

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:

Cultura 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 rindio
- 1.5 dimensioni (da definire)
- 1.6 Rimosso?

- 1.6.1 data rimozione
- 1.6.2 causa

V. rimozione \geq piantumazione

V. operatore dato servizio e squadra in cui opera

3. Operatore

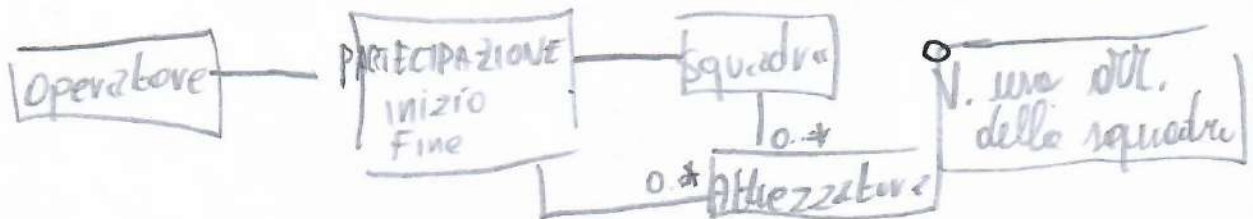
- 3.1 nome
- 3.2 cognome
- 3.3 data inizio servizio
- 3.4 data fine // V. fine $>$ inizio [0..1]
- 3.5 CF

2. Area Verde

- 2.1 denominazione
- 2.2 soggetti
- 2.3 fruibile/non fruibile
 - 2.3.1 Se Fruibile { sensibile, non sensibile }

4. Squadra

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



5. Attrezzatura

- 5.1 tipologia
- 5.2 nome

- 6.7 Squadra assegnata
- 6.8 istante assegnatura
- 6.9 ist. completamento [0..1]

6. Intervento

- 6.1 area
- 6.2 inizio
- 6.3 durata attesa
- 6.4 priorità
- 6.5 attività
- 6.6 piante interessate [0..*]
- 6.7 attrezzature necessarie
- 6.8 num operatori richiesti

