



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: .....



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.

## 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.)

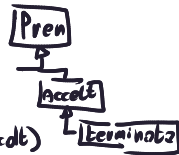
## Risposta

### 1. Utente

- 1.1 nome
- 1.2 cognome
- 1.3 sesso
- 1.4 città di residenza
- 1.5 ospitalità da offrire/casa
  - 1.5.1 distanza dal centro KM
  - 1.5.2 dist dalla stazione
  - 1.5.3 num membri famiglia (adulti/bambini)
  - 1.5.4 posti letto
    - 1.5.4.1 singolo/doppio
    - 1.5.4.2 camera sep/comune
    - 1.5.4.3 divano?
- 1.6 date di non disponibilità (no intersezioni)

### 2. Prenotazione

- 2.1 richiedente
- 2.2 alloggio
- 2.3 posti ( $\leq$  posti tot alloggio)
- 2.4 Accettata/Rifiutata
  - 2.4.1 se Accettata?
    - 2.4.1.1 terminata?
    - 2.4.1.2 voto 0..5 ospitato
    - 2.4.1.3 voto 0..5 ospitante (Facolt)
- 2.5 inizio (Data)
- 2.6 Fine (Data)



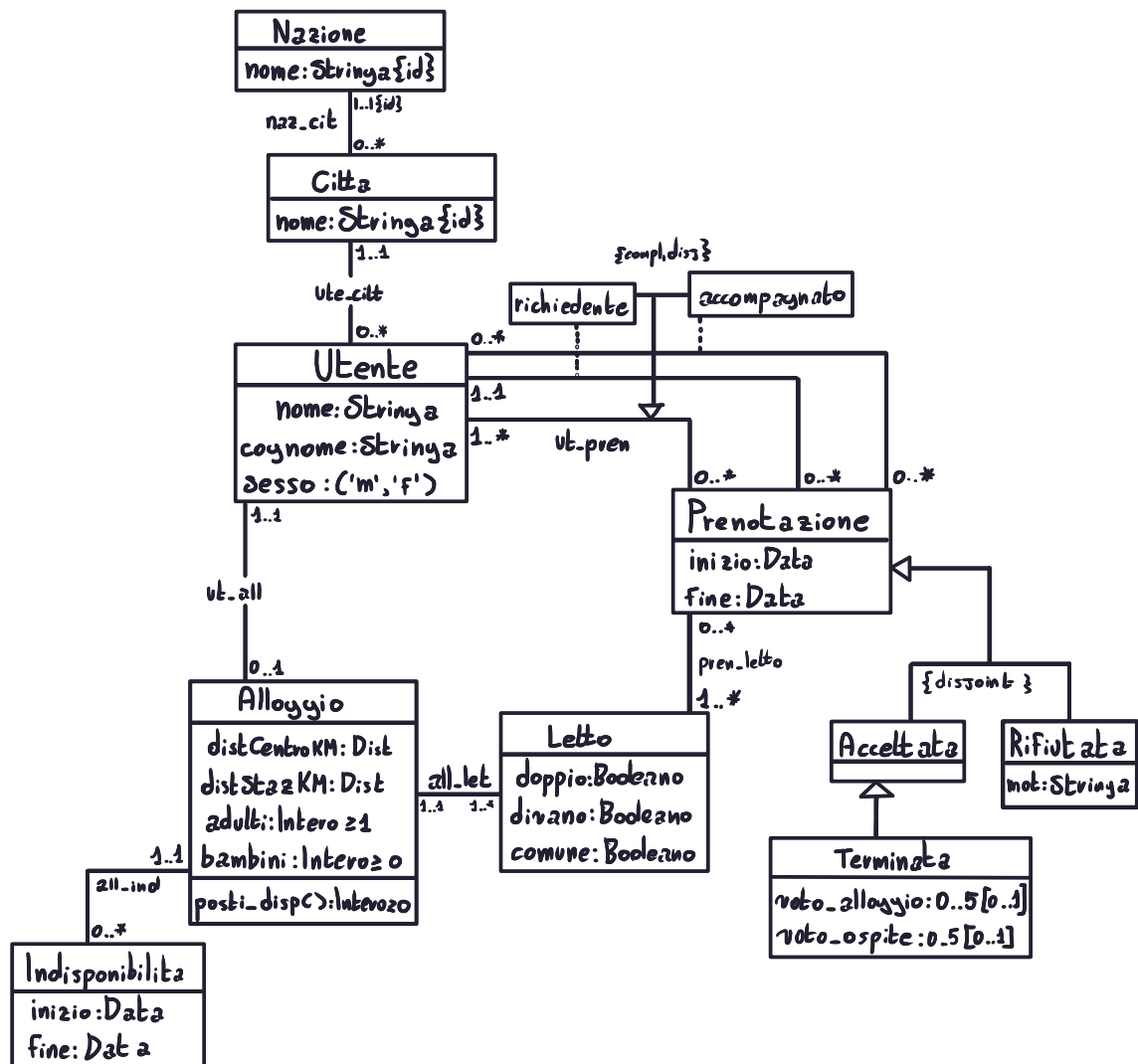
Risposta alla Domanda 1 (segue)

