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.ER – Modulo risposte prova scritta**

Dati dello studente e dell'esame

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

Data: .....

Corso di laurea e canale di appartenenza:

- ☐ Laurea in Informatica, canale 1 (A-L, Prof. G. Perelli)
- ☐ Laurea in Informatica, canale 2 (M-Z, Prof.ssa M. De Marsico)
- ☐ Laurea in Informatica in Modalità Teledidattica Unitelma Sapienza

Firma di un membro della Commissione per avvenuta identificazione:

.....

Rinuncia alla prova

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

[Versione 2024-03-29]

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 2024-03-29]

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 ER concettuale (*)
  - il relativo dizionario dei dati
  - le specifiche dei domini concettuali non di tipo base
  - eventuali vincoli esterni, espressi utilizzando il linguaggio della logica del primo ordine (*)
- 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)
  - la specifica delle operazioni di use-case necessarie a modellare i requisiti contrassegnati dalla barra laterale (come quella qui a sinistra) in termini di precondizioni e postcondizioni, utilizzando il linguaggio della logica del primo ordine (*) (30 minuti; 60 minuti al massimo)

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 ER concettuale e dei vincoli esterni (20 minuti; 30 minuti al massimo):
    - scelta del DBMS da utilizzare
    - progettazione della corrispondenza tra i domini concettuali ed opportuni domini SQL (domini base o utente, oppure realizzati mediante relazioni aggiuntive) supportati dal DBMS scelto
    - ristrutturazione del diagramma ER concettuale e 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 di use-case 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. (*)

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

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 33 in poi possono essere utilizzate per scrivere minute che non verranno valutate.

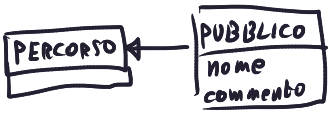
Questa pagina è stata intenzionalmente lasciata vuota

1 Analisi concettuale

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

Risposta

1. Utente
   - 1.1 username
   - 1.2 nome
   - 1.3 cognome
   - 1.4 genere
   - 1.5 nascita
   - 1.6 altezza } FACOLTATIVI
   - 1.7 peso }
   - 1.8 Allenamento
     - 1.8.1 data
     - 1.8.2 ora inizio
     - 1.8.3 ora fine
     - 1.8.4 difficolta' } opzionale
     - 1.8.5 Percorso (FACOLTATIVO)
       - 1.8.5.1 sequenza di (coordinata, istante) ~ (x, y, t)
       - 1.8.5.2 pubblico?
         - 1.8.5.2.1 nome
         - 1.8.5.2.2 commento
     - 1.8.6 sport
   - 1.9 Amici



2. Sport
   - 2.1 nome
   - 2.2 descrizione

3. SFida
   - 3.1 nome
   - 3.2 tipo (squadre, singolo)
     - 3.2.1 se squad => parb. max per squadra
       - nome squadre
       - componenti
   - 3.3 inizio
   - 3.4 fine
   - 3.5 durata minima allenamento
   - 3.6 sport [...+]
   - 3.7 cat ("piu' km" o "piu' ore")
   - 3.8 Utenti iscritti

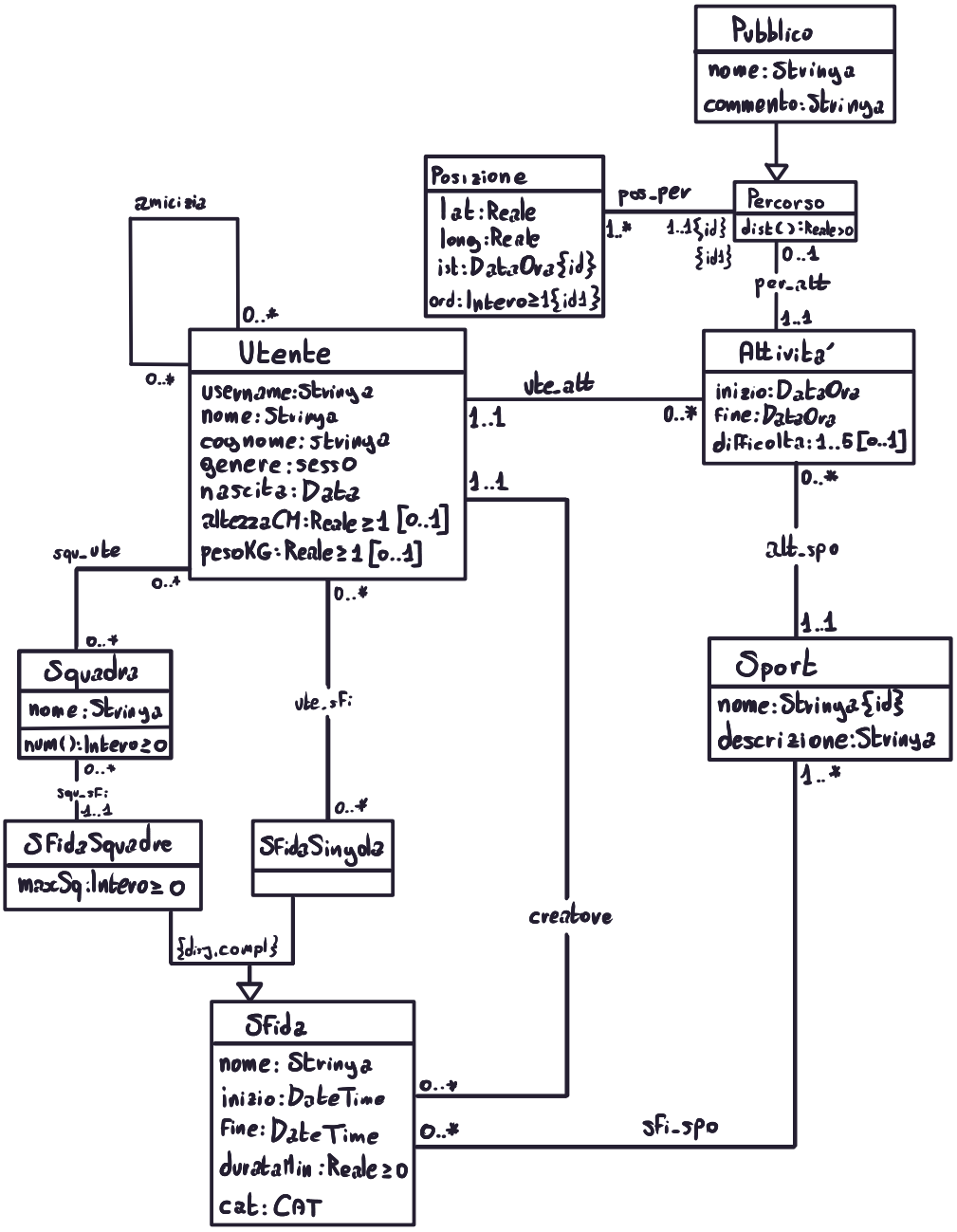
[continua alla pagina seguente]