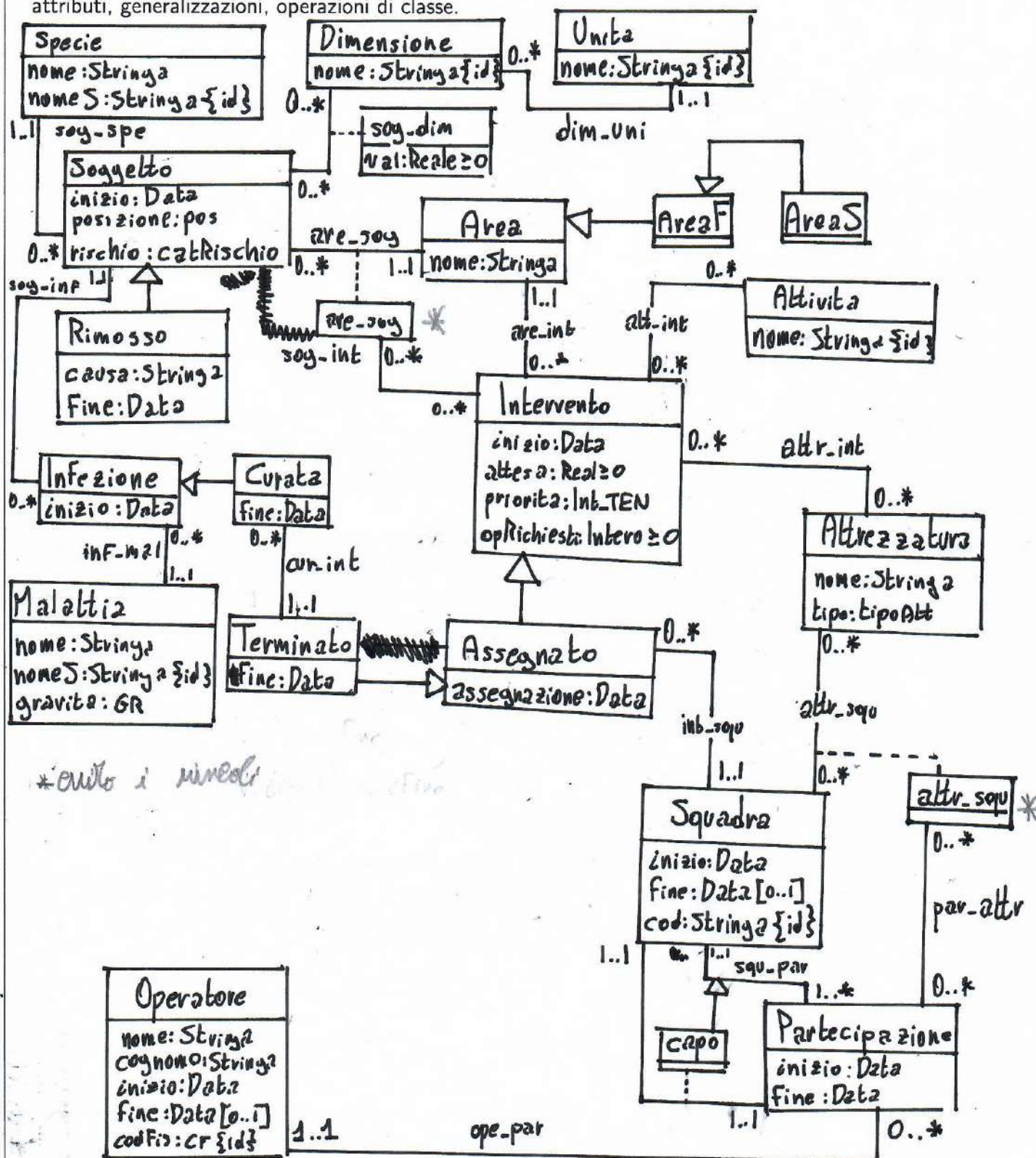
V. piante dell'intervento fanno parte dell'area verde

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

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



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 classe 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 $op(c, arg_1, \dots, arg_n, v)$: uno dei valori di ritorno di op , quando invocata sull'istanza c e con argomenti arg_1, \dots, 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

<p>1 Tipo: <u>Classe</u> Associazione (cerchiare)</p> <p>Nome: <u>Partecipazione</u></p> <p>Operazioni, vincoli:</p> <p>$[V.parparticipa_se_opera]$</p> <p>$\forall p, ip, fp, \sigma, io [Partecipazione(p) \wedge inizio(p, ip) \wedge fine(p, fp) \wedge ope_par(\sigma, p) \wedge inizio(\sigma, io) \rightarrow io \leq ip \wedge [\forall fo \text{ fine}(\sigma, fo) \rightarrow fo \geq fp]]$</p> <p>$[V.parparticipa_se_squadra]$</p> <p>simile a $[V.parparticipa_se_opera]$ ma con la partecipazione compresa nell'esistenza della squadra.</p>	<p>2 Tipo: <u>Classe</u> Associazione (cerchiare)</p> <p>Nome: <u>Infezione</u></p> <p>Operazioni, vincoli: $[V.infezione_curata]$ \textcircled{R}</p> <p>$\forall s, inf, int$</p> <p>$[Curata(inf) \wedge cur_int(inf, int) \wedge sog_inf(inf, s) \rightarrow \exists a \text{ ave_sog}(a, s) \wedge sog_int(a, s, int)]$</p> <p>$[V.infezione_se_esistente]$</p> <p>$\forall s, inf, is, ii [Soggetto(s) \wedge sog_inf(inf, s) \wedge inizio(inf, ii) \wedge inizio(s, is) \rightarrow ii \geq is \wedge [\forall fs, fi \text{ fine}(inf, fi) \wedge fine(s, fs) \rightarrow fs \geq fi]]$</p>
---	--

