

Esame Es.20240214 – Prova scritta del 14 Febbraio 2024

Si vuole progettare e realizzare *Deli*, un sistema di gestione per un sito internet di *food delivery*. Per mezzo del sistema, si vogliono gestire gli ordini dei clienti presso i ristoranti e le consegne da parte dei fattorini.

Deli è direttamente utilizzato dai clienti, dai ristoranti, e dai fattorini. I clienti ordinano cibo dai ristoranti; i ristoranti definiscono le pietanze che possono essere ordinate; i fattorini prendono in carico gli ordini da consegnare ed effettuano fisicamente le consegne.

Dei clienti è di interesse conoscere il nome, il cognome, l'indirizzo e-mail (che funge da nome utente); inoltre, un cliente può registrare degli indirizzi di consegna nel sistema.

Di ogni fattorino interessa, oltre al nome, il cognome e l'indirizzo e-mail (che funge da nome utente), il volume massimo di merce che possono trasportare in un singolo viaggio (in litri).

Dei ristoranti sono di interesse il nome, l'indirizzo email (che funge da nome utente), la partita IVA, l'indirizzo, le cucine che offre (pizza, sushi, messicana, ecc.), l'importo minimo per un ordine, il costo di consegna (questi ultimi in Euro), i giorni e gli orari di apertura. L'offerta di un ristorante è organizzata per categorie. Ogni categoria ha un nome e contiene uno o più prodotti, dei quali interessa il nome e il prezzo. Un prodotto può essere una singola pietanza oppure un menù, il quale raggruppa più pietanze, ognuna con una certa quantità. Di ogni pietanza interessa conoscere l'ingombro, misurato in litri, quando è confezionato per la spedizione; l'ingombro associato ad un menù, invece, è pari alla somma degli ingombri delle singole pietanze che lo compongono.

I clienti hanno la possibilità di usare il sistema per creare un ordine presso un certo ristorante. Di ogni ordine interessa l'istante di creazione, i prodotti relativi (ognuno con una quantità associata), l'istante richiesto per la consegna, il metodo di pagamento ("contanti alla consegna" o "carta al momento dell'ordine"), eventualmente un messaggio per il ristorante e l'indirizzo di consegna (da scegliere tra quelli registrati dall'utente ordinante). L'istante richiesto per la consegna di un ordine deve essere compatibile con gli orari di apertura del ristorante.

Un ordine appena creato è nello stato *Pendente*. Un ordine pendente può essere accettato oppure rifiutato dal ristorante. In caso di accettazione, il management del ristorante definisce per l'ordine un istante di consegna stimato, che può essere diverso da quello richiesto dal cliente. Quando il ristorante inizia effettivamente a lavorare sull'ordine, questo entra in *Lavorazione*. Un ordine in lavorazione, dopo un certo tempo, sarà pronto per il ritiro da parte di un fattorino. Il cliente ha facoltà di annullare un ordine solo se questo è ancora pendente o, al massimo, accettato, mentre non può annullarlo se questo è già in lavorazione oppure in uno stato successivo.

I fattorini utilizzano il sistema per prendere in carico ordini presso uno o più ristoranti, diventando dunque responsabili per la loro consegna ai clienti. Affinché un ordine possa essere preso in carico da un fattorino, l'ordine deve essere in uno degli stati accettato, in lavorazione oppure già pronto per il ritiro. È di interesse conoscere l'istante in cui l'ordine viene preso in carico da un fattorino.

Quando un fattorino prende in carico un ordine, lo aggiunge ad un *viaggio*, che rappresenta un insieme di ordini (potenzialmente da ristoranti diversi) di cui diventa il responsabile della consegna, appunto, in un singolo viaggio. Quando il fattorino prende fisicamente possesso (presso un ristorante) delle pietanze relative ad un ordine da egli preso in carico, l'ordine entra nello stato in consegna e viene annotato con l'istante della presa di possesso da parte del fattorino. Una volta consegnato

al cliente finale, l'ordine passa nello stato *consegnato*. Degli ordini consegnati interessa conoscere l'istante di consegna effettivo.

I clienti possono lasciare delle recensioni ai ristoranti dai quali hanno ordinato. Una recensione è composta da un voto tra 1 e 10 ed un testo (opzionale).