Domanda 2 (45 minuti; 75 minuti al massimo) Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma ER concettuale per l'applicazione, il dizionario dei dati ed eventuali vincoli esterni.

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

Diagramma ER

Produrre un diagramma ER concettuale per l'applicazione in termini di entità, relationship, attributi, relazioni is-a, generalizzazioni (disgiunte) complete e non.



[continua alla pagina seguente]

Dizionario dei dati Per ogni entità e relationship del diagramma ER **con** **attributi o vincoli**:

- Definire il dominio e la molteplicità degli attributi (se diversa da (1,1))
- Definire eventuali vincoli esterni in logica del primo ordine estesa con teoria degli insiemi e semantica di mondo reale, usando il seguente alfabeto:
  - Un simbolo di predicato $E/1$ per ogni entità E.
    Semantica di $E(x)$: x è una istanza di E.
  - Un simbolo di predicato $D/1$ per ogni dominio D.
    Semantica di $D(x)$: x è un valore di D.
  - Un simbolo di predicato r/n ($n > 0$) per ogni relationship n-aria r.
    Semantica di $r(x_1, \ldots, x_n)$: $x_1, \ldots, x_n$ è una istanza di r.
  - Un simbolo di predicato $a/2$ per ogni attributo a di entità
    Semantica di $a(x, v)$: uno dei valori dell'attributo a dell'istanza x è v.
  - Un simbolo di predicato $a/(n+1)$ per ogni attributo a di relationship n-aria.
    Semantica di $a(x_1, \ldots, x_n, v)$: uno dei valori dell'attr. a dell'istanza $(x_1, \ldots, x_n)$ della relat. è v.
  - Opportuni simboli di predicato (soggetti a *semantica di mondo reale*) per gestire confronti tra valori di domini numerici o comunque ordinati (tra cui $</2$, $\leq/2$, $>/2$, $\geq/2$).
  - Il predicato di uguaglianza $=/2$ (la cui interpretazione è la relazione che lega ogni elemento del dominio di interpretazione solo con se stesso).
  - Opportuni simboli di costante (soggetti a *semantica di mondo reale*), tra cui $adesso$, interpretato come il valore del dominio DataOra che rappresenta l'istante corrente.

Risposta

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

Nome: Utente

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.nasce_prima_di_attività]

$\forall u, n, a, ia, di$
$[Utente(u) \wedge nascita(u,n) \wedge ute\_att(u,a) \wedge inizio(a,ia)$
$\wedge Data(ia,di)] \rightarrow n < di$

[V.nasce_prima_di_sfida]

$\forall u, s, n, is, ds$
$[Utente(u) \wedge Sfida(s) \wedge [creatore(u,s) \vee ute\_sfi(u,s)$
$\vee [\exists sq \; squ\_ute(sq,u) \wedge squ\_sfi(sq,s)]] \wedge nascita(u,n)$
$\wedge inizio(s,is) \wedge Data(is,ds)] \rightarrow ds > n$

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

Nome: Utente

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.non_partecipa_stessa_sfida]

$\forall u, sq1, sq2, s$
$[Utente(u) \wedge Squadra(sq1) \wedge Squadra(sq2)$
$\wedge Sfida Squadre(s) \wedge squ\_ute(u, sq1) \wedge squ\_ute(u, sq2)$
$\wedge squ\_sfi(sq1, s) \wedge squ\_sfi(sq2, s)] \rightarrow sq1 = sq2$

