m) $\wedge$ inizio(inF1, i1) $\wedge$ inizio(inF2, i2) $\wedge$ sog-inF(inF1, s) $\wedge$ sog-inF(inF2, s)] $\rightarrow$ [$\forall F$ fine(inF2, F) $\rightarrow F < i2$] $\wedge$ [$\forall F$ fine(inF2, F) $\rightarrow F < i1$]

[continua alla pagina seguente]

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

[V. continuità]

$\forall k,i,f[[Soggetto(k) \vee Infezione(k) \vee Intervento(k) \vee Squadra(k) \vee Partecipazione(k) \vee Operatore(k)] \wedge inizio(k,i) \wedge Fine(k,F)] \rightarrow i \leq F$

Tipi di Dato

pos = (lat: Reale, long: Reale)

catRischio = {'b', 'm', 'a'}

GR = {'b', 'm', 'a', 'mort'}

Int_TEN = 1..10

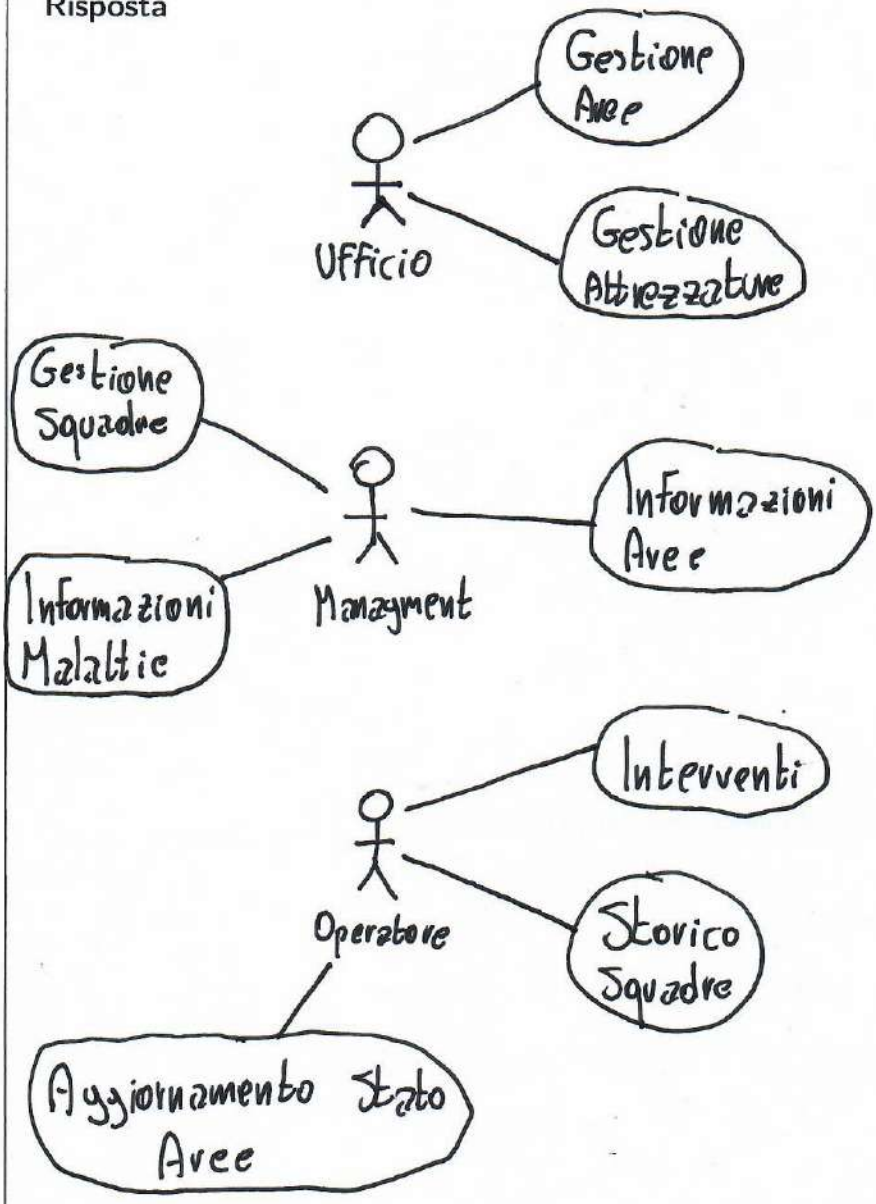
tipoAtt = {'I', 'V', 'VS'}

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

[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



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

aree_sensibili_senza_interventi(i:Data, f:Data) : AreaS [0..*]

