

## 2

## Specifiche dei Requisiti

Per l'applicazione sono di interesse i clienti (con nome, cognome ed indirizzo di residenza), i voli aerei, tutti giornalieri, e le prenotazioni fatte dai clienti. In particolare, dei voli interessa codice (una stringa), numero di miglia percorse, orari ed aeroporti di partenza ed arrivo (ad es., il volo "RA 1234" parte ogni giorno alle 13:50 dall'aeroporto di Roma Ciampino e arriva a Londra Stansted alle 15:50, percorrendo 1000 miglia), ed il velivolo usato, mentre delle prenotazioni è importante conoscere il momento in cui vengono effettuate.

Degli aeroporti da cui partono e arrivano voli bisogna rappresentare il codice (una stringa di 3 caratteri, ad es. "CIA") il nome, la città e lo stato dove è collocato, e l'ammontare delle tasse di decollo e atterraggio.

Dei velivoli è di interesse conoscere codice (una stringa), tipo, numero di posti e costo al miglio (ovvero quanto il velivolo costa alla compagnia per ogni miglio di volo).

Dei biglietti di un volo è di interesse conoscere il loro "prezzo base", base di partenza per il calcolo del prezzo finale. Il prezzo base dipende dal costo che la compagnia deve sostenere per effettuare il volo. Quest'ultimo è dato da due addendi: (i) il prodotto tra il numero di miglia effettuate e il costo al miglio del velivolo in uso e (ii) la somma delle tasse di decollo e atterraggio negli aeroporti di partenza e arrivo, rispettivamente. Il prezzo base di un biglietto è dato dal costo su descritto diviso per il numero di posti del velivolo, e incrementato del 20% (che è il ricarico che la compagnia vuole applicare).

Una prenotazione da parte di un cliente può essere relativa anche a più voli (almeno uno). Di ogni volo prenotato, interessa la data in cui il cliente vuole volare, ed il numero di posti richiesti.

Una prenotazione può, ma non obbligatoriamente, essere relativa anche ad un hotel, nel qual caso interessa conoscere la data di check-in, quella di check-out, e il numero di stanze prenotate (si ignorino, per semplicità, le diverse tipologie di stanze). Di un hotel sono di interesse il nome, l'indirizzo, la categoria (ovvero il numero di stelle, da una a cinque), la tariffa per stanza per notte, la città dove è collocato, la distanza dal centro

cittadino, e la possibile informazione sull'aeroporto più vicino, con relativa distanza (le distanze siano tutte in Km).

Alcuni clienti sono "frequent flyers". Di questi interessa conoscere il codice, la data di affiliazione, e il numero di miglia che hanno accumulato fino ad un certo momento. Il numero di miglia accumulate fino al momento richiesto è la somma delle miglia guadagnate con le prenotazioni effettuate a suo nome per voli prenotati dopo la data di affiliazione, ed effettuati fino a quel momento. In particolare, il guadagno di miglia relative ad un singolo volo è pari al numero di miglia percorse dal volo per il numero di posti prenotati. Se una prenotazione è relativa a più voli, le miglia guadagnate sono la somma di quelle relative ad ogni singolo volo.

Inoltre, sono previsti benefici per le prenotazioni che includono anche hotel (indipendentemente dalle date di check-in e check-out). In particolare, le miglia relative ad una prenotazione raddoppiano se questa prevede anche un hotel fino a 4 stelle, e triplicano se l'hotel prenotato è a 5 stelle.

1 { Il Sistema prenotazioni (un sistema esterno) necessita di calcolare il numero di posti disponibili all'istante corrente su un dato volo di una certa data. Il numero di posti disponibili è calcolato a partire dal numero di posti del velivolo che effettua il volo in questione, diminuito del numero di posti già prenotati (all'istante corrente).

Il Sistema prenotazioni deve anche poter calcolare il prezzo complessivo, all'istante corrente, di un certo numero di biglietti per un volo di un dato giorno. Tale prezzo è da considerarsi valido solo nel caso in cui il volo disponga, al momento corrente, di un numero sufficiente di posti per la data richiesta, e si calcola moltiplicando il prezzo di un biglietto singolo per il numero di biglietti richiesti.

Il prezzo di un singolo biglietto è altamente flessibile, ed è composto da diverse componenti, dovute a diversi fattori:

1. Il prezzo base che dipende dal volo;
2. Il numero di posti disponibili al momento corrente per la data di volo richiesta.

Il calcolo del prezzo del biglietto parte dal suo prezzo base, e subisce poi delle modifiche a seconda della disponibilità attuale di posti sulla data di volo richiesta. In particolare, al prezzo base si applica la seguente regola: se il numero di posti disponibili al momento della prenotazione per il volo in questione è maggiore della metà dei posti totali (ovvero quelli del velivolo che effettua il volo), si applica uno sconto del 2% per ogni posto libero oltre la metà. Al contrario, se il numero di posti disponibili è minore della metà, si applica un sovrapprezzo del 2% in modo del tutto analogo. Ad esempio, se il velivolo che effettua il volo ha 100 posti in totale, e, al momento della prenotazione, ne sono ancora liberi 53 per la data di volo richiesta, si applicherà uno sconto del 2% al prezzo base per tre volte (cosa diversa dall'applicare il 6% di sconto), mentre, se i posti liberi fossero 47, si applicherebbe un sovrapprezzo del 2% per tre volte.