[V.no_auto_amicizia]

$\forall u1, u2 \; [amicizia(u1, u2)] \rightarrow u1 \neq u2$

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

Nome: Squadra

attributo	dominio	moltepl. (*)

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

Vincoli:

[V. limite_membri_squadra]

$\forall sq, s, nm, nMax$

$[Squadra(sq) \wedge squ\_sFi(sq,s) \wedge maxSq(S, nMax) \wedge num(sq,nm)] \rightarrow nm \leq nMax$

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

Nome: sFida

attributo	dominio	moltepl. (*)

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

Vincoli:

[V. inizio_sFida_minore_Fine]

$\forall S, i, F\ [SFida(s) \wedge inizio(s,i) \wedge Fine(s,F)] \rightarrow i < F$

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

Nome: Posizione

attributo	dominio	moltepl. (*)

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

Vincoli:

[V. successione_ordini]

$\forall p, \sigma, per$

$[Posizione(p) \wedge ord(p,\sigma) \wedge pos\_per(p,per) \wedge \sigma \neq 1] \rightarrow [\exists p2, \sigma 2\ \ Posizione(p2) \wedge ord(p2,\sigma 2) \wedge pos\_per(p2,per) \wedge \sigma 2 = \sigma - 1]$

[V. istante_pos_fra_inizio_e_Fine]

$\forall pos, per, at, ip, iF, i$

$[Posizione(pos) \wedge pos\_per(pos,per) \wedge per\_alt(per,at) \wedge ist(pos,ip) \wedge inizio(at,i) \wedge Fine(at,iF)] \rightarrow i \leq ip \leq if$

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

Nome: Posizione

attributo	dominio	moltepl. (*)

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

Vincoli:

[V. coerenza_su_istanti_e_ordini]

$\forall p1, p2, i1, i2, \sigma 1, \sigma 2, pr\ [Posizione(p1) \wedge Posizione(p2) \wedge ord(p1,\sigma 1) \wedge ord(p2,\sigma 2) \wedge ist(p1,i1) \wedge ist(p2,i2) \wedge \sigma 1 = \sigma 2 - 1 \wedge pos\_per(p1,pr) \wedge pos\_per(p2,pr)] \rightarrow [\neg \exists p3, i3\ \ pos\_per(p3,pr) \wedge ist(p3,i3) \wedge i1 \leq i3 \leq i2]$

[V. coerenza_su_istanti_e_ordini2]

$\forall p1, p2, i1, i2, \sigma 1, \sigma 2, pr\ [Posizione(p1) \wedge Posizione(p2) \wedge ord(p1,\sigma 1) \wedge ord(p2,\sigma 2) \wedge ist(p1,i1) \wedge ist(p2,i2) \wedge pos\_per(p1,pr) \wedge pos\_per(p2,pr) \wedge \sigma 1 < \sigma 2] \rightarrow i1 < i2$

[Versione 2024-03-29]

7 Tipo: ~~Entità~~ | **Relationship** (cerchiare)

Nome: Attività'

attributo	dominio	moltepl. (*)

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

Vincoli:

[V.inizio_prima_di_Fine]

$\forall z, i, f\ [\text{Attività}(z) \wedge \text{inizio}(z,i) \wedge \text{Fine}(z,F)] \rightarrow i < f$

[V.disgiunzione_att_di_uno_stesso_utente]

$\forall z1, z2, u, i1, i2, F1, F2$
$[\text{Attività}(z1) \wedge \text{Attività}(z2) \wedge \text{Utente}(u) \wedge \text{ute\_att}(u, z1)$
$\wedge \text{ute\_att}(u, z2) \wedge \text{inizio}(z1, i1) \wedge \text{inizio}(z2, i2) \wedge$
$\text{Fine}(z1, F1) \wedge \text{Fine}(z2, F1)] \rightarrow$
$[F1 < i2] \vee [F_2 < i1]$

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: .....

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: .....

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: .....

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: .....

attributo	dominio	moltepl. (*)

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

Vincoli:

[Versione 2024-03-29]

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Nome: . . . . . .

attributo	dominio	moltepl. (*)

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

Vincoli:

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

Ulteriori vincoli esterni, specifica di eventuali operazioni ausiliarie invocate da tali vincoli, e specifica dei domini concettuali non di tipo base

TIPI

$CAT = \{"KM", "all"\}$

$Sesso = \{"M", "F"\}$

Squadra

num( ): Intero≥0
- pre-cond: nessuna
- post-cond: $Result = |\{u \mid squ\_ute(this, u)\}|$

Percorso

dist( ): Reale >0
- pre-cond: nessuna
- post-cond:

$$Coppie = \left\{ (p1, p2, x1, x2, y1, y2) \;\middle|\; \begin{array}{l} Posizione(p1) \wedge Posizione(p2) \wedge lat(p1, x1) \wedge lat(p2, x2) \\ \wedge\, long(p1, y1) \wedge long(p2, y2) \wedge [\exists \sigma1, \sigma2 \;\; ord(p1, \sigma1) \\ \wedge\, ord(p2, \sigma2) \wedge \sigma1 = \sigma2 - 1\;] \wedge pos\_per(this, p1) \wedge pos\_per(this, p2) \end{array} \right\}$$

$$Dist = \sum_{\substack{(p1, p2, x1, x2, y1, y2) \\ \in \\ Coppie}} \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \cdot 0{,}9996$$