3 Tipo: Classe Associazione (cerchiare)Nome: Intervento

Operazioni, vincoli:

[V. tempo_assegnazione]

$$\forall in, i, a [Intervento(in) \wedge inizio(in, i) \wedge assegnazione(in, a)] \rightarrow a \geq i$$

$$[\forall F Fine(in, F) \rightarrow F \geq a]$$

[V. soggetti_esistenti_intervento]

$$\forall in, s, ii, is, a [Intervento(in) \wedge Soggetto(s) \wedge are_sq(a, s) \wedge sq_int(a, s, in) \wedge inizio(s, is) \wedge inizio(in, ii)] \rightarrow is \leq ii$$
6 Tipo: Classe Associazione (cerchiare)Nome: Intervento

Operazioni, vincoli:

[V. prende_squadra_esistente]

$$\forall in, s, as, is [Intervento(in) \wedge Squadra(s) \wedge inizio(s, is) \wedge assegnazione(in, as) \wedge int_sq(in, s)] \rightarrow as \geq is \wedge [\forall Fs, Fi Fine(in, Fi) \wedge Fine(s, Fs) \rightarrow Fs \geq Fi]$$
4 Tipo: Classe Associazione (cerchiare)Nome: Squadra

Operazioni, vincoli:

[V. no_part_interseccate_stesso_op]

$$\forall p_1, p_2, s, o, i_1, i_2, f_1, f_2$$

$$[Partecipazione(p_1) \wedge Partecipazione(p_2) \wedge ope_par(o, p_1) \wedge ope_par(o, p_2) \wedge p_1 \neq p_2 \wedge sq_par(s, p_1) \wedge sq_par(s, p_2) \wedge inizio(p_1, i_1) \wedge inizio(p_2, i_2) \wedge fine(p_1, f_1) \wedge fine(p_2, f_2)] \rightarrow f_1 < i_2 \vee f_2 < i_1$$
7 Tipo: Classe Associazione (cerchiare)Nome: Squadra

Operazioni, vincoli:

[V. squadra_qualificata]

$$\forall in, s, at, ii [Assegnato(in) \wedge int_sq(in, s) \wedge assegnazione(in, ii) \wedge altr_int(at, in)] \rightarrow [\exists p, ip, fp Partecipazione(p) \wedge inizio(p, ip) \wedge fine(p, fp) \wedge sq_par(s, p) \wedge ii \geq ip \wedge [\forall F Fine(in, F) \rightarrow F \leq fp] \wedge altr_sq(at, s) \wedge par_altr(p, at, s)]$$
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]

$$\forall inf_1, inf_2, s, m, i_1, i_2 [Infezione(inf_1) \wedge Infezione(inf_2) \wedge inf_mal(inf_1, m) \wedge inf_mal(inf_2, m) \wedge inizio(inf_1, i_1) \wedge inizio(inf_2, i_2) \wedge sq_inf(inf_1, s) \wedge sq_inf(inf_2, s)] \rightarrow [\forall F Fine(inf_2, F) \rightarrow F < i_2] \wedge [\forall F Fine(inf_2, F) \rightarrow F < i_1]$$

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

[V.continuità]

$\forall K, i, f [[\text{Soggetto}(K) \vee \text{Infezione}(K) \vee \text{Intervento}(K) \vee \text{Squadra}(K) \vee \text{Partecipazione}(K) \vee \text{Operatore}(K)] \wedge \text{inizio}(K, i) \wedge \text{Fine}(K, f)] \rightarrow i \leq f$

Tipi di Dato

pos = (lat: Reale, long: Reale)