**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.



Risposta alla Domanda 2 (segue)



**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  $r(x_1, \dots, x_n)$ :  $x_1, \dots, x_n$  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  $a(x_1, \dots, x_n, v)$ : uno dei valori dell'attr.  $a$  dell'istanza  $(x_1, \dots, x_n)$  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

<div> <div>1</div> <div>Tipo: <b>Entità</b>   Relationship (cerchiare)</div> </div> <div> <div>Nome: <b>Letto</b></div> <table> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> </table> <div> (*) solo se diversa da (1,1) </div> <div> <div>Vincoli:</div> <div> <div>[V. no. 2 prenotazioni accettate - intersecano]</div> <div> <math>\forall l, p1, p2, i1, i2, f1, f2</math> <div> <div>[Letto(l) <math>\wedge</math> pren. letto(l, p1) <math>\wedge</math> pren. letto(l, p2) <math>\wedge</math> Accettata(p1)</div> <div> <div><math>\wedge</math> Accettata(p2)</div> <div> <math>\wedge</math> inizio(p1, i1) <math>\wedge</math> inizio(p2, i2) <math>\wedge</math> Fine(p1, f1) <math>\wedge</math> Fine(p2, f2) </div> </div> </div> <div><math>\rightarrow [f1 &lt; i2 \vee f2 &lt; i1]</math></div> </div> </div> </div> </div>	attributo	dominio	moltepl. (*)				<div> <div>2</div> <div>Tipo: <b>Entità</b>   Relationship (cerchiare)</div> </div> <div> <div>Nome: <b>Prenotazione</b></div> <table> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> </table> <div> (*) solo se diversa da (1,1) </div> <div> <div>Vincoli:</div> <div> <div>[V. inizio - prima - fine]</div> <div> <math>\forall p, i, f</math> <div> <div>[Prenotazione(p) <math>\wedge</math> inizio(p, i) <math>\wedge</math> Fine(p, f)]</div> <div><math>\rightarrow i &lt; f</math></div> </div> </div> <div> <div>[V. istante - terminazione]</div> <div> <math>\forall p, f, now, d</math> <div> <div>[Terminata(p) <math>\wedge</math> Fine(p, f) <math>\wedge</math> Adesso(now) <math>\wedge</math> Data(now, d)]</div> <div><math>\rightarrow f \leq d</math></div> </div> </div> </div> </div> </div> </div>	attributo	dominio	moltepl. (*)			
attributo	dominio	moltepl. (*)											
attributo	dominio	moltepl. (*)											