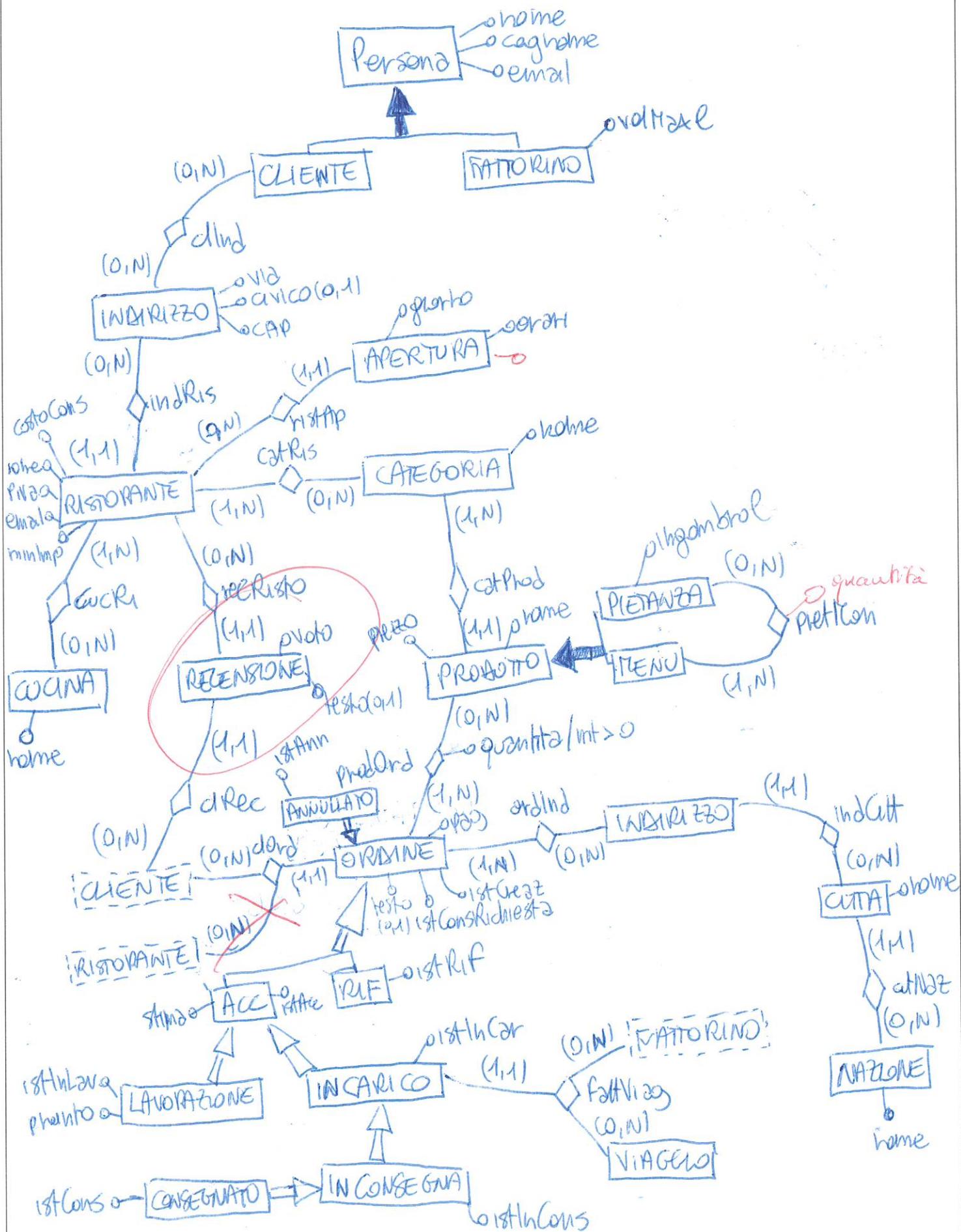
Il sistema deve offrire delle funzionalità specifiche alle diverse tipologie di utenti. In particolare, come descritto diffusamente sopra, i responsabili di un ristorante devono poter definire e modificare le loro offerte, accettare e rifiutare ordini; i clienti devono potersi iscrivere, aggiungere e rimuovere i loro indirizzi di consegna, ed effettuare e monitorare lo stato dei loro ordini; i fattorini devono potersi iscrivere, prendere in carico ordini da consegnare ai clienti, e comunicare le avvenute consegne.

Inoltre:

1. Il management di *Deli*, per monitorare la qualità del servizio offerto dalla rete dei fattorini, deve poter calcolare la lista di tutti i fattorini in servizio in un dato periodo, ognuno dei quali associato alla percentuale di ordini che ha consegnato (in quel periodo) entro l'orario stimato dal ristorante.
2. I fattorini devono poter prendere in carico un ordine ed associarlo ad un viaggio. Poiché ogni fattorino può trasportare un certo volume massimo di pietanze in un solo viaggio, il sistema deve impedire che un fattorino componga un viaggio che contenga un insieme di ordini il cui ingombro complessivo sia maggiore del volume massimo da egli trasportabile.

Risposta alla Domanda 2 (segue)

ER IN BELA



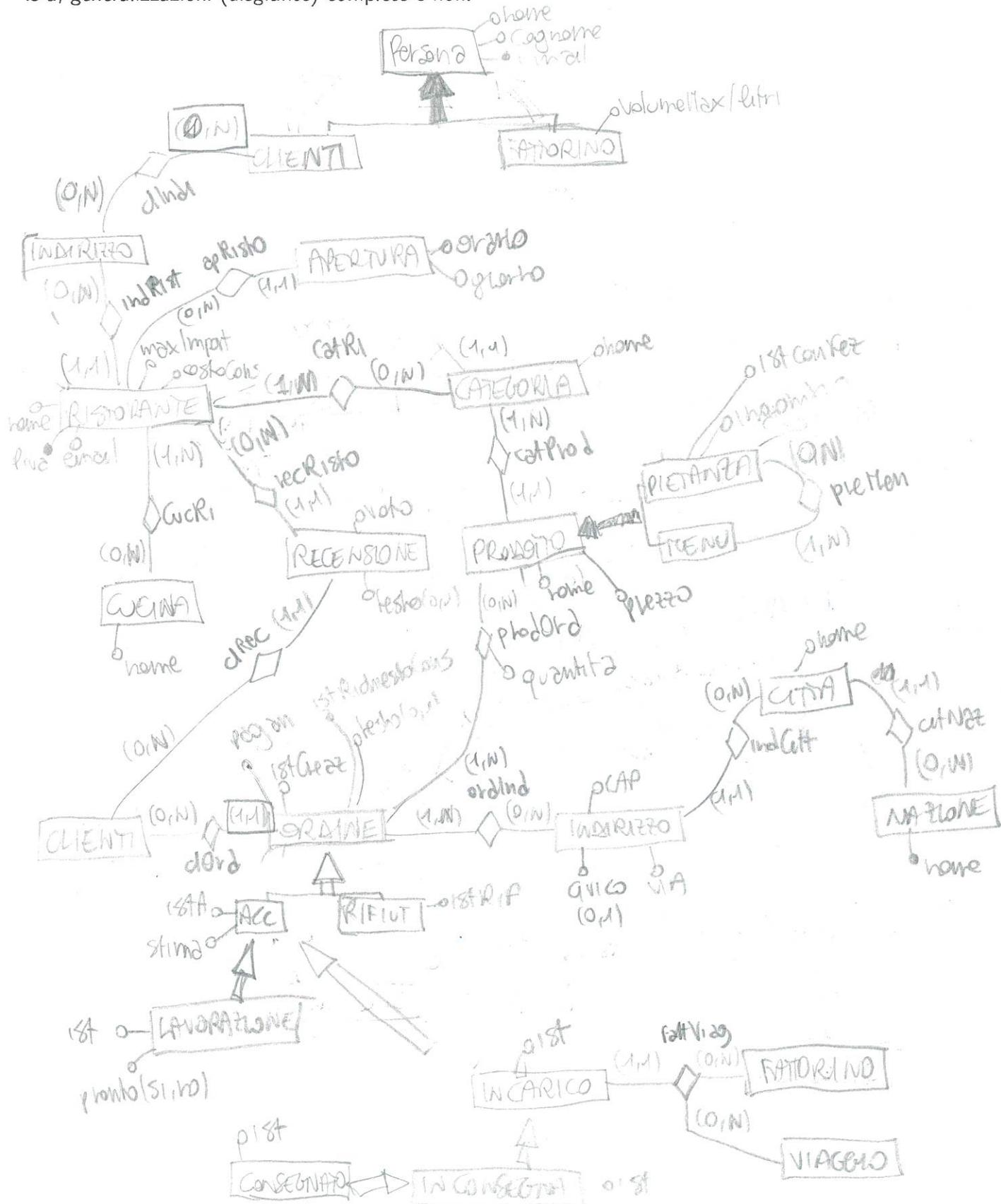
Domanda 2 (45 minuti; 75 minuti al massimo) Proseguire la fase di Analisi Concettuale dei requisiti, producendo un diagramma ER concettuale per l'applicazione, il dizionario dei dati ed eventuali vincoli esterni.