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

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

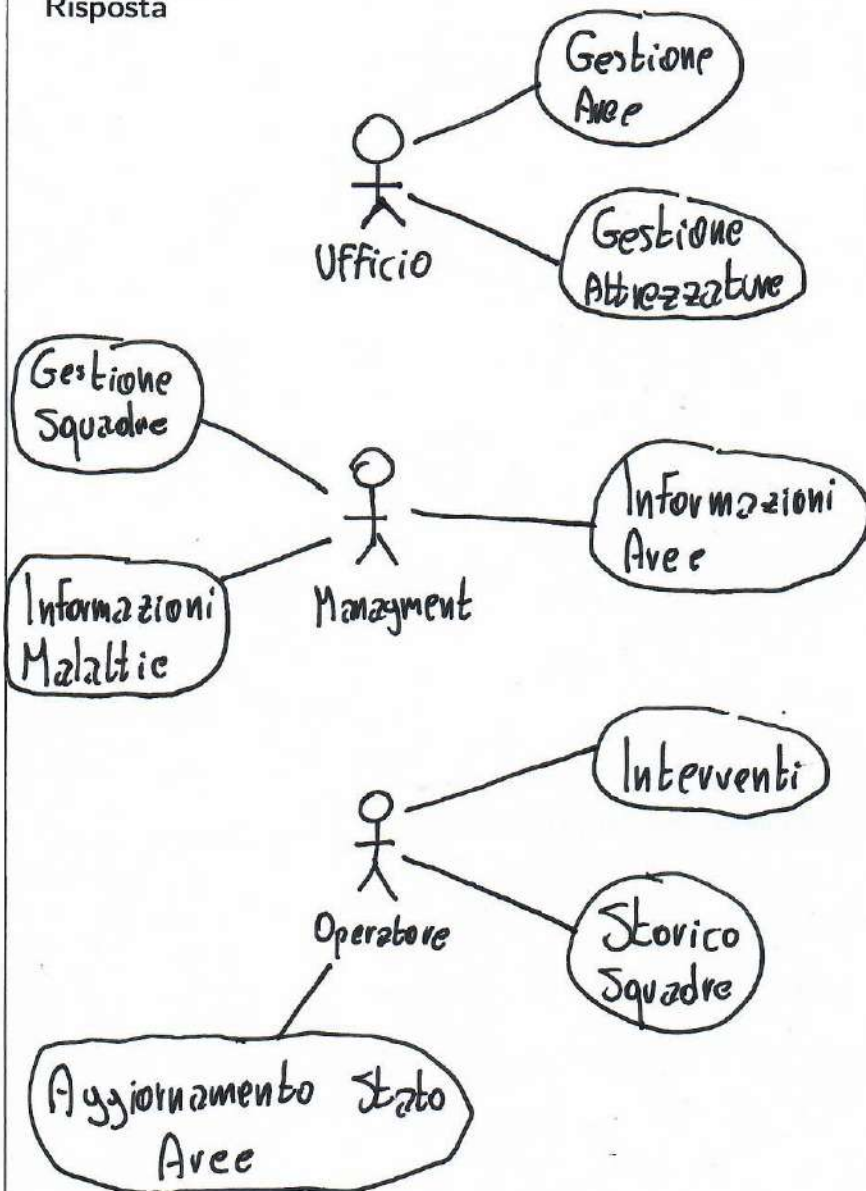
Int_TEN = 1..10

LipoAtt = { 'l', 'v', 'vs' }

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

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:
$$A = \left\{ a \mid AreaS(a) \wedge [\forall in, ii \text{ are_int}(a, in) \wedge inizio(in, ii) \rightarrow F < ii \wedge [\forall fi \text{ fine}(in, fi) \rightarrow fi < i]] \right\}$$