Il Sistema prenotazioni vuole offrire anche il seguente servizio: data una città e una tariffa massima, vuole suggerire un insieme di hotel in quella città che abbiano tutti una

**3** { tariffa al più pari a quella indicata. La scelta degli hotel avviene secondo le seguenti regole (a parte quella sulla tariffa, che deve essere sempre rispettata):

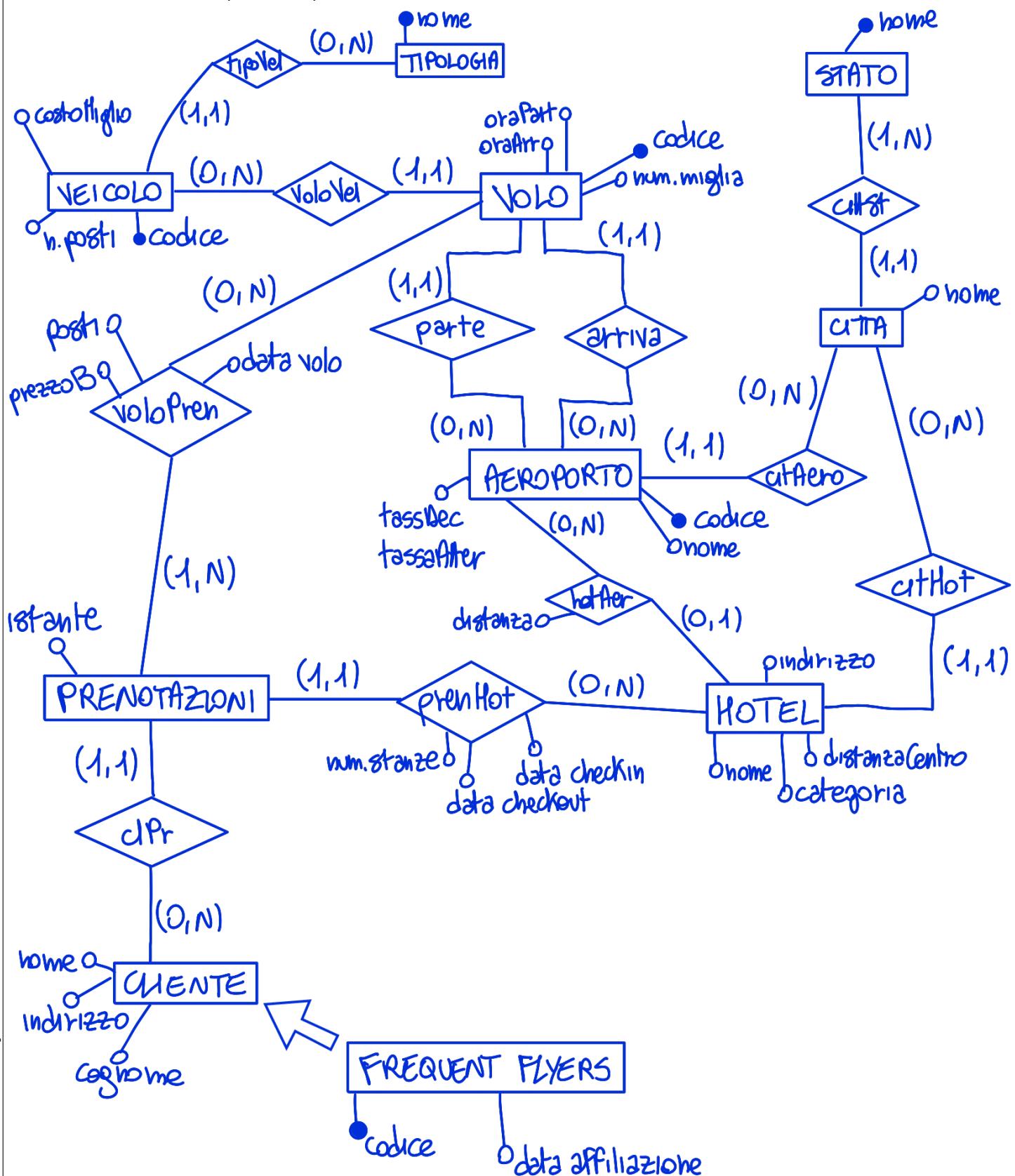
- Farà parte del risultato l'hotel più vicino al centro della città in questione con tariffa entro la soglia; sia questo hotel *A*.
- Farà inoltre parte del risultato qualunque altro hotel con lo stesso numero di stelle di *A* che abbia una distanza dal centro pari al più il 110% di quella di *A*.
- Infine, faranno parte del risultato anche quegli hotel che hanno più stelle di *A*, ma più lontani di *A* dal centro. In particolare, quelli per cui la distanza dal centro sia al massimo il 120% di quella di *A*.

**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  $r(x_1, \dots, x_n)$ :  $x_1, \dots, x_n$  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

<p>[1] Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <u>VEICOLO</u></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>codice</td> <td>str</td> <td></td> </tr> <tr> <td>n. posti</td> <td>int &gt; 0</td> <td></td> </tr> <tr> <td>costoMiglio</td> <td>Denaro</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltep. (*)	codice	str		n. posti	int > 0		costoMiglio	Denaro		<p>[2] Tipo: <b>Entità</b>   Relationship (cerchiare)</p> <p>Nome: <u>VOLO</u></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltep. (*)</th> </tr> </thead> <tbody> <tr> <td>codice</td> <td>str</td> <td></td> </tr> <tr> <td>num.miglia</td> <td>int &gt; 0</td> <td></td> </tr> <tr> <td>oraPartenza</td> <td>ora</td> <td></td> </tr> <tr> <td>oraArrivo</td> <td>ora</td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> <p><math>\langle V. Volo. aeroperti \neq V. Volo. aeroperti \rangle</math></p> <p><math>\forall v, ap, aa \ Volo(v) \wedge parte(v, ap)</math></p> <p><math>\wedge arriva(v, aa) \rightarrow ap \neq aa</math></p>	attributo	dominio	moltep. (*)	codice	str		num.miglia	int > 0		oraPartenza	ora		oraArrivo	ora	
attributo	dominio	moltep. (*)																										
codice	str																											
n. posti	int > 0																											
costoMiglio	Denaro																											
attributo	dominio	moltep. (*)																										
codice	str																											
num.miglia	int > 0																											
oraPartenza	ora																											
oraArrivo	ora																											

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

Nome: **STATO**

attributo	dominio	moltepl. (*)
-----------	---------	--------------

<b>nome</b>	<b>str</b>
-------------	------------

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

Vincoli:

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

Nome: **CITTÀ**

attributo	dominio	moltepl. (*)
-----------	---------	--------------

<b>nome</b>	<b>str</b>
-------------	------------

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

Vincoli:

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

Nome: **AEROPORTO**

attributo	dominio	moltepl. (*)
-----------	---------	--------------

<b>codice</b>	<b>str di 3 char</b>
---------------	----------------------

<b>nome</b>	<b>str</b>
-------------	------------

<b>tassaDec</b>	<b>Denaro</b>
-----------------	---------------

<b>tasseAff</b>	<b>Denaro</b>
-----------------	---------------

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

Vincoli:

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

Nome: **PRENOTAZIONI**

attributo	dominio	moltepl. (*)
-----------	---------	--------------

<b>istante</b>	<b>dataora</b>
----------------	----------------

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

Vincoli:

$\langle V.Prenotazione.istante.dataVolo \rangle$

$\forall p, ip \ Prenotazione(p) \wedge istante(ip, p)$

$\wedge voloPren(v, ip) \wedge data(v, p, dv) \wedge$

$oraPart(v, ov) \wedge dataora(iv) \wedge$

$data(iv, dv) \wedge ora(iv, ov)$

$\rightarrow iv > i$

<input checked="" type="checkbox"/> 7	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	CHENTE	
attributo	dominio	moltep. (*)
nome	str	
cognome	str	
Indirizzo	Indirizzo	

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

Vincoli:

<input checked="" type="checkbox"/> 9	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	FREQUENT PLAYERS	
attributo	dominio	moltep. (*)
codice	str	
data affiliaz.	data	

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

Vincoli:

<input checked="" type="checkbox"/> 8	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	prenVolo	
attributo	dominio	moltep. (*)
posti	int > 0	
dataVolo	data	

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

Vincoli:

<input checked="" type="checkbox"/> 10	Tipo: <b>Entità</b>   Relationship (cerchiare)	
Nome:	HOTEL	
attributo	dominio	moltep. (*)
nome	str	
indirizzo	Indirizzo	
categoria	int [1,5]	
distCentrokm	reale ≥ 0	
tariffa	Denaro	

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

Vincoli:

<p>11 Tipo: Entità   Relationship (cerchiare)</p> <p>Nome: <u>preNot</u></p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td><u>checkIn</u></td> <td><u>data</u></td> <td></td> </tr> <tr> <td><u>checkOut</u></td> <td><u>data</u></td> <td></td> </tr> <tr> <td><u>n.stanze</u></td> <td><u>int &gt; 0</u></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p> $\forall p, h, d_{in}$ $\text{Prenotazione}(p) \wedge \text{preNot}(p, h) \wedge$ $\text{dataCheckIn}(p, h, d_{in}) \wedge$ $\text{dataCheckOut}(p, h, d_{out})$ $\rightarrow d_{in} < d_{out}$	attributo	dominio	moltepl. (*)	<u>checkIn</u>	<u>data</u>		<u>checkOut</u>	<u>data</u>		<u>n.stanze</u>	<u>int &gt; 0</u>		<p>13 Tipo: Entità   Relationship (cerchiare)</p> <p>Nome: .....</p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)			
attributo	dominio	moltepl. (*)																	
<u>checkIn</u>	<u>data</u>																		
<u>checkOut</u>	<u>data</u>																		
<u>n.stanze</u>	<u>int &gt; 0</u>																		
attributo	dominio	moltepl. (*)																	

<p>12 Tipo: Entità   Relationship (cerchiare)</p> <p>Nome: .....</p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	attributo	dominio	moltepl. (*)				<p>14 Tipo: Entità   Relationship (cerchiare)</p> <p>Nome: .....</p> <table border="1"> <thead> <tr> <th>attributo</th> <th>dominio</th> <th>moltepl. (*)</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>(*) solo se diversa da (1,1)</p> <p>Vincoli:</p>	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

Domino Denaro:

- importo: reale  $\geq 0$

Il assunto che esista solo è

Domino Indirizzo:

VIA: str

CIVICO: int  $> 0$  (0,1)

CAP: str di 5 cifre

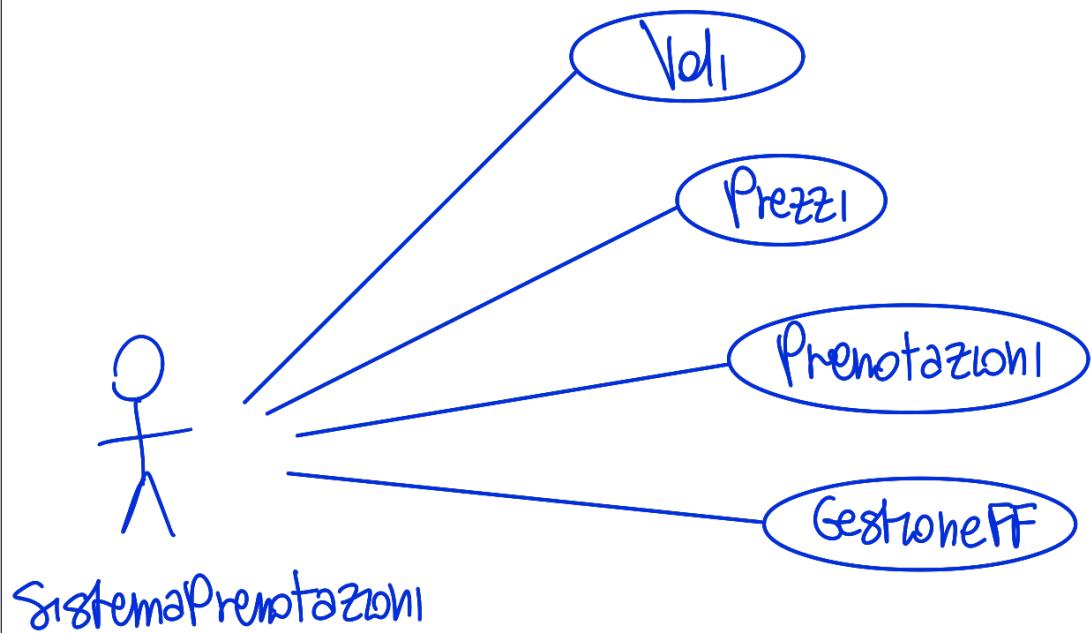
<\!\!V.prenotazione.istante.checkIn >

$\forall p, h, i, ci, id \quad \text{Prenotazione}(p) \wedge \text{Hotel}(h) \wedge \text{istante}(p, i)$   
 $\wedge \text{prenot}(p, h) \wedge \text{dataCheckIn}(p, h, ci)$   
 $\wedge \text{data}(i, id) \rightarrow id < ci$