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

Diagramma ER

[GIRARE RAG PER BELLA COPIA]

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



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

<input type="checkbox"/> Tipo: Entità Relationship (cerchiare)		
Nome: <i>Persona</i>		
attributo	dominio	moltep. (*)
<i>nome</i>	<i>str</i>	
<i>Cognome</i>	<i>str</i>	
<i>email</i>	<i>Email</i>	

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

Vincoli:

<input type="checkbox"/> Tipo: Entità Relationship (cerchiare)		
Nome: <i>fattorino</i>		
attributo	dominio	moltep. (*)
<i>VolumenM&L</i>	<i>reale > 0</i>	

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

Vincoli:

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

Nome: INDIRIZZO

attributo	dominio	moltep. (*)
CAP	str di 5 caratteri	
VIA	str	
CIVICO	int > 0	(0..1)

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

Vincoli:

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

Nome: CITA

attributo	dominio	moltep. (*)
name	str	

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

Vincoli:

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

Nome: NAZIONE

attributo	dominio	moltep. (*)
name	str	

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

Vincoli:

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

Nome: APERURA

attributo	dominio	moltep. (*)
giorno	Giorno	
orario	Periodo	

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

Vincoli:

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

Nome: Ristorante

attributo	dominio	moltep. (*)
costoLavoro	Denaro	
name	str	
email	Email	
piro	str	
importoTutto	Denaro	

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

Vincoli:

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

Nome: Categorie

attributo	dominio	moltep. (*)
name	str	

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

Vincoli:

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

Nome: PROMOZIONE

attributo	dominio	moltep. (*)
name	str	

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

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

Nome: PIETANZA

attributo	dominio	moltep. (*)
ingombro	reale>0	

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

Vincoli:

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

Nome: CUCINA

attributo	dominio	moltep. (*)
<u>Tipo</u>	<u>str</u>	

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

Vincoli:

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

Nome: RECENSIONE

attributo	dominio	moltep. (*)
<u>voto</u>	<u>int[1,10]</u>	
<u>testo</u>	<u>str</u>	<u>(0,1)</u>

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

Vincoli:

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

Nome: ORDINE

attributo	dominio	moltep. (*)
<u>testo</u>	<u>str</u>	<u>(0,1)</u>
<u>1st Creazione</u>	<u>dataora</u>	
<u>1st Cons/Richiesto</u>	<u>dataora</u>	
<u>metPagam</u>	<u>Pagamento</u>	

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

Vincoli:

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

Nome: ACQUISITATO

attributo	dominio	moltep. (*)
<u>1st Acc</u>	<u>dataora</u>	
<u>stimato</u>	<u>dataora</u>	

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

Vincoli:

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

Nome: RIFIUTATO

attributo	dominio	moltep. (*)
<i>istRIF</i>	<i>dataora</i>	

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

Vincoli:

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

Nome: LAVORAZIONE

attributo	dominio	moltep. (*)
<i>phonto</i>	<i>boolean</i>	
<i>istLavora</i>	<i>dataora</i>	
<i>istPhonto</i>	<i>dataora</i>	

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

Vincoli:

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

Nome: IN CARICO

attributo	dominio	moltep. (*)
<i>istInCar</i>	<i>dataora</i>	

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

Vincoli:

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

Nome: IN CONSEGNA

attributo	dominio	moltep. (*)
<i>istInCons</i>	<i>dataora</i>	

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

Vincoli:

<u>Entità Consegnata</u>	
attr	dominio
<i>istCons</i>	<i>dataora</i>

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

Domino Giorno: $\{\text{'Lun'}, \text{'Mar'}, \dots, \text{'Dom'}\}$

Domino Periodo:

da: datagra

a: datagra

$\forall p, x, y \text{ Periodo}(p) \wedge \text{da}(p, x) \wedge \text{a}(p, y) \rightarrow x \leq y$