$Cerca_squadre_qualificate(in:Intervento): Squadra[0..*]$

•pre-cond: $\neg Assegnato(in)$

•post-cond:

$$S = \left\{ s \mid \begin{array}{l} \exists is, ii \text{ inizio}(s, is) \wedge inizio(in, ii) \wedge is \leq ii \wedge \text{Squadra}(s) \wedge [\forall at \text{ altr_int}(at, in) \rightarrow \exists p, ip \text{ Partecipazione}(p) \\ \wedge \text{squ_par}(s, p) \wedge inizio(p, ip) \wedge ip \leq ii \wedge \text{altr_squ}(at, s) \wedge \\ \text{par_altr}(at, s, p)] \wedge \exists n, m \text{ opRichiesti}(in, m) \wedge \\ n = \left\{ p_2 \mid \text{Partecipazione}(p_2) \wedge \text{squ_par}(s, p_2) \wedge \exists ip_2 \right\} \wedge m \leq n \end{array} \right\}$$

Result = S

$Lasso_malattie(i:Data, f:Data, M:Malattia[1..*]): (Malattie, Realezo)[0..*]$

•pre-cond: $i \leq f$

•post-cond:

$$T = \left\{ \left(m, \frac{k}{n} \right) \mid \begin{array}{l} m \in M \wedge n = \left\{ s \mid \exists inf, if \text{ Infezione}(inf) \wedge inizio(inf, if) \wedge i \leq if \leq f \wedge \text{Soggetto}(s) \wedge \text{sog_inf}(inf, s) \right\} \\ k = \left\{ s \mid \exists inf, if \text{ Infezione}(inf) \wedge inizio(inf, if) \wedge i \leq if \leq f \wedge \text{Soggetto}(s) \wedge \text{sog_inf}(inf, s) \wedge \text{inf_mal}(inf, m) \right\} \end{array} \right\}$$

Result = T

FINITO IN 2 ORE E 40 MINUTI

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.

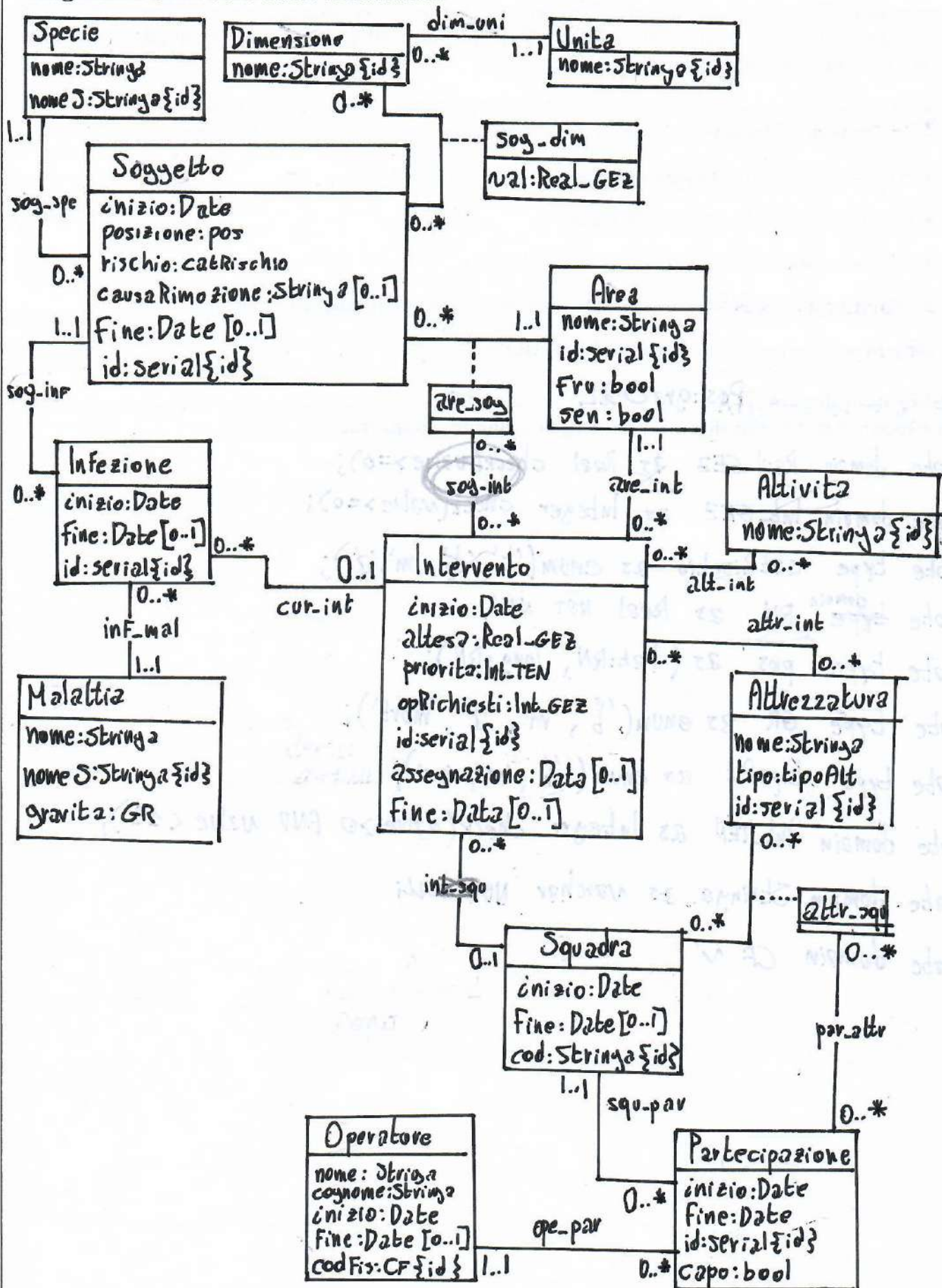
Descrivere brevemente le principali scelte effettuate.