<\!\!V.Volo.maiOverbooking >

$\forall v, d, i \quad \text{Volo}(v) \wedge \text{data}(d, v) \wedge \text{dataora}(i)$   
 $\rightarrow \text{postiDisponibili}(v, d, i) \geq 0$

**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**

Specifiche Use-Case: Posti

posti disponibili ( $v: Volo, d: data, i: dataora$ ): intero

precondizioni: nessuna

postcondizioni:

Modif. del liv. estensionale dei dati: nessuna

Valore di ritorno: Sia

$$Z = \{(z, po) \mid \text{Prenotazione}(z) \wedge \text{voloPren}(z, v) \wedge \text{pax}(v, z, po) \\ \wedge \text{data}(v, z, d) \wedge \exists ip \text{ istante}(z, ip) \wedge ip < i\}$$

Sia cap il valore posti di Velivolo utilizzato per un Volo v, tale da soddisfare:  $\exists vel \text{ voloVel}(vel, v) \wedge \text{posti}(vel, cap)$

$$\text{result} = [cap - (\sum_{(z, po) \in Z} po)]$$

Specifiche UseCase: Prezzi

prezzoBase ( $v: Volo$ ): Denaro

precondizioni: nessuna

postcondizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: Siano: m, cm, cap, td e ta (rispettivamente: numero di miglia, costo al miglio, capacità del velivolo, tasse dec, tassa alter relative al Volo v) tali da soddisfare

$$\text{miglia}(m, v) \wedge \exists vel \text{ voloVel}(v, vel) \wedge \text{costoMiglio}(cm, vel) \\ \wedge \text{posti}(vel, cap) \wedge \exists ap, td \text{ parte}(v, ap) \wedge \text{tassaDec}(ap, td) \\ \wedge \exists aa \text{ arriva}(v, aa) \wedge \text{tasseDec}(aa, ta).$$

$$\text{result} = \left( \frac{m \cdot mc + ta + td}{cap} \right) \cdot (1 + 0.2)$$

prezzoComplessivoSingoloPosto(v: Volo, d: data) : Denaro

precondizioni: Posti.postiDisponibili(v, d, adesso) > 0

post condizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: Sia:

pd = Posti.postiDisponibili(v, d, adesso) > 0

Sia cap tale da soddisfare:  $\exists \text{vel} \text{ Velivo}(vel) \wedge \text{voloVel}(v, vel)$   
 $\wedge \text{posti}(vel, cap)$

Sia diff =  $(\lfloor \frac{cap}{2} \rfloor - cap)$ , result deve soddisfare:

diff ≤ 0 → result = Prezzi.prezzoBase(v) •  $(1 - 0,02)^{-\text{diff}}$

diff > 0 → result = Prezzi.prezzoBase(v) •  $(1 + 0,02)^{-\text{diff}}$

prezzoComplessivo(v: Volo, d: data, n: int > 0) : Denaro

precondizioni: Posti.postiDisponibili(v, d, adesso) ≥ n

post condizioni:

Mod. del liv. estens. dei dati: nessuna

Valore di ritorno:

result = n • Posti.prezzoComplessivoSingoloPosto(v, d)

## Specifiche Use-Case Prenotazioni

- suggerisciHotel( $c : \text{Città}, t : \text{Denaro} : \text{Hotel}$  (0,N) (Req. 9.5.)

precondizioni: nessuna

postcondizioni:

**Modifica del livello estensionale dei dati:** Il livello estensionale dei dati al termine dell'esecuzione della funzione differisce da quello di partenza come segue:

**Variazioni nel dominio di interpretazione:** nessuna

**Variazioni nelle ennuple di predici:**

**Valore di ritorno:** Sia  $CAT_{d_{\min}}$  l'insieme delle categorie degli hotel nella città  $c$  a distanza minima dal centro e a tariffa entro la soglia  $t$  (per comodità, l'insieme definisce anche la distanza dal centro, uguale per tutte le categorie di hotel riportate):

$$CAT_{d_{\min}} = \left\{ \begin{array}{l|l} \text{cat, d} & \exists h, ta \quad \text{Hotel}(h) \wedge \text{citHot}(c, h) \wedge \\ & \text{categoria}(h, \text{cat}) \wedge \text{tariffa}(h, ta) \wedge \\ & ta \leq t \wedge \text{distCentro}(h, d) \wedge \\ & \exists h', ta', d' \quad \text{Hotel}(h') \wedge \text{citHot}(c, h') \wedge \\ & \text{tariffa}(h', ta') \wedge ta' \leq t \wedge \\ & d' < d \end{array} \right\}$$

Sia  $\text{cat}_A$  la categoria più alta rappresentata in  $CAT_{d_{\min}}$ , e sia  $d_A$  la distanza dal centro associata. Formalmente,  $(\text{cat}_A, d_A)$  soddisfa la seguente formula (l'assegnamento è unico, perché gli elementi dell'insieme hanno tutti lo stesso valore per la componente  $d$ ):

$$(\text{cat}_A, d_A) \in \underset{(\text{cat}, d) \in CAT_{d_{\min}}}{\operatorname{argmax}} (\text{cat})$$

Siano:

$$\text{result} = \left\{ \begin{array}{l|l} h & \exists \text{cat}, d \quad \text{Hotel}(h) \wedge \text{citHot}(c, h) \wedge \text{categoria}(h, \text{cat}) \wedge \\ & \text{distCentro}(h, d) \wedge \text{cat} \geq \text{cat}_A \wedge \\ & \text{cat} = \text{cat}_A \rightarrow \\ & d \leq (1 + \text{DELTA\_DIST\_STESSA\_CAT}) \times d_A \wedge \\ & \text{cat} > \text{cat}_A \rightarrow \\ & d \leq (1 + \text{DELTA\_DIST\_CAT\_SUP}) \times d_A \end{array} \right\}$$