Domino Denaro:

importo: stringa di 3 caratteri

valuta: reale > 0

Domino Email: str secondo standard

Domino Pagamento: $\{\text{'contanti consegna'}, \text{'carta ordine'}\}$

oggetto-ing-Menu (p, Prezzi(0, N)): reale ≥ 0

pfe: nessuno

post: No side-effect

Val. ritorno

$I = \{(m, i) \mid \text{Menu}(m) \wedge \text{prezzo}(p, m) \wedge \text{ingombro}(i, p)\}$

$$\text{result} = \sum_{(m, i) \in I} i$$

[V. Ristorante, ggChiusuraOrdine]

$\forall r, u, o, i, a$

Ristorante(r) \wedge Utente(u) \wedge Ordine(o) \wedge istCreat(i, o) \wedge ord(o, o, r) \wedge

ristApp(a, r) \wedge orari(a, per) \wedge da(per, x) \wedge a(per, y)

$$\rightarrow x \leq i \wedge i \leq y$$

Risposta alla Domanda 2 (segue)

[IstanteAnnullato]

$\forall o \text{ Annullato}(o) \Leftrightarrow \neg \text{Lavorazione}(o) \wedge \neg \text{Rifiutato}(o)$

[Ordine.istanti]

$\forall o, i_c, i_r \text{ Ordine}(o) \wedge \text{istCreate}(i_c, o) \wedge \neg \text{ConsRichi}(i_r) \rightarrow i_c < i_r$

[Acc.ordine]

$\forall o, i_c, i_d \text{ Accettato}(o) \wedge \text{istCreate}(i_c, o) \wedge \text{istAcc}(i_d, o) \rightarrow i_c > i_d$

[Lavorazione.ordine]

$\forall o, i_L, i_d \text{ Lavorazione}(o) \wedge \text{istInLaw}(i_L, o) \wedge \text{istAcc}(i_d, o) \rightarrow i_L < i_d$

[InCar.ordine]

$\forall o : \text{InCarico}(o) \Leftrightarrow \text{Accettazione}(o) \vee \text{Lavorazione}(o) \vee \text{proto}(o, \text{true})$

[InCons.ordine]

$\forall o, i_C, i_R \text{ InConsegna}(o) \wedge \text{istInCons}(i_C, o) \wedge \text{istInCar}(i_R, o) \rightarrow i_C > i_R$

[Consegnato.ord]

$\forall o, i_C, i_C \text{ Consegnato}(o) \wedge \text{istCons}(i_C, o) \wedge \text{istInCon}(i_C, o) \rightarrow i_C < i_{CC}$

[Ordine.diss] ? No

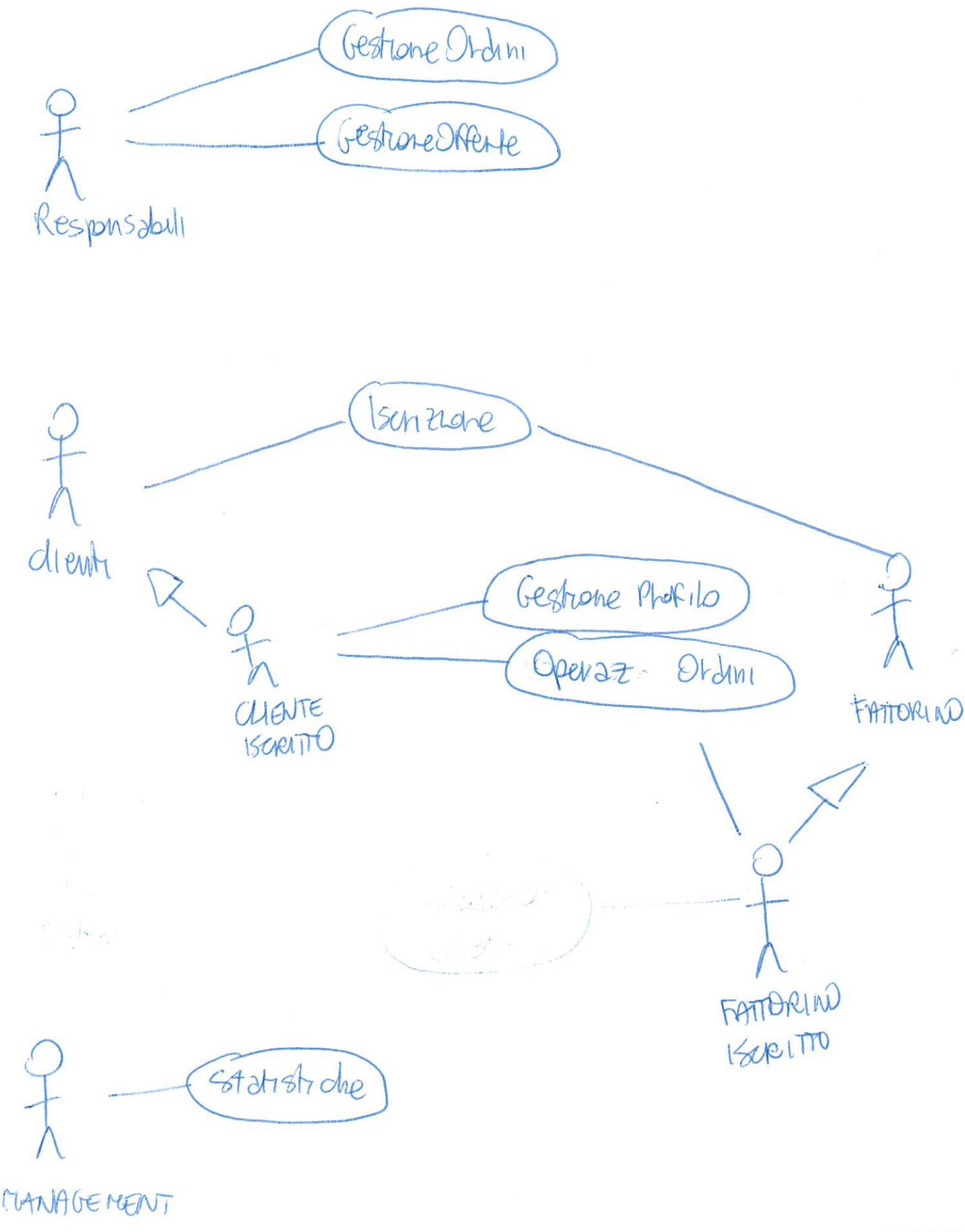
$\forall o, o', u, i, i' \text{ Ordine}(o) \wedge \text{Ordine}(o') \wedge \text{dOrd}(u, o) \wedge \text{dOrd}(u, o')$