DBMS da utilizzare PostgreSQL

Corrispondenza tra tipi di dato concettuali e domini supportati dal DBMS

```
create domain Real_GE2 as Real check(value >= 0);
create domain Int_GE2 as Integer check(value >= 0);
create type catRischio as enum('l', 'b', 'm', 'a');
create typedomain 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('i', 'v', 'us');
create domain Int_TEN as Integer check(value > 0 AND value < 11);
create domain Stringa as varchar NOT NULL;
create domain CF as
```

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 omettono i vincoli esterni la cui formulazione è rimasta identica a seguito della ristrutturazione)

[V.sen-sc-fru]

$\forall A [Area(A) \wedge sen(A, true)] \rightarrow fru(A, true)$ T

[V.finito-se-ass]

$\forall i [Intervento(i) \wedge \exists f fine(i, f)] \rightarrow \exists s assegnazione(i, s)$ T

[V.squadra-se-assegnato]

$\forall i [Intervento(i) \wedge \exists s int_squ(i, s)] \leftrightarrow \exists as assegnazione(i, as)$ T

[V.soggetto_rimosso]

$\forall s [Soggetto(s) \wedge \exists f fine(s, f)] \leftrightarrow \exists c causaRimozione(s, c)$ T

[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]

$\forall s Squadra(s) \rightarrow \exists p squ_par(s, p) \wedge capo(p, true)$

Risposta alla Domanda 6 (segue)

[v.infezione_curata]

$\forall s, inf, int \ [\exists F \text{ Fine}(inf, F) \wedge \text{con-int}(inf, int) \wedge \text{soy-inf}(inf, s)]$

~~$\forall s, inf, int \rightarrow \exists a \text{ con-soy}(a, s) \wedge \text{soy-int}(a, s, int)$~~

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	<u>nome</u>	<u>nomeS</u>						
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 Unità (nome) Derivante da: ~~classe~~ | associazione (cerchiare)

Attributi	<u>nome</u>							
Domini	Stringa							

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

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

La relazione accorpa le relazioni che implementano le seguenti associazioni:

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

Attributi	<u>nome</u>	<u>unità</u>						
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 unità ref Unità(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: dim-uni

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

Attributi	<u>id</u>	<u>nome</u>	<u>fru</u>	<u>sen</u>				
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	<u>inizio</u>	<u>posizione</u>	<u>rischio</u>	<u>causaRimozione*</u>	<u>Fine*</u>	<u>id</u>	<u>specie</u>	<u>area</u>
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->pe, ave-soy