Con  $\text{Delta\_Dist\_Stessa\_Cat} = 0,1$

" - " -  $\text{Cat\_Sup} = 0,2$

Specifico Use-Case: GestioneFF

numMigliaVoli( $p$ : Prenotazione): intero > 0

precondizioni: nessuna

postcondizioni:

Mod. del liv. estens. dei dati: nessuna

Valore di ritorno: Sia:

$$V = \{(v, n, m) \mid \text{voloPrem}(v, p) \wedge \text{pax}(v, p, n) \wedge \text{miglia}(v, m)\}$$

$$\text{result} = \sum_{(v, n, m) \in V} n \times m$$

amplificazione( $p$ : Prenotazione): int  $\in [1, 3]$

precondizioni: nessuna

postcondizioni:

Mod. del liv. estensionale dei dati: nessuna

Val. di ritorno: Result soddisfa:

$$[(\neg \exists h \text{ prenlotel}(p, h)) \rightarrow \text{result} = 1]$$

↑

$$[(\exists h \text{ prenlotel}(p, h)) \rightarrow$$

$$\forall h, c \text{ prenlotel}(p, h) \wedge \text{cat}(c, h) \rightarrow$$
$$((c \leq 4 \rightarrow \text{result} = 2)$$

↑

$$(c = 5 \rightarrow \text{result} = 3))]$$

$\text{numMiglia}(p: \text{Prenotazione}): \text{intero} \geq 0$

precond: nessuna

postcond:

Mod. del liv. estens. dei dati: nessuna

Val di ritorno:

$\text{result} = \text{GestioneFF. numMigliaVoli}(v) \times \text{GestioneFF. amplificazione}(p)$

$\text{numMigliaFF}(f: \text{FF}, mom: \text{dataora}): \text{intero} > 0$

precond: nessuna

postcond:

Mod. del liv. estens. dei dati: nessuna

Val di ritorno: delta:

$P = \{ p \mid \text{dPr}(f, p) \wedge \exists a, i, d_i$

$\text{affiliazione}(a) \wedge \text{istante}(p, i) \wedge \text{data}(i, d_i)$   
 $\wedge d_i \geq a \wedge i < mom\}$

$\text{result} = \sum_{p \in P} \text{GestioneFF. numMiglia}(p)$

## 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 multivale 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 StringS as varchar(50);

create domain StringM as varchar(200);

create domain StringL as varchar(1000);

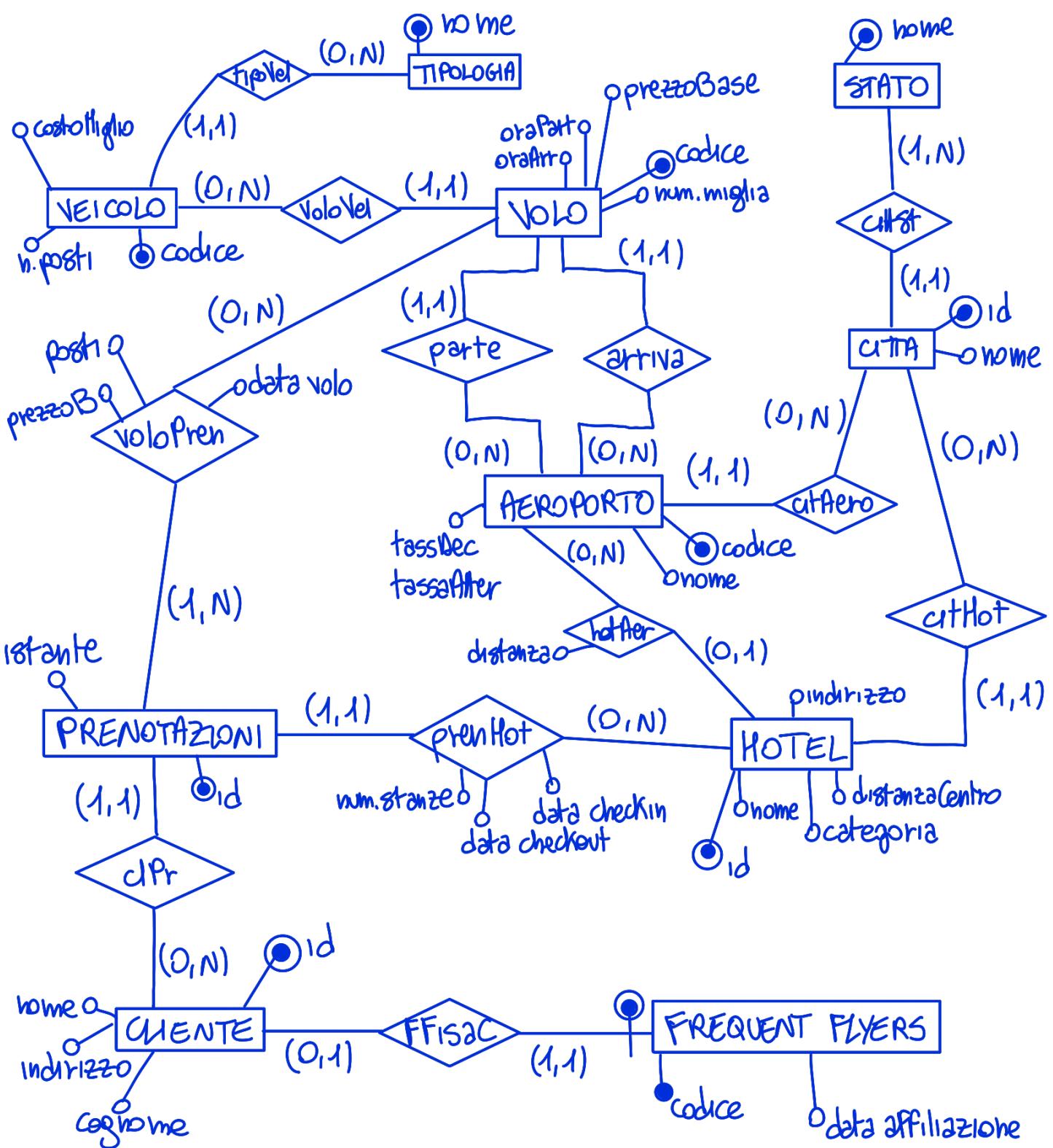
create domain IntegerGZ as integer  
check (value > 0);

create domain CategorieHotel as integer  
check (value ≥ 1 and value ≤ 5);

create type Indirizzo as (  
 via StringM  
 civico IntegerGZ (0,1)  
 cap char(5)  
)

create domain HoneyGZ as real  
check (value ≥ 0);