$\wedge o \neq o' \wedge \neg \text{istCreate}(i, o) \wedge \neg \text{istCreate}(i', o') \rightarrow \exists t \text{ data} \forall (t) \wedge$
 $[i \leq t \wedge (\forall \text{istCons}(F, o) \wedge t \leq F) \wedge$
 $i' \leq t \wedge (\forall \text{istCons}(F', o') \wedge t \leq F')]$

Mancano alcuni vincoli

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 Ordini (nome use-case)

Operazioni dello use-case:

accetta(O : Ordine)

rifiuta(O : Ordine)

2 Specifica use-case: Gestione Profilo (nome use-case)

Operazioni dello use-case:

aggiungiIndirizzo(i : Indirizzo($O; N$))

rimuoviIndirizzo(i : Indirizzo(O, N))

3 Specifica use-case: Iscrizione (nome use-case)

Operazioni dello use-case:

iscriviCliente(no : str, co : str, e : Email): Cliente

iscriviFattorino(no : str, co : str, e : Email, v : reale > 0): Fattorino

4 Specifica use-case: ... Operazioni Ordini (nome use-case)

Operazioni dello use-case:

creaOrdine(texto: str(0,1), d: dataora, d': dataora, p: Pagamento, pr: Prodotto(1,N),
mantieneOrdine(o: Ordine) q: int ≥ 0, i: Indirizzo,
r: ristorante): Ordine

5 Specifica use-case: ... Operaz. Ordini (nome use-case)

Operazioni dello use-case:

funzFattorini(ord: Ordine, v: Viaggio): boolean

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

Operazioni dello use-case:

ListaFattorini(per: Periodo): [f: Fattorino, r: real](0,N)

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 assume lo stesso vocabolario definito alla Domanda 2.

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

Risposta

ListaFattorini (per: Periodo): (f : fattorino, r : intervallo) (0, N)

pre: per.da < per.a

post:

$$\text{fatt-perc} = \{ (\text{fatt}, \text{perc}) \mid \text{Fattorino}(\text{fatt}) \wedge$$

$$n = |\{ \text{ord} \mid \text{Ordine}(\text{ord}) \wedge \exists \text{ viag, ic}$$

$$\text{fattViagg}(\text{fatt}, \text{viagg}, \text{ord}) \wedge \text{Consegnato}(\text{ord}) \wedge$$

$$\text{InCarico}(\text{ord}) \wedge \text{istIn}(\text{ord}, \text{ic}, \text{ord}) \wedge$$

$$\text{per.da} \leq \text{ic} \wedge \text{per.a} \geq \text{ic} \}| \wedge$$

$$\text{in-time} = |\{ \text{ord} \mid \text{Ordine}(\text{ord}) \wedge \exists \text{ viagg, istm, ic}$$

$$\text{fattViagg}(\text{fatt}, \text{viagg}, \text{ord}) \wedge \text{Ave}(\text{ord}) \wedge$$

$$\text{istStimato}(\text{ord}, \text{istm}) \wedge \text{Consegnato}(\text{ord}) \wedge$$

$$\text{istCons}(\text{ic}, \text{ord}) \wedge \text{ic} \leq \text{istm} \wedge$$

$$\text{per.da} \leq \text{ic} \wedge \text{per.a} \geq \text{ic} \}| \wedge$$

$$1 \quad \text{perc} = (\text{in-time} / n) * 100 \}$$

$$\text{result} = \text{fatt-perc}$$

Risposta alla Domanda 5 (segue)

`funtFattor(ORD Ordine, v Viaggio, fatt Fattorino): boolean`

PtE:

$$\text{fatt_new} = \{ (ps) \mid \begin{array}{l} \text{fatt.fattorino} \in \text{Fattorino}, \text{fatt.fattorino}(\text{fatt}, v, \text{ord}) \wedge \\ \text{prodOrd}(pt, \text{ord}) \wedge \text{quantita}(\text{ord}, pr, qual) \wedge \\ [\text{Pietanza}(pr) \wedge \text{ingombro}(pr, ing)] \wedge \\ ps = \text{ing} + \text{qua} \end{array} \} \vee$$

$$[\text{Item}(pr) \wedge \text{tot_ing_men} = \text{get_ing_men}(pt) \wedge$$

and $ps = \text{tot_ing_men} + \text{qua}$]

`peso_ord = { (ps, fatt_max) \mid \text{prodFattor prodOrd}(pr, ord) \wedge`

`quantita(ORD, pr, qual) \wedge`

`[Pietanza(pr) \wedge \text{ingombro}(pr, ing)] \wedge`

`ps = ing + qua] \vee`

`[\text{Item}(pr) \wedge \text{tot_ing_men} = \text{get_ing_men}(pt) \wedge`

`ps = tot_ing_men + qua] \wedge`

`Fattorino(fatt) \wedge \text{valMax}(\text{fatt}, \text{fatt_max}) \}`

Mancava la somma

if $\text{fatt_new} + \text{peso_ord}, ps > \text{fatt_max}$:

ritorna

~~ritorna~~

post: inserire in `FattViagg` l'ennupla(`fatt, v, ORD`)