Result = Dist

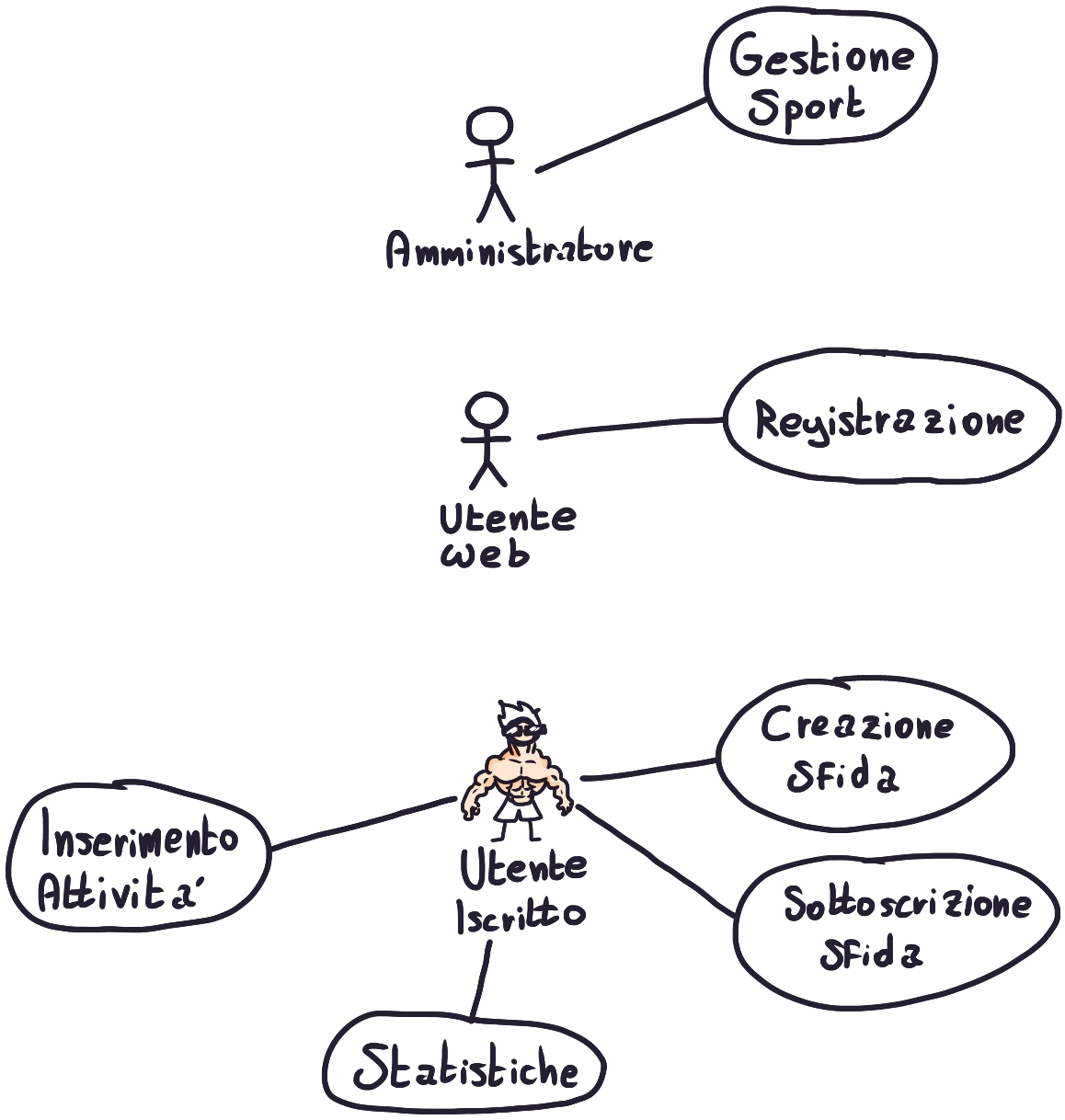
[Versione 2024-03-29]

[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



Domanda 4 (10 minuti) Proseguire la fase di Analisi Concettuale dei requisiti definendo le operazioni degli use-case.

In particolare, per ogni use-case definito nella risposta alla Domanda 3 definire la **segnatura** di tutte le operazioni che lo compongono, in termini di nome dell'operazione, nomi e dominio concettuale degli argomenti, dominio concettuale dell'eventuale valore di ritorno.