- pre-cond: $i \leq f$
- post-cond: $A = \{ a \mid \text{AreaS}(a) \wedge [\forall in, ii \ \ are\_int(a, in) \wedge inizio(in, ii) \rightarrow F < ii \wedge [\forall fi \ \ Fine(in, fi) \rightarrow fi < i]] \}$

cerca_squadre_qualificate(in:Intervento): Squadra [0..*]

- pre-cond: $\neg$Assegnato(in)
- post-cond:

$$S = \{ s \mid \exists is, ii \ \ inizio(s, is) \wedge inizio(in, ii) \wedge is \leq ii \wedge \text{Squadra}(s) \wedge [\forall at \ \ altr\_int(at, in) \rightarrow \exists p, ip \ \ Partecipazione(p) \wedge squ\_par(s, p) \wedge inizio(p, ip) \wedge ip \leq ii \wedge altr\_squ(at, s) \wedge par\_attr(at, s, p)] \wedge \exists n, m \ \ opRichiesti(in, m) \wedge n = |\{ pa \mid Partecipazione(pa) \wedge squ\_par(s, p) \wedge \exists ip2 \ inizio(pa, ip2) \wedge ip2 \leq ii \}| \wedge m \leq n \}$$

Result = S

tasso_malattie(i:Data, f:Data, M: Malattia [1..*]): (Malattia, Reale≥0) [0..*]

- pre-cond: $i \leq f$
- post-cond:

$$T = \left\{ \left(m, \frac{k}{n}\right) \mid m \in M \wedge n = |\{ s \mid \exists inF, iF \ \ Infezione(inF) \wedge inizio(inF, iF) \wedge i \leq iF \leq f \wedge \text{Soggetto}(s) \wedge sog\_inf(inF, s) \}| \wedge k = |\{ s \mid \exists inF, iF \ \ Infezione(inF) \wedge inizio(inF, iF) \wedge i \leq iF \leq F \wedge \text{Soggetto}(s) \wedge sog\_inf(inF, s) \wedge inf\_mal(inF, m) \}| \right\}$$

Result = T

FINITO IN 2 ORE E 40 MINUTI

[continua alla pagina seguente]

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.