## Diagramma ER ristrutturato



Breve descrizione delle scelte effettuate durante la ristrutturazione

- 1) Non sono presenti attr. multivalue
- 2) Eliminazione 1s-a tramite sostituzione con relationship
- 3) Identificazione attr. primari
- 4) Eliminazione ridondanza con aggiunta dell'attributo prezzoBase

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. Volo. prezzoBase >

$\forall v, ap, aa, td, ta, vel, cm, m, val, cap$

$valVolo(v, val) \wedge prezzoBase(v, p) \rightarrow$

$$p = \left( \frac{m \cdot mc + ta + td}{cap} \right) \cdot (1 + 0.2)$$

**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.

<input type="checkbox"/> 1 Relazione <u>TIPLOGIA</u> .. (nome)	Derivante da: <b>entità</b>   relationship (cerchiare)
Attributi <u>nome</u>	
Domini <u>StringM</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 accorda le relazioni che implementano le seguenti relationship: .....

<input type="checkbox"/> 2 Relazione <u>VelVolo</u> ..... (nome)	Derivante da: <b>entità</b>   relationship (cerchiare)
Attributi <u>codice</u>   Posti   <u>KostoMigli</u>   <u>Tipologia</u>	
Domini <u>StringS</u>   <u>IntegerGZ</u>   <u>MoneyGEZ</u>   <u>StringM</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: tipologia references Tipologia(nome)

La relazione accorda le relazioni che implementano le seguenti relationship: tipVelo .....

<input type="checkbox"/> 3 Relazione <u>Volo</u> ..... (nome)	Derivante da: <b>entità</b>   relationship (cerchiare)
Attributi <u>codice</u>   pBase   orarioArrivo   orarioPartenza   miglia   <u>Velvolo</u>   <u>aeroPart</u>   <u>aeroArr</u>	
Domini <u>StringS</u>   <u>MoneyGEZ</u>   <u>Time</u>   <u>Time</u>   <u>IntegerGZ</u>   <u>StringS</u>   <u>char(3)</u>   <u>char(3)</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: velvolo refer VelVolo(codice)      PK: aeroPart refer Aeroporto(codice)

FK: aeroPart refer Aeroporto(codice)      ennupla: aeroArr ≠ aeroPart [V.Volo.aeroDiversi]

La relazione accorda le relazioni che implementano le seguenti relationship: voloVel, parteArriva .....

<input type="checkbox"/> 4 Relazione <u>Città</u> ..... (nome)	Derivante da: <b>entità</b>   relationship (cerchiare)
Attributi <u>id</u>   <u>nome</u>   <u>stato</u>	
Domini <u>Integer</u>   <u>StringM</u>   <u>StringM</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: stato refer Stato(nome)

seriale: i valori della colan. id sono generat. autom. dal DBMS

La relazione accorda le relazioni che implementano le seguenti relationship: cittSt .....

<input type="checkbox"/> 5 Relazione <u>Aeroporto</u> .... (nome)	Derivante da: <b>entità</b>   relationship (cerchiare)
Attributi <u>codice</u>   <u>nome</u>   <u>hasDecoll</u>   <u>hasArr</u>   <u>alt</u>	
Domini <u>char(3)</u>   <u>StringM</u>   <u>MoneyGEZ</u>   <u>MoneyGEZ</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: alt refer Citta(id)

La relazione accorda le relazioni che implementano le seguenti relationship: cittAero .....

<b>6 Relazione</b>	<u>VoloPrem</u> ..... (nome)	Derivante da: <b>entità</b>   <b>relationship</b> (cerchiare)
Attributi	pax   data   <u>volo</u>   <u>pren</u>	
Domini	integerGZ   date   stringS   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: pren refer Pren(tante)

FK: volo refer Volo(codice)

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

<b>7 Relazione</b>	<u>HOTEL</u> ..... (nome)	Derivante da: <b>entità</b>   <b>relationship</b> (cerchiare)
Attributi	<u>id</u>   nome   indirizzo   categoria   tariffa   distCentro   distAerop   aeroporto	* * * * *
Domini	integer   stringM   Indirizzo   CATNOTE   MoneyGZ   IntegerGZ   IntegerGZ   char(3)	*

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): (ulta: integer) ↑

FK: aeroporto refer Aeroporto(codice) PK: ulta refer Citta(id)

seriale: i valori di id gen. autom. ennupla: aeroporto ≠ NULL → distAerop ≠ NULL

La relazione accorda le relazioni che implementano le seguenti relationship: hotelAero, citHot .....

<b>8 Relazione</b>	<u>Pren</u> ..... (nome)	Derivante da: <b>entità</b>   <b>relationship</b> (cerchiare)
Attributi	<u>id</u>   istante   in*   out*   nstante*   hotel*   cliente	
Domini	integer   timestamp   date   date   IntegerGZ   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):

Ovvi pag 220

FK: hotel refer Hotel(id) seriale: val id. gen. automatic

FK: cliente refer Cliente(id) inclusione: id ∈ voloPrem (Pren)

La relazione accorda le relazioni che implementano le seguenti relationship: ..... prenHot, clPr

<b>9 Relazione</b>	<u>FF</u> ..... (nome)	Derivante da: <b>entità</b>   <b>relationship</b> (cerchiare)
Attributi	<u>codice</u>   affilazione   cliente	
Domini	stringS   date   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):

chiave: codice

FK: cliente ref Cliente(id)

La relazione accorda le relazioni che implementano le seguenti relationship: ..... FFisac .....