<div> <div>3</div> <div>Tipo: <u>Entità</u>   Relationship (cerchiare)</div> </div> <div> <div>Nome: Prenotazione</div> <table> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> <tr> <td colspan="3"></td> </tr> </table> <div>(*) solo se diversa da (1,1)</div> <div> <div>Vincoli:</div> <div> <div>[V.utente-no-pren-che-si-intersecano]</div> <div> <math>\forall u, p_1, p_2, i_1, i_2, f_1, f_2 \neg R_{\text{Rifiutata}}(p_1) \wedge \neg R_{\text{Rifiutata}}(p_2)</math> </div> <div> <math>[Utente(u) \wedge ute\_pren(u, p_1) \wedge ute\_pren(u, p_2) \wedge</math> </div> <div> <math>inizio(p_1, i_1) \wedge inizio(p_2, i_2) \wedge Fine(p_1, f_1) \wedge Fine(p_2, f_2)]</math> </div> <div> <math>\rightarrow [f_1 &lt; i_2] \vee [f_2 &lt; i_1]</math> </div> <div> <div>[V.no-prenotazioni-casa-propria]</div> <div> <math>\forall u, a, l [Utente(a) \wedge Letto(l) \wedge ut\_all(u, a) \wedge</math> </div> <div> <math>all\_let(a, l)] \rightarrow [\neg \exists p Prenotazione(p) \wedge pren\_letto(p, l)</math> </div> <div> <math>\wedge ute\_pren(u, p)]</math> </div> </div> </div> </div> </div>	attributo	dominio	moltepl. (*)				<div> <div>5</div> <div>Tipo: <u>Entità</u>   Relationship (cerchiare)</div> </div> <div> <div>Nome: Indisponibilit�</div> <table> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> <tr> <td colspan="3"></td> </tr> </table> <div>(*) solo se diversa da (1,1)</div> <div> <div>Vincoli:</div> <div> <div>[V.no-pren-durante-indisp]</div> <div> <math>\forall i, a, l, i', f, p, i', p'</math> </div> <div> <math>[Indisponibilit�(i) \wedge Alloggio(a) \wedge all\_ind(a, i) \wedge inizio(i, i')</math> </div> <div> <math>\wedge Fine(i, f) \wedge Letto(l) \wedge all\_let(a, l) \wedge</math> </div> <div> <math>pren\_letto(p, l) \wedge inizio(p, i') \wedge Fine(p, f')]</math> </div> <div> <math>\rightarrow [f &lt; i' \vee f' &lt; i]</math> </div> <div> <div>[V.inizio-ind-maggiore-Fine]</div> <div> <math>\forall i, i', f [Indisponibilit�'(i') \wedge inizio(i', i) \wedge Fine(i', f)]</math> </div> <div> <math>\rightarrow i' &lt; f</math> </div> </div> </div> </div> </div>	attributo	dominio	moltepl. (*)			
attributo	dominio	moltepl. (*)											
attributo	dominio	moltepl. (*)											

<div>4</div> <div>Tipo: Entità   <u>Relationship</u> (cerchiare)</div> <div>Nome: Prenotazione.....</div> <table><thead><tr><th>attributo</th><th>dominio</th><th>moltepl. (*)</th></tr></thead><tbody><tr><td colspan="3"> </td></tr></tbody></table> <div>(*) solo se diversa da (1,1)</div> <div>Vincoli:</div> <div><div>[V.pren_stesso_alloggio]</div><div><math>\forall p, l_1, l_2, a</math></div><div><math>[Prenotazione(p) \wedge Alloggio(a) \wedge pren\_letto(p, l_1) \wedge</math></div><div><math>pren\_letto(p, l_2) \wedge all\_let(a, l_1)] \rightarrow all\_let(a, l_2)</math></div><div>[V.udata_tutte]</div><div><math>\forall p, u</math></div><div><math>[Utente(p) \wedge \neg Terminata(p) \wedge \neg Rifiutata(p) \wedge ute\_pren(u, p)]</math></div><div><math>\rightarrow [\forall p_1 Terminata(p_1) \wedge ute\_pren(u, p_1) \wedge</math></div><div><math>\exists \kappa voto\_alloggio(p_1, \kappa)]</math></div></div>	attributo	dominio	moltepl. (*)				<div>6</div> <div>Tipo: Entità   Relationship (cerchiare)</div> <div>Nome: .....</div> <table><thead><tr><th>attributo</th><th>dominio</th><th>moltepl. (*)</th></tr></thead><tbody><tr><td colspan="3"> </td></tr></tbody></table> <div>(*) solo se diversa da (1,1)</div> <div>Vincoli:</div>	attributo	dominio	moltepl. (*)			
attributo	dominio	moltepl. (*)											
attributo	dominio	moltepl. (*)											