6	Relazione <u>Attività</u> (nome)	Derivante da: <u>classe</u> associazione (cerchiare)
Attributi	<u>nome</u>	
Domini	<u>stringa</u>	

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

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

La relazione accorpa le relazioni che implementano le seguenti associazioni:

7	Relazione <u>Squadra</u> (nome)	Derivante da: <u>classe</u> associazione (cerchiare)
Attributi	<u>inizio</u> <u>fine</u> * <u>cod</u>	
Domini	<u>Date</u> <u>Date</u> <u>stringa</u>	

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

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): check(inizio <= fine);

La relazione accorpa le relazioni che implementano le seguenti associazioni:

8	Relazione <u>Intervento</u> (nome)	Derivante da: <u>classe</u> associazione (cerchiare)
Attributi	<u>inizio</u> <u>altezza</u> <u>priorita</u> <u>opRichiesti</u> <u>id</u> <u>assegnazione</u> * <u>fine</u> * <u>squadra</u> *	
Domini	<u>Date</u> <u>Real-GE2</u> <u>Int-TEN</u> <u>Int-GE2</u> <u>Serial</u> <u>Date</u> <u>Date</u> <u>Stringa</u>	

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

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): check(assegnazione >= fine);
check(fine >= assegnazione); FK squadra ref Squadra(cod); check(fine is NULL OR
check(squadra is NULL = assegnazione is NULL); ((fine is NOT NULL AND
int_squ assig is NOT NULL));

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

9	Relazione <u>Intervento</u> (nome)	Derivante da: <u>classe</u> associazione (cerchiare)
Attributi	<u>area</u>	
Domini	<u>Integer</u>	

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

Vincoli (foreign key, inclusione, altra chiave, di ennupla, di dominio): fk area ref Area(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni: 21c_int

10	Relazione <u>seg_int</u> (nome)	Derivante da: <u>classe</u> <u>associazione</u> (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	<u>nome</u>	<u>nomeS</u>	<u>gravita</u>					
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 enupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni:

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

Attributi	<u>inizio</u>	<u>fine *</u>	<u>id</u>	<u>soggetto</u>	<u>malattia</u>	<u>intervento*</u>		
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 enupla, 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: ag-inf, cu-inf, inf-wz)...

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

Attributi	<u>id</u>	<u>nome</u>	<u>tipologia</u>					
Domini	Serial	Stringa	tipoAlt					

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

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

La relazione accorpa le relazioni che implementano le seguenti associazioni:

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

Attributi	<u>attivita</u>	<u>intervento</u>						
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 enupla, di dominio):

FK attivita ref Attivita(nome); FK intervento ref Intervento(id)

La relazione accorpa le relazioni che implementano le seguenti associazioni:

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

Attributi	<u>squadra</u>	<u>altvezzo</u>						
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 enupla, di dominio):

FK squadra ref Squadra(cod); FK altvezzo ref Altvezzo(id);

La relazione accorpa le relazioni che implementano le seguenti associazioni:

16 Relazione altv.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 enupla, di dominio):

FK altezza ref Altvezzo(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 * | CodFrs | | | |

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 enupla, 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 enupla, di dominio): check(inizio <= fine);

FK squadra ref Squadra(cod);

FK operatore ref Operatore(codFrs);

La relazione accorpa le relazioni che implementano le seguenti associazioni: opc-par, squ-par

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

Attributi | opu | partecipazione | squadra | altvezzo | | | |

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 enupla, di dominio):

V.inclusione = (squadra, altvezzo) <= altv-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 enupla, di dominio):

La relazione accorpa le relazioni che implementano le seguenti associazioni:

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.infelto_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

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_intervenire(*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-GE>*)

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.nome*
GROUP BY *in.malattia*;

Result = *Q*

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(pa.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