<b>10 Relazione</b>	<u>Cliente</u> ..... (nome)	Derivante da: <b>entità</b>   <b>relationship</b> (cerchiare)
Attributi	<u>id</u>   indirizzo   nome   cognome	
Domini	integer   Indirizzo   stringM   StringM	

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):

seriale: i valori di id solo gen. autom. dal DBMS

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

## • Prenotazione

enupla : ( $\text{hotel} == \text{NULL} \wedge \text{in} == \text{NULL} \wedge \text{out} == \text{NULL} \wedge \text{ristanze} == \text{NULL}$ )  
V

( $\text{hotel} \neq \text{NULL} \wedge \text{in} \neq \text{NULL} \wedge \text{out} \neq \text{NULL} \wedge \text{ristanze} \neq \text{NULL}$ )

enupla :  $\text{in} \neq \text{NULL} \rightarrow \text{out} > \text{in}$  [V. prethotel.date]

enupla :  $\text{in} \neq \text{NULL} \rightarrow \text{in} \geq \text{ristante}$  [V. Prenotazione.checkIn]

### 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.

[V. Volo. maiOverbooking]

1) inserimento o modifica di una ennupla nella relaz. volPren

DB.postiDisponibili(v: string, d: date, i: datetime): integer

algoritmo:

- 1  $Q \leftarrow$  risultato della query seguente
- 2 select vel.posti - x.postiPrenotati as postiliberi  
from Velivolo vel, Volo v,  
(select sum(vp.pax) as postiPrenotati  
from volopren vp, Prenotazione p  
where vp.prenotazione = p.id  
and vp.velo = :v and vp.data = :d  
and vp.istante <= :i  
) x  
where v.codice = :v and v.velivolo = vel.codice
- 3 if  $Q == \text{NULL}$  then: genera errore
- 4 else: return il valore di Q

### Trigger

Operazione: inserimento di una ennupla nella relaz. volPren

Istante di invocazione: dopo l'operazione intercettata

Funzione:

- 1 new  $\leftarrow$  ennupla dopo la modifica

[continua alla pagina seguente]

## Risposta alla Domanda 7 (segue)

2 `isError ← exist (`

```
select *
  From Volo v,
  Where v.codice = new.codice
    and DB.postiDisponibili(v.codice, new.data,
      current_timestamp) < 0
  )
```

3 if `isError` then: blocca l'operazione

4 else permetti l'operazione

## [V. Prenotazione.date]

1) inserimento o modifica di una ennupla nella relazione `VoloPreh`

## Trigger

Operazione: inserimento di una ennupla nella relazione `VoloPreh`

Istante di invocazione: prima dell'operazione intercettata

Funzione:

1 `new ← l'ennupla inserita;`

2 `isError ←`

`exist (`

`select *`

`from Volo v`

`where v.codice = new.volo`

`and current_timestamp ≥ (new.data + v.oraPartenza)`

3 if `isError` then blocca l'operazione;

4 else permetti l'operazione;

## Risposta alla Domanda 7 (segue)

2 `isError ← exist (`

```
select *
  From Volo v,
  Where v.codice = new.codice
    and DB.postiDisponibili(v.codice, new.data,
      current_timestamp) < 0
  )
```

3 if `isError` then: blocca l'operazione

4 else permetti l'operazione

## [V. Prenotazione.date]

1) inserimento o modifica di una tupla nella relazione `VoloPreh`

## Trigger

Operazione: inserimento di una tupla nella relazione `VoloPreh`

Istante di invocazione: prima dell'operazione intercettata

Funzione:

1 `new ← l'entità inserita;`

2 `isError ←`

`exist (`

`select *`

`from Volo v`

`where v.codice = new.volo`

`and current_timestamp ≥ (new.data + v.oraPartenza)`

3 if `isError` then blocca l'operazione;

4 else permetti l'operazione;

Risposta alla Domanda 7 (segue)

[V. Volo. prezzoBase]

1) Inserim. o modifica di una ennupla nella relaz. Volo, in particolare se si dovesse modificare l'aeroporto di partenza o quello di arrivo

DB. prezzoBase(v: String): Money GEZ

1 Q ← result della seguenti query

2 select [(v.miglia \* vel.costoMiglio) + ap.tassaDec +  
+ aa.tassaArr] / vel.posti \* (1 + :RICARICO) as prezzo

from Volo v, Velivolo vel, Aeroporto ap, Aeroporto aa

where v.aeroplano = ap.codice

and v.aeroArr = aa.codice

and v.velivolo = vel.codice

and v.codice = :v

3 if Q == NULL then : genera errore

4 else return il valore di Q

## Trigger

Operazioni: inserimento di una ennupla nella relazione Volo

Istante di invocazione: dopo l'operazione intercettata

Funzione:

1 new ← ennupla inserita o risultato della modifica;

2 Esegui il seguente comando SQL

update Volo

set prezzoBase = DB. prezzoBase(new.codice)

where v.codice = new.codice

## Risposta alla Domanda 7 (segue)

Trigger:

Operazioni: modifica di una tupla nella relaz. Aeroporto

Istante: dopo l'operazione intercettata

Funzione:

1 old ← l' tupla oggetto della modifica;

2 new ← l' tupla che si vuole inserire

3 if new.tassadecollo ≠ old.tassadecollo

or

new.tassatterraggio ≠ old.tassatterraggio then

4 Esegui il comando SQL:

update Volo

set prezzoBase = AB.prezzoBase(codice)

where aeropart = new.codice or aerofatt = new.codice

if comando precedente ha restituito error then:

impedisca l'operazione

else permetti l'operazione

5 else: permetti l'operazione

**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

Specifiche UseCase: Posti

postiDisponibili(v:StringS, d:date, i:timestamp):integer

algoritmo:

1 Q ← risultato della query

2 select DB.postiDisponibili(:v, :d, :i) as postiliberi

3 if Q rappresenta un errore then: genera errore

4 else: return il valore di 'postiliberi' dell'unica ennupla in Q

Specifiche UseCase: Prezzi

prezzoBase(v:StringS):MoneyGEZ

algoritmo:

1 Q ← risultato della query

select v.prezzoBase as prezzoBase

from Volo v

where v.codice =:v

2 if Q rappresenta un errore then: inoltre errore

3 else: return il valore della colonna 'prezzoBase' di Q

Risposta alla Domanda 8 (segue)

`prezzoComplessivoSingoloPosto(v: StringS, d: date): MoneyGEZ`

algoritmo:

1  $Q \leftarrow$  risultato della query

select DB.prezzoComplessivoSingoloPosto(:v, :d) as prezzoComplessivo

2 if  $Q$  rappresenta un errore then: inoltre errore

3 else: return il valore della colonna 'prezzoComplessivo' di  $Q$

`prezzoComplessivo(v: StringS, d: date, n: IntegerGEZ): MoneyGEZ`

algoritmo:

1  $Q \leftarrow$  risultato della query

select (:n \* DB.prezzoComplessivoSingoloPosto(:v, :d))  
as prezzoComplessivo

2 if  $Q$  rappresenta un errore then: inoltre errore

3 else: return il valore della colonna 'prezzoComplessivo' di  $Q$

Specifico UseCase: Prenotazioni

`suggerisciHotel(c: integer, t: MoneyGEZ): Insieme(<id:integer, nome:StringM,  
cognome:StringM, indirizzo:Indirizzo, categoria:CFHOTEL,  
tariffa:MoneyGEZ, aeroporto*:char(3), distAereo*:IntegerGEZ>)`

algoritmo:

1  $H \leftarrow$  risultato della query SQL

## Risposta alla Domanda 8 (segue)

```

Select h.id, h.nome, h.indirizzo, h.categoria, h.tariffa,
    h.aeroporto, h.distanzaAero
From Hotel h, Hotel piu-vicino-max-cat
Where piu-vic-max-cat.citta = :c and piu-vic-max-cat.tariffa <= t
and piu-vic-max-cat.distCentro <= all(
    Select altro_t.distCentro
    From Hotel altro_t
    Where altro_t.citta = :c and altro_t.tariffa <= :t)
and piu-vic-max-cat.distCentro >= all(
    Select altro_t.categoria
    From Hotel altro_t
    Where altro_t.citta = :c and altro_t.tariffa <= :t)
and h.citta = :c
and (
    (h.categoria = piu-vic-max-cat.categoria
    and h.distCentro <= (1+0,1) * piu-vic-max-cat.distCentro)
    or
    (h.categoria > piu-vic-max-cat.categoria
    and h.distCentro <= (1+0,2) * piu-vic-max-cat.distCentro)
)

```

3 if H rappresenta un errore then: molta errore

4 else:  
if H e' NULL then: molta errore  
else: return H

Risposta alla Domanda 8 (segue)

## Specifiche Use Case: GestioneFF

`numeroMigliaVoli(p: integer): IntegerGZ`

1  $Q \leftarrow$  risultato della query

select DB.numMigliaVoli(:p) as numeroMiglia

2 if  $Q$  è vuoto then: return errore

3 else: return il valore di 'numeroMiglia' di  $Q$

`numMiglia(p: integer): Integer GZ`

1  $Q \leftarrow$  risultato della query

select DB.numMigliaVoli(:p) as numeroMiglia

2 if  $Q$  è vuoto then: return errore

3 else: return il valore di  $Q$

`numeroMigliaFF(F: integer, mom: timestamp): IntegerGZ`

1  $Q \leftarrow$  risultato della query

select (sum(DB.numMiglia(:p))) as numeroMiglia

from Prenotazione p, FF f

where f.cliente = :f and p.cliente = f.cliente

and p.istante >= f.affiliazione and p.istante < :mom

2 if  $Q$  è vuoto then: return errore

3 else: return il valore di  $Q$

Risposta alla Domanda 8 (segue)

# FUNZIONI:

DB. prezzoComplessivoSingoloPosto (vel: string, d: date): MoneyGEZ

1 Q  $\leftarrow$  risultato della query seguente ottenuto sostituendo a ':v' e ':d'  
il valore degli omonimi parametri attuali:

select v.prezzoBase •

(case when x.diff > 0

then power(1 - 0.02, x.diff)

else power(1 + 0.02, x.diff) end) as prezzoComplessivo

from Volo v, ( select (vel.pax / 2) +

- DB.postiDisponibili(:v, :d, :current\_time) as diff

from Velivolo vel, Volo v

where v.codice = :v

and v.velivolo = vel.codice ) x

where v.codice = :v

2 if Q è l'insieme vuoto then: genera errore

3 else: return il valore della colonna 'prezzoComplessivo' di Q

DB. numeroMigliaVoli (p: integer): IntegerGEZ

1 Q  $\leftarrow$  risultato della query

select sum(v.miglia \* vp.pax) as numeroMiglia

from Volo v, volopren vp, Prenotazione p

where vp.volo = v.codice and vp.prenotazione = p.id

and p.id =: p

2 if Q è vuoto then: return errore

3 else: return il valore di 'numeroMiglia' di Q

Risposta alla Domanda 8 (segue)

DB.amplificazione(p:integer): IntegerGEZ

1 Q ← risultato della query

select (case

when (h.categoria is NULL) then 1  
when (h.categoria <= 4) then 2  
else 3 end) as fattore

from Prenotazione p left outer join Hotel h on p.hotel = h.id  
where p.id =: p

2 if Q è vuoto then: return errore

3 else: return il valore di 'fattore' di Q

DB.numMiglia(p:integer): IntegerGEZ

1 Q ← risultato della query

select DB.numMigliaVol(:p) \* DB.amplificazione(:p)  
as migliaTotali

2 if Q è vuoto then: return errore

3 else: return il valore di 'migliaTotali' di Q