1 **Specifica use-case:** Gestione Sport ..... **(nome use-case)**

Operazioni dello use-case:

inserisci_sport(nome:Stringa, desc:Stringa): Sport

2 **Specifica use-case:** Registrazione ..... **(nome use-case)**

Operazioni dello use-case:

sign_in(un:Stringa, nome:Stringa, cognome: Stringa, gen:sesso, nascita, h:Reale ≥ 1 [0..1]
ω:Reale ≥ 1 [0..1]): Utente

3 **Specifica use-case:** Inserimento Attivita' ..... **(nome use-case)**

Operazioni dello use-case:

new_att(i:Data2Ore, f:Data2Ore, d:1..5, p:Percorso[0..1]): Attivita'

[Versione 2024-03-29]

4 Specifica use-case: Creazione SFida ..... (nome use-case)

Operazioni dello use-case:

crea(n:Stringa, i:DateTime, F:DateTime, dm:Reale≥0, c:CAT, maxSq:Intero≥2 [0..1]): SFida

5 Specifica use-case: Sottoscrizione SFida ..... (nome use-case)

Operazioni dello use-case:

crea_squadra(nome:Stringa, S:SFidaSquadre): Squadra

partecipa_sfida(s:SFideSingola)

entra_squadra(S:Squadra)

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

Operazioni dello use-case:

resoconto(u:Utente, i:DataOra, f:DataOra) :(Sport, Reale≥0, Reale≥0)[0..*]

7 Specifica use-case: ..... (nome use-case)

Operazioni dello use-case:

[Versione 2024-03-29]

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

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

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

Risposta

resoconto_sport(u:Utente, i:DataOra, f:DataOra, s:Sport) : (Reale≥0, Reale≥0)

- pre-cond: $\exists\, a$ Attività(a) $\wedge$ vte_att(u,a) $\wedge$ att_spo(a,s) $\wedge$ i < f
- post-cond:

$$A = \{(a, ia, fa) \mid vte\_att(u,a) \wedge att\_spo(a,s) \wedge inizio(a,ia) \wedge fine(a,if) \wedge ia \geq i \wedge fa \leq f\}$$

$$tempo\_tot = \sum_{(a,ia,fa) \in A} (fa - ia)$$

$$B = \{(a, ia, fa, p, dp) \mid vte\_att(u,a) \wedge att\_spo(a,s) \wedge inizio(a,ia) \wedge fine(a,if) \wedge per\_atta(a,p) \wedge dist(p,dp) \wedge ia \geq i \wedge fa \leq f\}$$

$$dist\_tot = \sum_{(a,ia,fa,p,dp) \in B} dp$$

Result = (tempo_tot, dist_tot)

resoconto(u:Utente, i:DataOra, f:DataOra) : (Sport, Reale≥0, Reale≥0)[0..*]

- pre-cond: i < f
- pre-cond:

$$S = \{(s, oa, dp) \mid \exists at\ Attività(u,at) \wedge vte\_att(u,at) \wedge att\_spo(u,s) \wedge resoconto\_sport(u,i,f,s,oa,dp)\}$$

Result = S

[Versione 2024-03-29]

[continua alla pagina seguente]

Risposta alla Domanda 5 (segue)

ore_dedicate_a_sport_amici (u:Utente, s:Sport, i:DataOra, f:DataOra) : Reale ≥ 0

- pre-cond: $i < f$
- post-cond:

$$U = \{(u1, ore) \mid amicizia(u, u1) \wedge \exists K \ resoconto\_sport(u1, i, f, s, ore, K)\}$$

$$Result = \sum_{(u,ore) \in U} ore$$

sport_moda_amici(u:Utente, i:DataOra, f:DataOra) :(Sport, Reale ≥ 0)[0..*]

- pre-cond: $i < f$
- post-cond: $S = \{(s, \sigma) \mid ore\_dedicate\_a\_sport\_amici(u, s, i, f, \sigma)\}$

$$S_m = \underset{(s,\sigma) \in S}{argmax}(\sigma)$$

$$Result = S_m$$

[Versione 2024-03-29]

2 Progettazione della base dati e delle funzionalità

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

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

Descrivere brevemente le principali scelte effettuate.