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 type Giorno as enum('Lun', ..., 'Dom')

create domain Periodo as (da timestamp, a timestamp)

create domain StringS as varchar(50)

| "StringM" " " (200);

| "StringL" " " (50);

create domain Email as stringM cheche (isValidEmail(value))

create type Pagoamento as enum('Contanti', 'CartaOrdine')

create domain IntFZ as integer check (value > 0)

| "IntGEZ" " " check (val > 0)

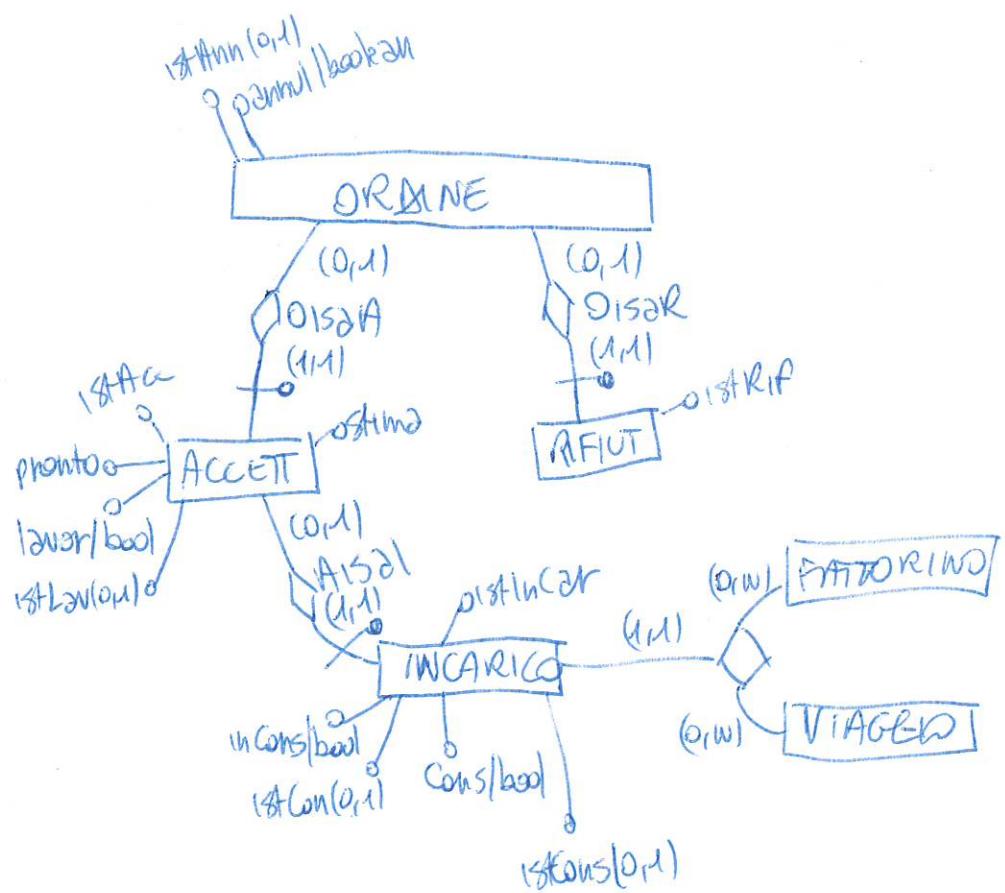
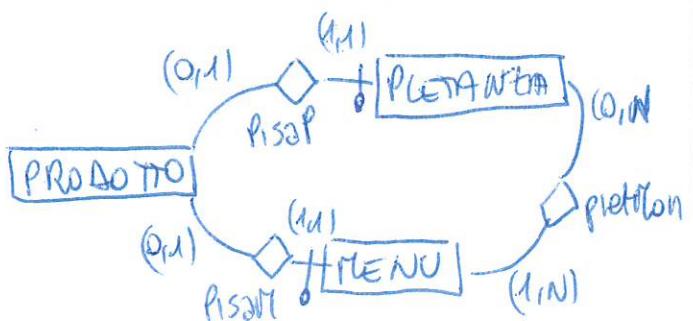
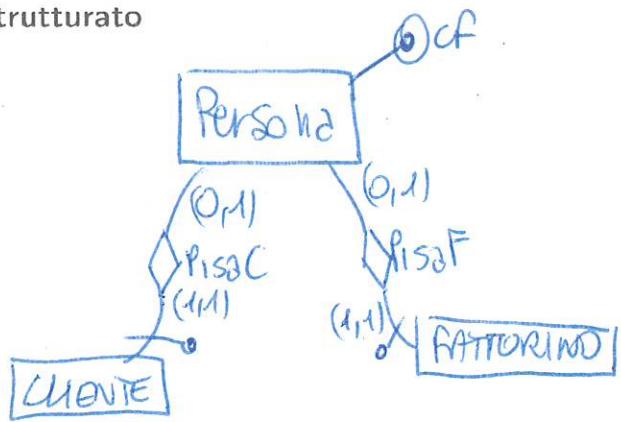
| "RealFZ" as real check (Val > 0)

| "RealGEZ" " " check (value > 0)

create domain CodFisc as char(16) check (isValidCF(value))

create domain Voto as integer check (val ≥ 1 AND val ≤ 10)

Diagramma ER ristrutturato



Breve descrizione delle scelte effettuate durante la ristrutturazione

sost di ISB con token in Person

a a a a . . . Prodotto

a a a a . . . in Ordine per Ace e R.F.