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

### Alloggio

posti\_disp(): Intero  $\geq 0$

• pre-condizioni: nessuna

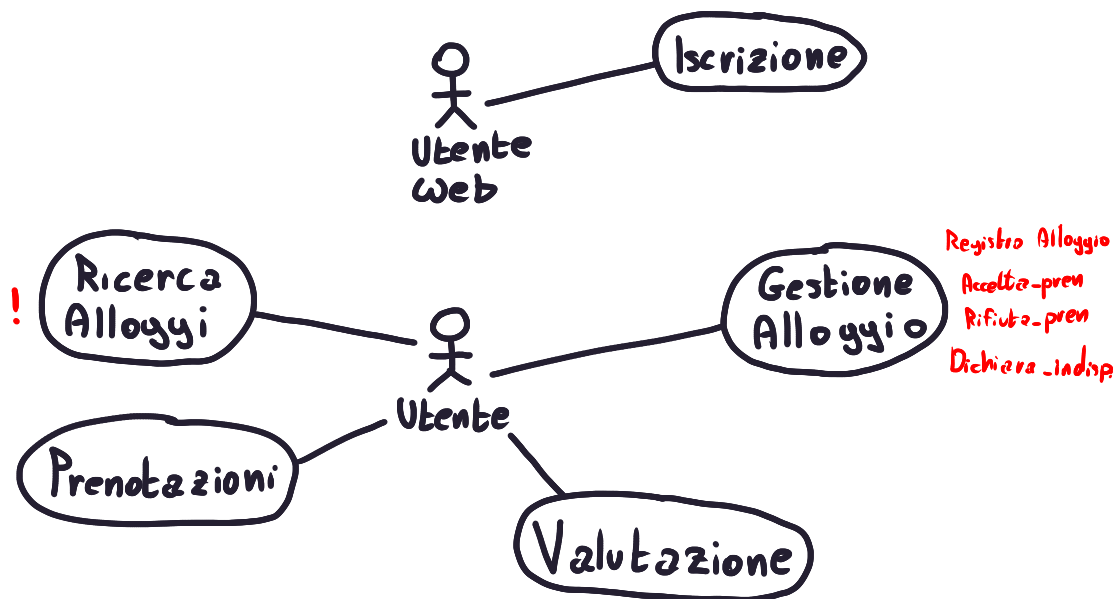
• pre-condizioni:  $PD = \left\{ \ell \mid \begin{array}{l} \text{all\_let}(\ell, \ell) \wedge \neg \exists p \text{ Accettata}(p) \wedge \\ \neg \text{Terminata}(p) \wedge \text{pren\_letto}(p, \ell) \wedge \\ \text{doppio}(\ell, \text{TRUE}) \end{array} \right\}$

$P = \left\{ \ell \mid \begin{array}{l} \text{all\_let}(\ell, \ell) \wedge \neg \exists p \text{ Accettata}(p) \wedge \\ \neg \text{Terminata}(p) \wedge \text{pren\_letto}(p, \ell) \wedge \\ \text{doppio}(\ell, \text{FALSE}) \end{array} \right\}$

Result =  $|P| + 2 \cdot |PD|$

**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.

### Mancano le segnature degli usecase

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

Operazioni dello use-case:

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

Operazioni dello use-case:

**3** Specifica use-case: ..... (nome use-case)

Operazioni dello use-case:

**4** Specifica use-case: ..... (nome use-case)

Operazioni dello use-case:

**5** Specifica use-case: ..... (nome use-case)

Operazioni dello use-case:

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

Operazioni dello use-case:

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

Operazioni dello use-case:

**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

### Ricerca Alloggi

$cerca\_ospitanti(c:Città, dal:Data, al:Data, n:Interò): Alloggio[0..*]$

• pre-cond:  $al > dal$

• post-cond: 
$$A = \left\{ a \mid \begin{array}{l} Alloggio(a) \wedge [\exists u \text{ utt-all}(u,a) \wedge \text{ute-citt}(u,c)] \\ \wedge \exists k \text{ posti\_disp}(a,k) \wedge k \geq n \wedge [\forall in,i,f \\ [\text{indisponibilita}(in) \wedge \text{inizio}(in,i) \wedge \text{fine}(in,f) \wedge \\ a1\_ind(a,in) \rightarrow [f < dal \vee al < i]]] \end{array} \right\}$$

Result = A

## 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 domain Stringa as varchar NOT NULL;`

`Create domain voto as Integer check(value >= 0 and value < 6);`

`create type gen as enum('M','F');`

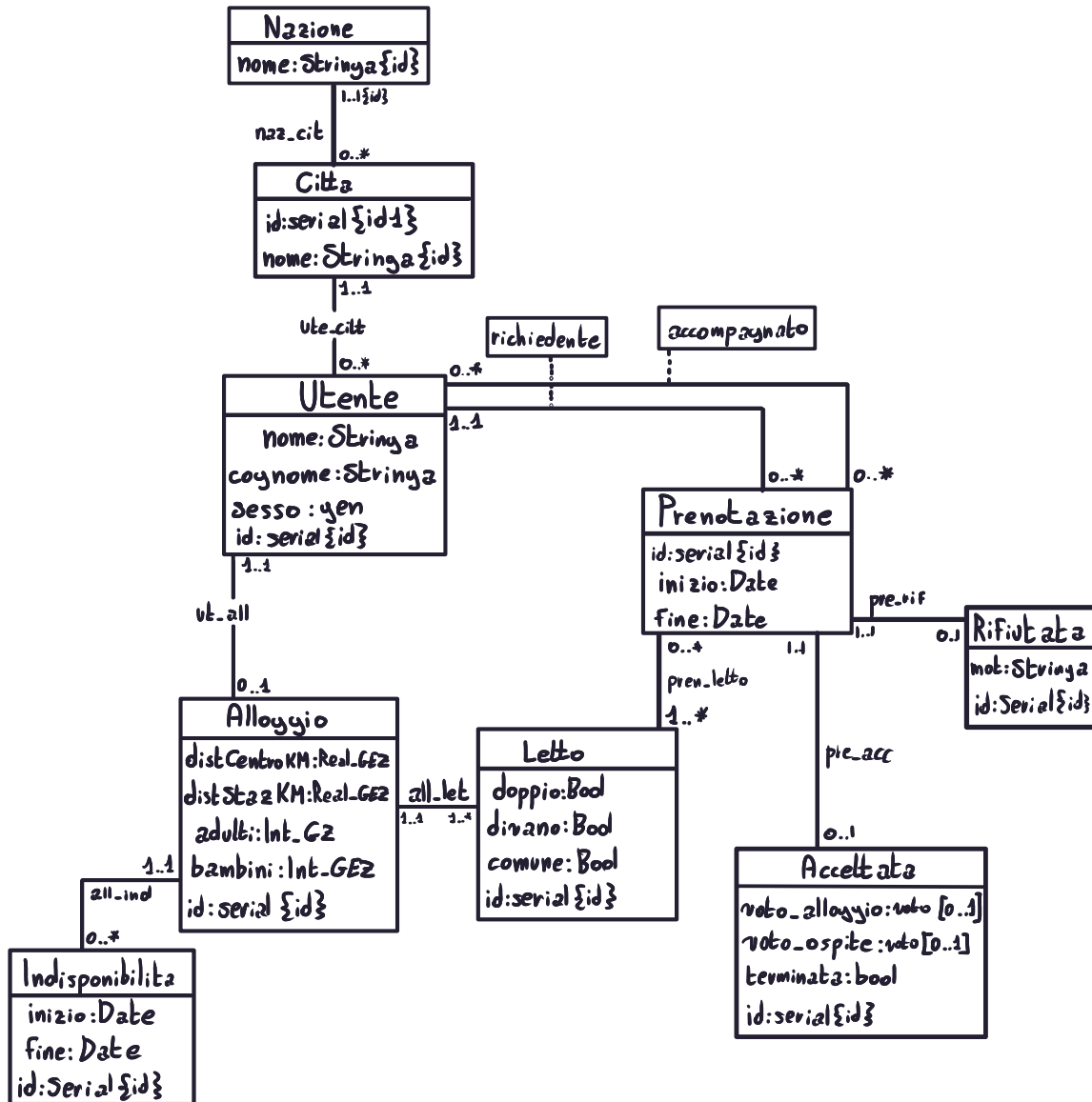
`create domain Int-GZ as Integer check(value > 0);`