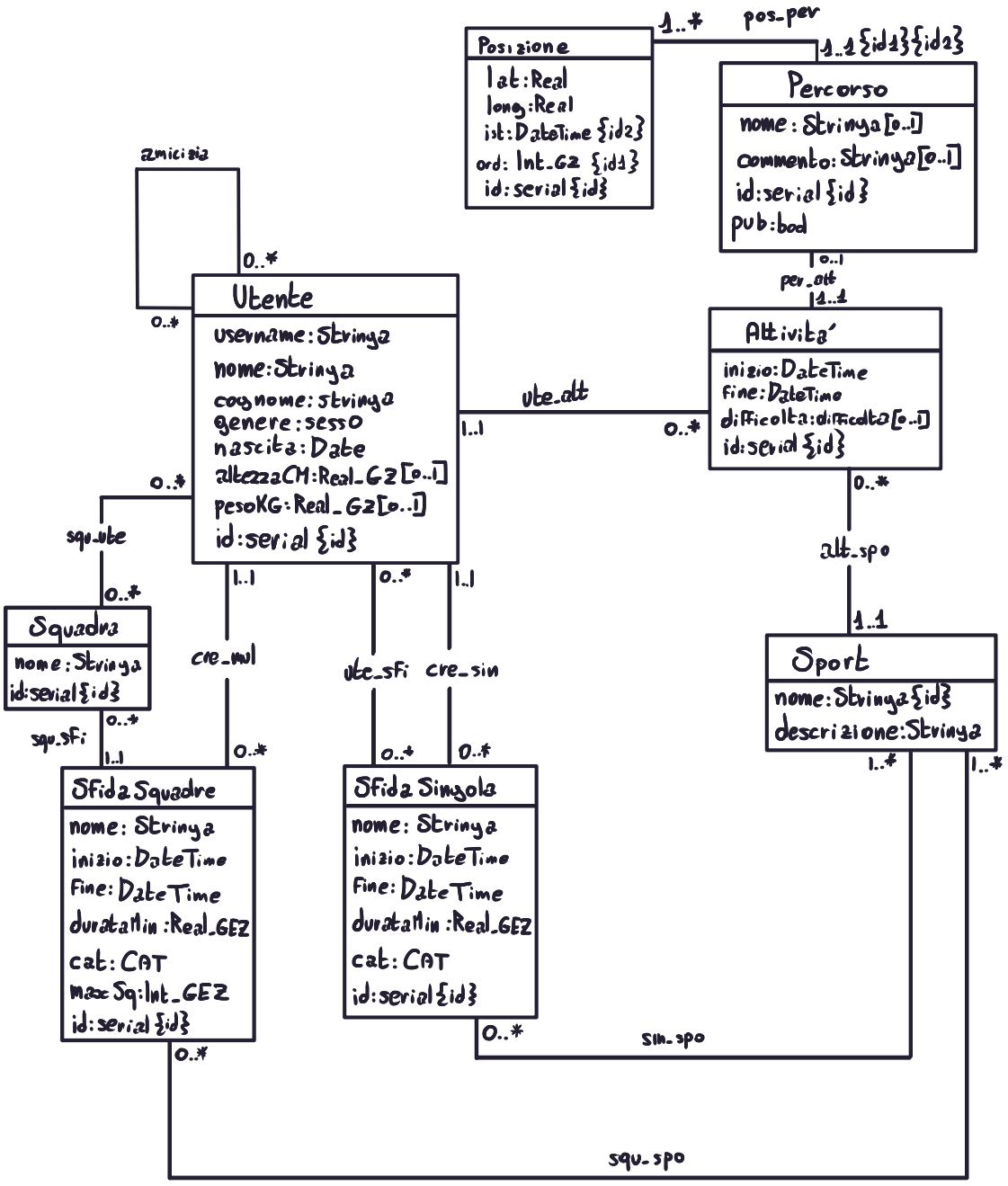
DBMS da utilizzare PostgreSQL

Corrispondenza tra domini concettuali e domini supportati dal DBMS

```
create type CAT as enum("KM","All");

create type sesso as enum("M","F");

create domain Stringa as varchar NOT NULL;

Create domain difficolta as Integer check(value>=1 AND value<=5);

create domain Real_G2 as Real    check(value > 0 );
create domain Int_G2  as Integer check(value > 0 );

create domain Real_GE2 as Real    check(value>= 0 );
create domain Int_GE2  as Integer check(value >= 0 );
```

[Versione 2024-03-29]

Diagramma ER ristrutturato



Breve descrizione delle scelte effettuate durante la ristrutturazione

Fusione su percorso

Divisione su sfida

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.percorso_pubblico]

$\forall p,t\ [Percorso(p) \wedge pub(p,t) \wedge t = \text{'True'}] \leftrightarrow [\exists n,c\ \ nome(p,n) \wedge commento(p,c)]$

$\forall p\ [Percorso(p) \wedge \exists n\ nome(p,n)] \leftrightarrow [Percorso(p) \wedge \exists c\ commento(p,c)]$

[V.inizio_sfida_minore_fine]

$\forall s,i,f\ [[SfidaSingola(s) \vee SfidaSquadre(s)] \wedge inizio(s,i) \wedge Fine(s,f)] \rightarrow i < f$

[V.nasce_prima_di_sfida]

$\forall u,s,n,is,ds$
$[Utente(u) \wedge [SfidaSingola(s) \vee SfidaSquadre(s)] \wedge [cre\_mul(u,s) \vee ute\_sfi(u,s) \vee cre\_sin(u,s) \vee [\exists sq\ \ squ\_ute(u,sq)$
$\wedge squ\_SFi(sq,s)]] \wedge nascita(u,n) \wedge inizio(s,is) \wedge Data(is,ds)] \rightarrow ds > n$

[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 ER ristrutturato.

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

1 **Relazione** Utente ..... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	username	nome	cognome	genere	nascita	altezzaCM *	pesoKG *	id
Domini	Stringa	Stringa	Stringa	sesso	Data	Real_GZ	Real_GZ	serial

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

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

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

2 **Relazione** amicizia ..... (nome) Derivante da: **entità** | ~~**relationship**~~ (cerchiare)

Attributi	u1	u2						
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 u1 ref Utente(id);   check(u1 <> u2);
fk u2 ref Utente(id);
```

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

3 **Relazione** Sport ..... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	nome	descrizione						
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 relationship: .....

4 **Relazione** Attività ..... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	inizio	Fine	difficolta *	id	sport		utente	
Domini	DateTime	DateTime	difficolta	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):

```
fk sport ref Sport(nome);
                    check(inizio < Fine);
fk utente ref Utente(id);
```

La relazione accorpa le relazioni che implementano le seguenti relationship: alt_spo, vte_alt

5 **Relazione** Percorso ..... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	nome *	commento *	id	pub	attività			
Domini	Stringa	Stringa	serial	bool	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):