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)

[N. Pers. dissjCompl]

$$\forall p \text{ Persona}(p) \rightarrow [\exists c \in \text{PisaC}(c, p)] \rightarrow [\exists f \in \text{PisaF}(p, f)] \quad \text{dss}$$

$$1[\exists c \in \text{PisaC}(c, p)] \vee [\exists f \in \text{PisaF}(p, f)] \quad \text{com}$$

[N. prod. dissjCompl]

$$\forall p \text{ Prodotto}(p) \rightarrow [\exists pp \text{ PisaP}(p, pp)] \rightarrow [\exists m \text{ PisaM}(m, p)] \quad \text{dss}$$

$$1[\exists pp \text{ PisaP}(p, pp)] \vee [\exists m \text{ PisaM}(m, p)] \quad \text{com}$$

Nei Trigger divide dissj e Compl

Risposta alla Domanda 6 (segue)

[V. Ord. annullato]

$\forall o \quad \text{Ordine}(o) \rightarrow \text{annullato}(o, \text{true}) \leftrightarrow \nexists r \text{ ha } \text{OraF}(r, o) \wedge \text{OraA}(o, a) \wedge \text{lavorazione}(o, \text{true})$

[V. Acc. Lav]

$\forall a \quad \text{Accettato}(a) \rightarrow \text{lavoro}(a, \text{true}) \leftrightarrow \exists i \text{ istLav}(i, a)$

[V. Incar. cont]

$\forall c \quad \text{Incarico}(c) \rightarrow \text{inCons}(c, \text{true}) \leftrightarrow \exists i \text{ istInCon}(i, c)$

[V. Cons. date]

$\forall c \quad \text{Incarico}(c) \wedge \text{inCon}(c, \text{true}) \rightarrow \text{consegnato}(c, \text{true}) \leftrightarrow \exists i \text{ istCons}(i, c)$

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	Person (nome)	Derivante da:	entità	relationship (cerchiare)
-------------	--------	--------------	---------------	--------	--------------------------

Attributi	CF	name	Cognome	Email				
Domini	CodFisc	Strs	Strs	Email				

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:

2 Relazione	CLIENTE (nome)	Derivante da:	entità	relationship (cerchiare)
-------------	---------	--------------	---------------	--------	--------------------------

Attributi	persona							
Domini	CodFis							

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):

PK: persona refer Persona(cf)

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

3 Relazione	FATTORINO (nome)	Derivante da:	entità	relationship (cerchiare)
-------------	-----------	--------------	---------------	--------	--------------------------

Attributi	persona							
Domini	CodFis							

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):

PK: persona refer Persona(cf)

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

4 Relazione	INDRIZZO (nome)	Derivante da:	entità	relationship (cerchiare)
-------------	----------	--------------	---------------	--------	--------------------------

Attributi	VIA	UNICO*	CAP	città	id			
Domini	Strs	IntGz	char(5)	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):

PK: città refer Città(id)

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

5 Relazione	Città (nome)	Derivante da:	entità	relationship (cerchiare)
-------------	-------	--------------	---------------	--------	--------------------------

Attributi	name	nazione	id					
Domini	Strs	Strs	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):

PK: nazione refer Nat(name)

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

6 Relazione <u>client</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>cliente</u> <u>Indirizzo</u>	
Domini <u>CodFis</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):

PK: indir refer Indirizzo(id)

VK: cliente refer Persona(cf)

La relazione accorda le relazioni che implementano le seguenti relationship:

7 Relazione <u>APERTURA</u> ... (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>giorno</u> <u>offri</u> <u>RISTORANTE</u> <u>id</u>	
Domini <u>giorno</u> <u>perLavo</u> <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): seriale: id

PK: rsto refer Risto(id)

ennupla: per.da < per.a

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

8 Relazione <u>RISTORANTE</u> . (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>id</u> <u>P.via</u> <u>email</u> <u>minImp</u> <u>contoCon</u> <u>Indirizzo</u>	
Domini <u>integer</u> <u>strs</u> <u>Email</u> <u>Numero</u> <u>Numero</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): seriale: id

PK: Indir refer Indirizzo(id) Ind:id ⊆ CatRis(Ristorante)

Indus: id ⊆ CatRis(Ristorante)

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

9 Relazione <u>WCINA</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>home</u>	
Domini <u>strs</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:

10 Relazione <u>WCRI</u> (nome)	Derivante da: entità relationship (cerchiare)
Attributi <u>nsto</u> <u>wand</u>	
Domini <u>integer</u> <u>strings</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):

PK: nsto refer Risto(id)

VK: wana refer Wana(home)

La relazione accorda le relazioni che implementano le seguenti relationship:



11 Relazione RECENSIONE (nome)	Derivante da: entità relationship (cerchiare)
Attributi voto <u>id</u> testo* nota cliente	
Domini Voto integer string id Cognome	

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: id

PK: voto refer Ristorante(id)

PK: cliente refer Persona(cf)

La relazione accorda le relazioni che implementano le seguenti relationship: recristo, id Rec.

12 Relazione CATEGORIA (nome)	Derivante da: entità relationship (cerchiare)
Attributi nome <u>id</u> 	
Domini string 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): seriale: id

PK: id catProd (categorial)

PK: id ...

La relazione accorda le relazioni che implementano le seguenti relationship:

13 Relazione CatRIS (nome)	Derivante da: entità relationship (cerchiare)
Attributi Categ <u>Riso</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 ennupla, di dominio):

PK: categ refer Categorial(id)

PK: Riso refer Ristorante(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

14 Relazione PRODOTTO (nome)	Derivante da: entità relationship (cerchiare)
Attributi nome prezzo <u>id</u> Categ 	
Domini string integer id Categorial 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): seriale: id

PK: categ refer Categorial(id)

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

15 Relazione PIETANZA (nome)	Derivante da: entità relationship (cerchiare)
Attributi ingred <u>prodotto</u> 	
Domini RealGZ integer Prodotto 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):

PK: prodotto refer Prodotto(id)

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

16 Relazione <u>MOV</u>(nome)									Derivante da: entità relationship (cerchiare)
-------------------------------------	--	--	--	--	--	--	--	--	---

Attributi <u>prodotto</u>								
Domini <u>integer</u>								

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

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

PK: prodotto refer prodotto(id)

INC: prodotto \subseteq prodMov(prodanz) dove?