`create domain Int-GEZ as Integer check(value >= 0);`

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



## Diagramma ER ristrutturato



Breve descrizione delle scelte effettuate durante la ristrutturazione

Fusione su *accettato*

Sostituzione con *ass.* su *Prenotazioni*

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)

$[V. utente\_no\_pren\_che\_si\_intersecano]$

$\forall u, p_1, p_2, i_1, i_2, f_1, f_2 \neg Rifutata(p_1) \wedge \neg Rifutata(p_2)$

$[Utente(u)]$

$\wedge [richiedente(u, p_1) \vee accompagnato(u, p_1)]$

$\wedge [richiedente(u, p_2) \vee accompagnato(u, p_2)]$

$inizio(p_1, i_1) \wedge inizio(p_2, i_2) \wedge fine(p_1, f_1) \wedge fine(p_2, f_2)$

$\rightarrow [f_1 < i_2] \vee [f_2 < i_1]$

Ne mancano altri  
da riscrivere

$[V. no\_prenotazioni\_casa\_propria]$

$\forall u, a, l [Utente(a) \wedge letto(l) \wedge ut\_all(u, a) \wedge$

$all\_let(a, l) \rightarrow [\neg \exists p Prenotazione(p) \wedge pren\_letto(p, l)$

$\wedge [richiedente(u, p) \vee accompagnato(u, p)]]$

$[V. valuta\_tutte]$

$\forall p, u$

$[Utente(p) \wedge \neg Terminata(p) \wedge \neg Rifutata(p) \wedge [richiedente(u, p) \vee accompagnato(u, p)]]$

$\rightarrow [V p_1 Terminata(p_1) \wedge [richiedente(u, p_1) \vee accompagnato(u, p_1)]] \wedge$

$\exists k voto\_alloggio(p_1, k)]$

Risposta alla Domanda 6 (segue)

[V.terminazione\_pren]

$\forall p \text{ [Accettata}(p) \wedge [\exists k \text{ voto-ospite}(p,k) \vee \exists k' \text{ voto-alloggio}(p,k')] ] \rightarrow [\text{terminata}(p, \text{TRUE})]$

[V.compl-disj-pren]

$\forall p \text{ [Prenotazione}(p) \wedge \exists r \text{ pre-rif}(p,r) ] \rightarrow [\neg \exists a \text{ pre-acc}(p,a)]$

[V.Utente-richiede-non-accompagna]

$\forall u,p \text{ [Utente}(u) \wedge \text{richiedente}(u,p) ] \rightarrow [\neg \text{accompagnato}(u,p)]$

**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 Nazione..... (nome)** Derivante da: entità | relationship (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 relationship: .....

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

Attributi	<u>id</u>	nome	nazione					
Domini	serial	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 nazione ref Nazione(nome);

Unique(nome, nazione);

La relazione accorpa le relazioni che implementano le seguenti relationship: naz-cit .....