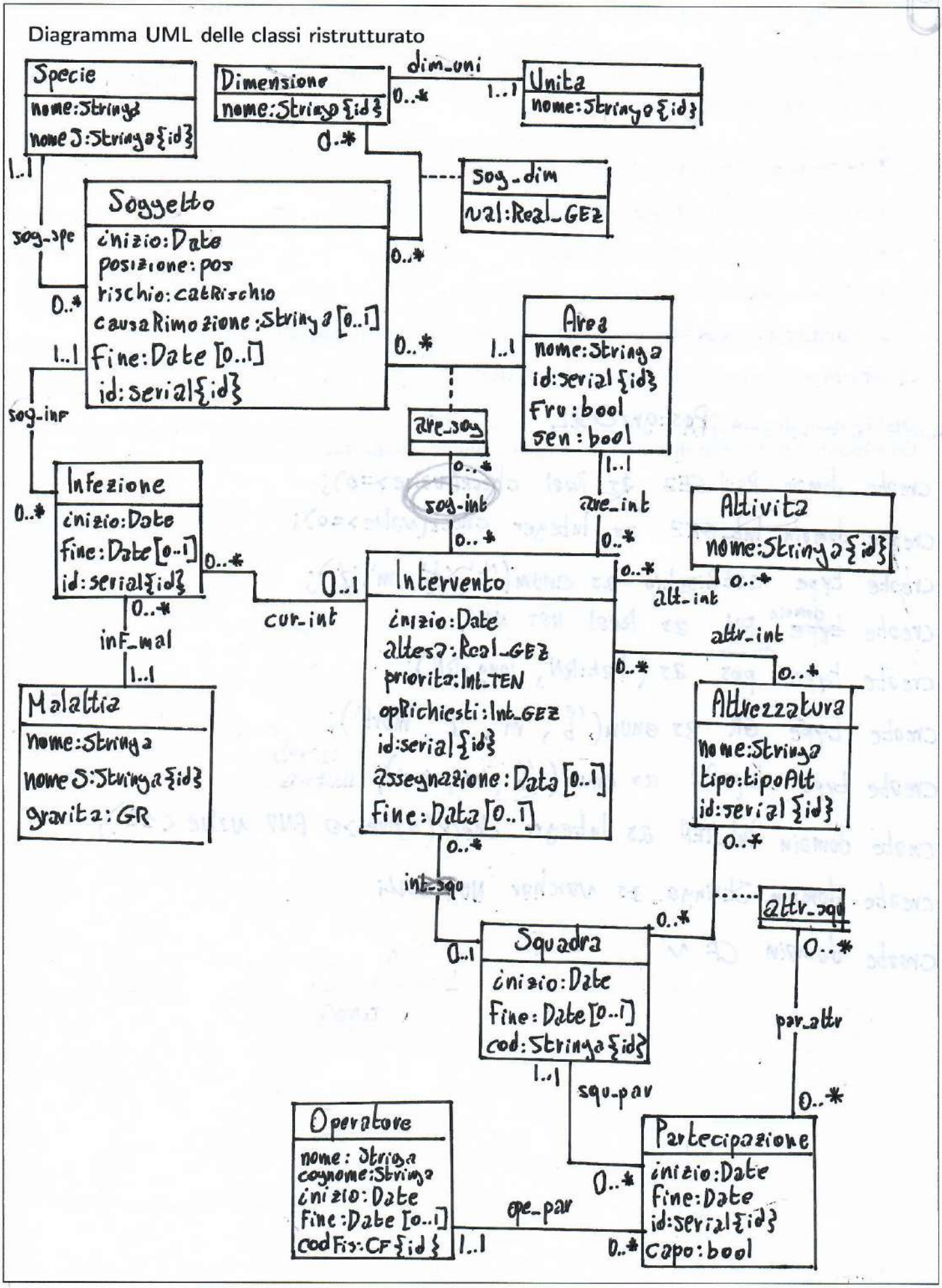
Descrivere brevemente le principali scelte effettuate.

DBMS da utilizzare .PostgreSQL.....

Corrispondenza tra tipi di dato concettuali e domini supportati dal DBMS

```
create domain Real_GEZ as Real check(value>=0);
create domain Int_GEZ as Integer check(value>=0);
create type catRischio as enum('t','b','m','a');
create ~~type~~ domain RN as Real NOT NULL;
create type pos as (lat:RN, long:RN);
create type GR as enum('b','m','a','mort');
create type tipoAlt as enum('l','v','vs');
create domain Int_TEN as Integer check(value>0 AND value<11);
create domain Stringa as varchar NOT NULL;
create domain CF v' Regex'
```

Diagramma UML delle classi ristrutturato



Breve descrizione delle scelte effettuate durante la ristrutturazione

Fusione su: Intervento, Area, Soggetto, Infezione

Il capo è definito nella classe Partecipazione

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.sen_se_fru]
$\forall A\ [Area(A) \wedge sen(A, true)] \rightarrow fru(A, true)$ ⊤

[V.finito_se_ass] ⊤
$\forall i\ [Intervento(i) \wedge \exists f\ fine(i,f)] \rightarrow \exists s\ assegnazione(i,s)$

[V.squadra_se_assegnato] ⊤
$\forall i\ [Intervento(i) \wedge \exists s\ int\_squ(i,s)] \leftrightarrow \exists as\ assegnazione(i,as)$

[V.soggetto_rimosso] ⊤
$\forall s\ [Soggetto(s) \wedge \exists f\ fine(s,f)] \leftrightarrow \exists c\ causaRimozione(s,c)$