```
unique(attività);
fk attività ref Attività(id);
```

La relazione accorpa le relazioni che implementano le seguenti relationship: per_alt

[Versione 2024-03-29]

6 **Relazione** Posizione ..... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	lat	long	id	ist	ord	percorso		
Domini	Real	Real	serial	DateTime	Int_G2	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 percorso ref Percorso(id);

unique(percorso,ist);

unique(percorso,ord);

Percorso(id) ⊆ Posizione(percorso);

La relazione accorpa le relazioni che implementano le seguenti relationship: pos-per .....

7 **Relazione** SfidaSquadre ... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	nome	inizio	fine	durataMin	cat	maxSq	id	creatore
Domini	Stringa	DateTime	DateTime	Real_G2	CAT	Int_GE2	serial	Integer

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

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

check(inizio < fine);

Fk creatore ref Utente(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: cre-mul .....

8 **Relazione** SfidaSingola ... (nome) Derivante da: ~~entità~~ | **relationship** (cerchiare)

Attributi	nome	inizio	fine	durataMin	cat		id	creatore
Domini	Stringa	DateTime	DateTime	Real_G2	CAT		serial	Integer

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

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

check(inizio < fine);

Fk creatore ref Utente(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: cre-sin .....

9 **Relazione** SM-SPO ..... (nome) Derivante da: **entità** | ~~**relationship**~~ (cerchiare)

Attributi	sfida	sport						
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 sfida ref SfidaSingola(id);

Fk sport ref Sport(nome);

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

10 **Relazione** SQ-SPO ..... (nome) Derivante da: **entità** | ~~**relationship**~~ (cerchiare)

Attributi	sfida	sport						
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 sfida ref SfidaSquadre(id);

Fk sport ref Sport(nome);

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

[Versione 2024-03-29]

11 **Relazione** Squadra (nome) Derivante da: **entità** | **relationship** (cerchiare)

Attributi	nome	id	sfida					
Domini	Stringa	serial	Integer					

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

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

Fk sfida ref SfidaSquadre(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: squ_sfi

12 **Relazione** squ_ute (nome) Derivante da: **entità** | **relationship** (cerchiare)

Attributi	utente	squadra						
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 utente ref Utente(id);

Fk squadra ref Squadra(id);

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

13 **Relazione** ute_sfi (nome) Derivante da: **entità** | **relationship** (cerchiare)

Attributi	utente	sfida						
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 utente ref Utente(id);

Fk sfida ref SfidaSingola(id);

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

14 **Relazione** ..... (nome) Derivante da: **entità** | **relationship** (cerchiare)

Attributi								
Domini								

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

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

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

15 **Relazione** ..... (nome) Derivante da: **entità** | **relationship** (cerchiare)

Attributi								
Domini								

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

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

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

Ulteriori vincoli esterni

Per ogni ulteriore vincolo esterno (non ancora espresso perché non definibile mediante vincoli di chiave, foreign key, ennupla, dominio, inclusione), progettare un trigger che lo implementi, definendo: (a) gli eventi da intercettare (inserimento, modifica, eliminazione di 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.

Trigger: [V.nasce_prima_di_attivita]

op: Insert o Update in Attivita
quando: post

```
isError:= EXISTS ( SELECT *
                   FROM Utente u
                   WHERE u.id=new.utente
                   AND cast(new.inizio as Date) < u.nascita);
```

Se isError=TRUE raise exception
Altrimenti permetti operazione

Trigger: [V.no_partecipa_stessa_sfida]

op: Insert o Update su squ_ute
quando: post

```
isError := EXISTS (SELECT *
                   FROM squ_ute, Squadra sq, Squadra sq1
                   WHERE squ_ute.utente=new.utente
                   AND squ_ute.squadra<>new.squadra
                   AND sq.id=squ_ute.squadra
                   AND sq1.id=new.squadra
                   AND sq.sfida=sq1.sfida);
```

Se isError=TRUE raise exception
Altrimenti permetti operazione

Trigger: [no_amicizia_incrociata]

op: Insert o Update su amicizia
quando: post

```
isError := EXISTS( SELECT *
                   FROM amicizia a
                   WHERE a.u1=new.u2
                   AND a.u2=new.u1);
```

Se isError=TRUE raise exception
Altrimenti permetti operazione

[continua alla pagina seguente]

Risposta alla Domanda 7 (segue)

Trigger: [V.limite_membri_squadra]
op: Insert o Update su squ_ute

```
isError:= EXISTS( SELECT * FROM (SELECT count(*) as tot
                                 FROM Utente u, squ_ute
                                 WHERE squ_ute.squadra = new.squadra
                                 AND squ_ute.utente = u.id),
                                (SELECT s.maxSq as M
                                 FROM Squadra sq, SfidaSquadre s
                                 WHERE sq.id = new.squadra
                                 AND sq.sfida = S.id )
                WHERE tot > M);
```

Se isError = TRUE raise exception
Altrimenti permetti operazione

Trigger: [V.coerenza_su_istanti_e_ordini2]
op: Insert o Update Posizione
quando: post