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

Attributi	<u>nome</u>	cognome	sex	<u>id</u>	citta			
Domini	Stringa	Stringa	gen	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 citta ref Citta(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: ute-cit .....

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

Attributi	distCentroKM	distStazKM	adulti	bambini	<u>id</u>	utente		
Domini	Real-GE2	Real-GE2	Int-G2	Int-G2	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 utente ref Utente(id);

Unique(utente);

La relazione accorpa le relazioni che implementano le seguenti relationship: ute-all .....

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

Attributi	<u>inizio</u>	<u>fine</u>	<u>id</u>	alloggio				
Domini	Date	Date	serial	Integer				

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

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

check(inizio <= fine);

FK alloggio ref Alloggio(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: ..... all-ind .....

6 Relazione Letto ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>id</u>	<u>doppio</u>	<u>divano</u>	<u>comune</u>	<u>alloggio</u>			
Domini	<u>serial</u>	<u>bool</u>	<u>bool</u>	<u>bool</u>	<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 alloggio ref Alloggio(id); alloggio ≤ Alloggio(id);

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

7 Relazione Prenotazione .... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>id</u>	<u>inizio</u>	<u>fine</u>	<u>richiedente</u>				
Domini	<u>serial</u>	<u>Date</u>	<u>Date</u>	<u>id</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 richiedente ref Utente(id); check(inizio ≤ fine);

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

8 Relazione accompagnato ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>utente</u>	<u>prenotazione</u>						
Domini	<u>Integer</u>	<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 utente ref Utente(id);

fk prenotazione ref Prenotazione(id);

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

9 Relazione Rifiutata ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>mot</u>	<u>id</u>	<u>prenotazione</u>					
Domini	<u>Stringa</u>	<u>serial</u>	<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):

unique(prenotazione);

fk prenotazione ref Prenotazione(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: pre-rif .....

10 Relazione Accettato ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>voto_alloggio</u> *	<u>voto_ospite</u> *	<u>terminata</u>	<u>id</u>	<u>prenotazione</u>			
Domini	<u>voto</u>	<u>voto</u>	<u>bool</u>	<u>serial</u>	<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):

unique(prenotazione); check(((voto\_alloggio <> NULL OR voto\_ospite <> NULL) AND terminata = TRUE) OR terminata = 'false'); check(terminata = 'true' AND fine < cast(NOW() as Date)) OR terminata = 'false');

fk prenotazione ref Prenotazione(id);

La relazione accorpa le relazioni che implementano le seguenti relationship: pre-acc .....

6 Relazione pren.letto ..... (nome) Derivante da: entità | relationship (cerchiare)

Attributi	<u>pren</u>	<u>letto</u>						
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 pren ref Prenola e:ou(id)

FK letto ref Letto(id)

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

7 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 enupla, di dominio):

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

8 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 enupla, di dominio):

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

9 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 enupla, di dominio):

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

10 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 enupla, di dominio):

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

[V. no. inters. prev. accettate]

Trigger: Insert o Update Accettata

```
Error = EXISTS ( SELECT *  
FROM Accompagnato ac, Utente u, Prenotazione pr  
WHERE ac.utente = u.id AND  
(u.id = new.richiedente OR  
ac.prenotazione = new.id) AND  
ac.prenotazione = pr.id AND  
(pr.inizio, pr.fine) INTERSECT (new.inizio, new.fine) = 'True');
```

Se Error = True, segnala errore,  
altrimenti permetti operazione.

[V. no. 2\_ accettate sullo stesso letto]

Trigger: Insert o Update Accettata

```
FROM pren.letto pl, Accettata a  
WHERE pl.pren = new.id  
AND
```

**Ulteriori vincoli esterni**

Per ogni ulteriore vincolo esterno (non ancora espresso perché non definibile mediante vincoli di chiave, foreign key, enunpla, 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.



Risposta alla **Domanda 7** (segue)

**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**

Risposta alla **Domanda 8** (segue)

Matricola: .....

*Minute*

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.]





Matricola: .....

*Minute*

[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.]