[V.curata_da_intervento]
$\forall in\ [Infezione(in) \wedge \exists f\ fine(in,f)] \leftrightarrow \exists t\ Intervento(t) \wedge \exists k\ fine(t,k) \wedge cur\_int(in,t)$

[V.almeno_un_capo]
$\forall s\ Squadra(s) \rightarrow \exists p\ squ\_par(s,p) \wedge capo(p, True)$

[continua alla pagina seguente]

Risposta alla Domanda 6 (segue)

[v. infezione_curata]

$\forall s, inf, int \; [\exists F \; Fine(inf, F) \wedge cur\_int(inf, int) \wedge sog\_inf(inf, s)] \rightarrow \exists a \; ave\_sog(a, s) \wedge sog\_int(a, s, int)$

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 Specie ..... (nome) Derivante da: ~~classe~~ | **associazione** (cerchiare)

Attributi	nome	nomeS						
Domini	Stringa	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 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: .....

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

Attributi	nome	unita						
Domini	Stringa	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 Unita ref Unita(id);

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

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

Attributi	id	nome	fru	sen				
Domini	Serial	Stringa	bool	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):

check(sen=False OR (sen=True AND fru=True));

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

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

Attributi	inizio	posizione	rischio	causaRimozione*	Fine *	id	specie	area
Domini	Date	pos	catRischio	Stringa	Date	serial	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): check(inizio <= fine);
FK specie ref Specie(nomeS); FK area ref Area(id);
check(Fine IS NULL = CausaRimozione IS NULL);

La relazione accorpa le relazioni che implementano le seguenti associazioni: soy_spe, are_soy .....

6 Relazione Attività ..... (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 Squadra ..... (nome) Derivante da: ~~classe~~ | associazione (cerchiare)

Attributi	inizio	Fine *	cod				
Domini	Date	Date	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): check(inizio <= Fine);

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

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

Attributi	inizio	attesa	priorità	opRichiesti	id	assegnazione*	Fine *	squadra *
Domini	Date	Real_GEZ	Int_TEN	Int_GEZ	Serial	Date	Date	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): check(assegnazione >= Inizio); check(Fine >= assegnazione); FK squadra ref Squadra(cod); check(Fine IS NULL OR ((Fine IS NOT NULL AND assegnazione IS NOT NULL)); check(squadra IS NULL = assegnazione IS NULL);

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

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

Attributi	area						
Domini	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 area ref Area(id);

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

10 Relazione Seg_Int ..... (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: .....

11 Relazione Malattia .... (nome) Derivante da: ~~classe~~ | associazione (cerchiare)

Attributi	nome	nomeS	gravità					
Domini	Stringa	Stringa	GR					

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: .....

12 Relazione Infezione .... (nome) Derivante da: ~~classe~~ | associazione (cerchiare)

Attributi	inizio	fine *	id	soggetto	malattia	intervento*		
Domini	Date	Date	Serial	Integer	Stringa	Integer		

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

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): check(inizio <= fine); FK soggetto ref Soggetto(id); Fk malattia ref Malattia(nomeS); Fk intervento ref Intervento(id); check (fine is NULL = intervento is NULL);

La relazione accorpa le relazioni che implementano le seguenti associazioni: sog-inf, cur-int, inf-ma) ...

13 Relazione Attrezzatura .... (nome) Derivante da: ~~classe~~ | associazione (cerchiare)

Attributi	id	nome	tipologia					
Domini	Serial	Stringa	tipoAtt					

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: .....

14 Relazione ..alt_int .... (nome) Derivante da: classe | ~~associazione~~ (cerchiare)

Attributi	attività	intervento						
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 attività ref Attività(nome); fk intervento ref Intervento(id)

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

15 Relazione .altr_squ .... (nome) Derivante da: classe | associazione (cerchiare)

Attributi	squadra	attrezzo						
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 squadra ref Squadra(cod); Fk attrezzo ref Attrezzo(id);

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

16 Relazione attr_int (nome) Derivante da: classe | associazione (cerchiare)

Attributi	altezza	intervento					
Domini	Integer	Integer					

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

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

FK altezza ref Altezza(id);

FK intervento ref Intervento(id);