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

17 Relazione <u>prodOrd</u>(nome)									Derivante da: entità relationship (cerchiare)
--	--	--	--	--	--	--	--	--	---

Attributi <u>quant</u> <u>prodotto</u> <u>ordine</u>								
Domini <u>IntGz</u> <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):

PK: prod refer Prodotto(id)

PK: ordine refer Ordine(id)

La relazione accorda le relazioni che implementano le seguenti relationship:

18 Relazione <u>ORDINE</u>(nome)									Derivante da: entità relationship (cerchiare)
---------------------------------------	--	--	--	--	--	--	--	--	---

Attributi <u>id</u> <u>istAnn</u> <u>annuat</u> <u>testo</u> <u>isCreate</u> <u>isConRic</u> <u>paes</u>								
Domini <u>timestamp</u> <u>boolean</u> <u>string</u> <u>timestamp</u> <u>timestamp</u> <u>paes</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): setiale: id

INC: id \subseteq prodOrd(ORDINE) enup: istAnn \neq null \leftrightarrow annuat = true

INC: id \subseteq ordind(ORDINE) enup: istAnn \neq null \rightarrow istAnn $<$ isCreate enup: isCr $<$ isConRic

La relazione accorda le relazioni che implementano le seguenti relationship:

19 Relazione <u>ORDINE</u>(nome)									Derivante da: entità relationship (cerchiare)
---------------------------------------	--	--	--	--	--	--	--	--	---

Attributi <u>cliente</u> <u>ristorante</u>								
Domini <u>CostFis</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):

PK: cliente refer Personal(cf)

PK: ristorante refer Ristorante(id)

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

20 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):

CONTINUAR A MINUTE 33

La relazione accorda le relazioni che implementano le seguenti relationship:

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.]

- **ACCETT**(istAcc: timestamp, pronto: boolean, lavoro: boolean, ordine: integer, istLav*: timestamp) include relen AISAL
 Stima: timestamp
 PK: ordine refer Ordine(id)
- esempio: istLav ≠ null \rightarrow Lavato = TRUE
 enn: istLav ≠ null \rightarrow istLav < istAcc
- **RIFUT**(istRif: timestamp, ordine: integer) include relen AISAR
 PK: ordine refer Ordine(id)
- **InCarico**(istInCar: timestamp, istCons*: timestamp, istInCons: timestamp, inCons: boolean, consegnata: boolean, accettazione: integer, viaggio: integer, fatturato: CF)
 PK: accettaz refer Ordine(id)
 FK: viaggio refer Viaggio(id)
 FK: fatturato refer Personale(cf)
 include relen AISAL, fattViaggio
 esmp: consegnata = true \rightarrow istCons ≠ null
 ennmp: istCons ≠ null / istInCons ≠ null \rightarrow istCons < istInCons
 ennmp: inConsIncarico = true \rightarrow istConsIncarico ≠ null

Viaggio(id: integer)
 schede: id

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

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 ennupla); (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. Pers. diss]

Opraz: insert o modify in Cliente e Fatturato

Istante: before

Funz:

```
ISEROR = select exists (select *
                        from Cliente c, fatturato f
                        where c.persona = new.persona
                          and f.persona = new.persona)
```

if ISEROR: block operazione
else: permetti l'operazione

[V. Pers. comp]

Opraz: insert o modify in Persona

Istante: before

Funz:

```
ISEROR = select exists (select *
                        from Cliente c, fatturato f
                        where c.persona = new.persona or
or f.persona = new.persona)
```

if ISEROR: block operazione
else: permetti l'operazione

Risposta alla Domanda 7 (segue)

[V. Prod. dsj]

Operaz: insert o modify in Menu e Pretanza

Istante: before

Funz:

```
isElter = select exists(select *
                        from Menu m, Pretanza p
                        where m.prodotto = new.prodotto
                          and p.prodotto = new.prodotto)
```

if isElter: block

else: permetti l'operazione

[V. Prod. compl]

Operaz insert o modify in Prodotto

Istante: before

Funz:

```
isElter = select exists(select *
                        from Menu m, Pretanza p
                        where m.prodotto = new.prodotto or
                          and p.prodotto = new.prodotto)
```

if isElter: block

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

ListaFattorini (per Periodo): ($f:\text{CodFis}, r:\text{IntGZ}$)

tot_consegne = select fatt.persona, count(fatt.persona) as num
 from Fattorino fatt, Incarico inc, Viaggio v
 where inc.fattorino=fatt.persona and inc.viaggio=v.id
 and inc.consegnato=True and :per.da <= inc.istIncar
 and :per.a >= inc.istIncar
 group by fatt.persona

totcons-inttemp = select fatt.persona, count(fatt.persona) as num
 from Fattorino fatt, Incarico inc, Viaggio v, Accettaz acc
 where inc.fattorino=fatt.persona and inc.viaggio=v.id
 and inc.consegnato=True and :per.da <= inc.istIncar
 and :per.a >= inc.istIncar and
 acc.ordine=inc.accettazione and
 acc.stima>=inc.istCon
 group by fatt.persona

(gira pag)

Risposta alla Domanda 8 (segue)

result = select tot-cons-persona,

case

when tot-cons-in-tempo-number != 0

then tot-cons-number / tot.cons.intempo.number

else null

END AS division-result

from tot-consegne

inner join

tot-consegne-intempo on

tot-cons-persona = tot.cons.intempo-persona

funzione

select