```
isError := EXISTS( SELECT
                   FROM Posizione pos
                   WHERE new.percorso = pos.percorso
                   AND (pos.ord < new.ord
                   AND pos.ist > new.ist)
                   OR (pos.ord > new.ord
                   AND pos.ist < new.ist);
```

Se isError = TRUE raise exception
Altrimenti permetti operazione

Trigger: [V.percorso_pubblico]
op: Insert o Update su Percorso
quando: post

```
isError := new.pub = TRUE AND (new.nome = NULL OR new.commento = NULL)
           OR new.pub = FALSE AND (new.nome <> NULL OR new.commento <> NULL);
```

Se isError = TRUE raise exception
Altrimenti permetti operazione

Risposta alla Domanda 7 (segue)

Trigger: [V.successione_ordini]
op: Insert o Update Posizione
quando: post

```
OK = EXISTS ( SELECT *
              FROM Percorso per, Posizione pos
              WHERE (per.id = new.percorso
              AND pos.id<> new.id
              AND pos.percorso =per.id
              AND new.ord = pos.ord-1)
              OR new.ord =1 );
```

Se OK = FALSE raise exception
Altrimenti permetti operazione

Trigger: [V.istante_pos_fra_inizio_e_fine]
op: Insert o Update Posizione
quando: post

```
isError: EXISTS (SELECT *
                 FROM Percorso per, Attivita at
                 WHERE per.id = new.percorso
                 AND per.attivita = at.id
                 AND (new.ist < at.inizio OR new.ist > at.fine));
```

Se isError = TRUE raise exception
Altrimenti permetti operazione

Trigger: [V.disgiunzione_attivita_di_uno_stesso_utente]
op: Insert o Update Attivita
quando: dopo

```
isError := EXISTS ( SELECT *
                    FROM Attivita at
                    WHERE at.utente = new.utente
                    AND (new.inizio, new.fine)
                    OVERLAPS (at.inizio, at.fine) );
```

Se isError = TRUE raise exception
Altrimenti permetti operazione

Risposta alla Domanda 7 (segue)

Trigger: [V.coerenza_su_istanti_e_ordini]

op: Insert o Update Posizione
quando: post

```
isError := EXISTS(SELECT *
                  FROM Posizione pos1, Posizione pos2
                  WHERE pos1.percorso = new.percorso
                  AND pos1.ord = new.ord-1
                  AND pos2.percorso = new.percorso
                  AND pos2.ist > pos1.ist AND
                  pos2.ist < new.ist) OR

           EXISTS(SELECT *
                  FROM Posizione pos1, Posizione pos2
                  WHERE pos1.percorso = new.percorso
                  AND new.ord = pos.ord-1
                  AND pos2.percorso = new.percorso
                  AND pos2.ist < pos1.ist AND
                  pos2.ist > new.ist);
```

Se isError = TRUE raise exception
Altrimenti permetti operazione

Trigger: [V.nasce_prima_di_sfida]

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

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

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

Risposta

```
CREATE FUNCTION dist(per:Integer):Real_GEZ  //dato un Percorso(id) restituisce la distanza

begin:
    Q= SELECT  SUM( SQRT( POWER(p1.lat-p2.lat,2)+POWER(p1.long-p2.long,2)))
       FROM Posizione p1, Posizione p2
       WHERE  p1.ord = p2.ord - 1
       AND p1.percorso=p2.percorso
       AND p1.percorso = :per

    return Q
end



resoconto(u:Integer, i:DateTime, F: DateTime): Insieme(<Stringa, Real_GEZ, Real_GEZ>)

     if(i>F): genera errore

    SpOre =(SELECT at.sport, sum(at.fine-at.inizio) as ore_dedicate
           FROM Attivita at
           WHERE at.utente=:u
           AND at.inizio>=:i AND  at.fine<= :F
           GROUP BY at.sport)

    SpDist = (SELECT at.sport, sum(dist(per.id)) as distanza_totale
           FROM Attivita at JOIN Percorso per on per.attivita = at.id
           WHERE at.utente= :u
           AND at.inizio>=:i AND  at.fine<= :F
           GROUP BY at.sport)

    Q = SELECT *
       FROM  SpDist  FULL  OUTER JOIN  SpOre  on  SpOre.sport=SpDist.sport ;

    Return Q
```

[Versione 2024-03-29]

[continua alla pagina seguente]

Risposta alla Domanda 8 (segue)

```
Sport_moda_amici(u:Integer, i:DateTime, f:DateTime): Insieme(<String>)

    if(i>f): genera errore

    A=EXISTS(SELECT * FROM amicizia WHERE u1=:u OR u2=:u)

    if(A=FALSE): genera errore

    Q=(SELECT at.sport, sum(at.fine-at.inizio) as ore
       FROM amicizia am, Attivita at
       WHERE (u1=:u
       AND at.utente=u2) OR
       (u2=:u AND at.utente=u1)
       AND at.inizio>=:i AND
       at.Fine<=:F
       GROUP BY at.sport)

    S=SELECT sport
      FROM Q
      WHERE ore=(SELECT MAX(ore) FROM Q);

    Return S
```

[Versione 2024-03-29]

Tempo totale stimato per svolgere questa prova: 180 minuti (tempo totale concesso: 300 minuti).

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]

[Spazio per minute. Questa pagina non sarà valutata a meno che non sia puntata da pagine precedenti.]