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

17 Relazione Operatore (nome) Derivante da: classe | associazione (cerchiare)

Attributi	nome	cognome	inizio	fine *	CodFis		
Domini	Stringa	Stringa	Date	Date	CF		

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);

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

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

Attributi	inizio	fine	id	capo	squadra	operatore		
Domini	Date	Date	serial	bool	Stringa	CF		

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 squadra ref Squadra(cod);

FK operatore ref Operatore(codFis);

La relazione accorpa le relazioni che implementano le seguenti associazioni: ope_par, squ_par

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

Attributi		partecipazione	squadra	altezzaturn			
Domini		Integer	Stringa	Integer			

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

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

V.inclusione = (squadra, altezzaturn) ≤ attr_squ

FK partecipazione ref Partecipazione(id);

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.partecipo_se_opera
op: Insert o Update Partecipazione

```
OK = EXISTS(SELECT * FROM Operatore o WHERE o.id = new.operatore
    AND o.inizio <= new.inizio AND o.Fine >= new.Fine);
```

IF OK=Permetti op
else: errore e rollback

T.infetto_se_esistente
op: Insert o Update Infezione

```
OK=EXISTS(SELECT * FROM Soggetto s WHERE s.id= new.soggetto
AND s.inizio<=new.inizio AND s.Fine >= new.Fine);
```

IF OK: Permetti
else :errore e rollback

T.no_inf_cont_con_stessa_malattia
op:Insert o Update Infezione

```
Error = EXISTS (SELECT * FROM Infezione i WHERE i.soggetto = new.soggetto
AND i.id <> new.id AND i.malattia = new.malattia AND
(i.inizio, i.fine) OVERLAPS (new.inizio, new.fine));
```

IF (Error): errore e rollback
else: permetti

[continua alla pagina seguente]

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

```
aree_sensibili_senza_interventi(i:Date, F:Date): Insieme(<Integer>)
if(F<i): termina operazione
Q=(SELECT id FROM Area WHERE sen = True)
      EXCEPT
   (SELECT a.id FROM Area a, Intervento i
    WHERE a.id = i.area AND a.sen=True AND
    (i.inizio, i.Fine) OVERLAPS (i,F));
Result = Q
```

```
tasso_malattie(i:Date, F:Date, M:Insieme(<Stringa>)): Insieme(<Stringa, Real-GE2>)
iF (F<i): termina operazione
Q= WITH  SM as( SELECT count(DISTINCT s.id) as n
                FROM Soggetto s, Infezione in WHERE in.soggetto = s.id
                AND i<=in.inizio AND F>=in.inizio)

   SELECT in.malattia, count(DISTINCT s.id)/SM.n as tasso
   FROM Soggetto s JOIN Infezione in ON s.id=in.soggetto, M, SM
   WHERE i<=in.inizio AND F>=in.inizio AND in.malattia=M.nomeS
   GROUP BY in.malattia;
Result = Q
```

[continua alla pagina seguente]

Risposta alla Domanda 8 (segue)

```
cerca_squadre_qualificate (in:Integer): Insieme(<Integer))
Error = EXISTS( SELECT * FROM Intervento WHERE assegnazione IS NOT NULL
                                          AND in=id);

if (Error=True): termina operazione

Q = WITH AT as ( SELECT a.id FROM Attrezzatura a, attr_int ai
WHERE ai.intervento = in AND ai.attrezzatura = a.id )

NUMAT AS (SELECT count(*) as n FROM AT)

SQNA AS (SELECT s.id, count(DISTINCT pa.attrezzatura) as na,
        count(p.id) as numP
    FROM Partecipa p, Squadra S, par_attr pa, Intervento I, AT
    WHERE p.id=pa.partecipazione AND I.id=in AND I.inizio >= p.inizio
    AND p.squadra = S.cod AND AT.a.id = pa.attrezzatura
    GROUP BY S.id)

SELECT s.id FROM Squadra S, Intervento I, SQNA, NUMAT
WHERE I.id=in AND SQNA.numP >= I.opRichiesti AND
NUMAT.n = SQNA.na AND S.id = SQNA.s.id;

Result = Q
```