SAPIENZA
UNIVERSITÀ DI ROMA

Basi di Dati, Modulo 2

Sapienza Università di Roma
Facoltà di Ing. dell'Informazione, Informatica e Statistica
Laurea in Informatica
Prof. Toni Mancini, Prof. Federico Mari
http://tmancini.di.uniroma1.it
http://mari.di.uniroma1.it

Progetto 20060411 (P.20060411)

Travel to the Moon

Versione 2016-05-14

Indice

1 Introduzione

Si vuole sviluppare un sistema per la gestione di una agenzia turistica che organizza crociere.

2 Specifica dei Requisiti

I dati di interesse per il sistema sono le crociere offerte dall'agenzia con le relative prenotazioni e le destinazioni in catalogo.

Il sistema deve essere in grado di rappresentare le crociere offerte dall'agenzia, con codice, date di inizio e fine, e la nave utilizzata. Delle navi, che hanno un nome (ad es. "LoveBoat"), interessa il grado di comfort, espresso in un numero di stelle che può variare da 3 a 5, e il numero massimo di passeggeri che possono ospitare.

Ciascuna crociera consta di un itinerario caratterizzato da un nome (ad es. "Panorami d'Oriente") il quale prevede una sequenza –ordinata– di destinazioni. Di queste interessa il nome e il continente in cui si trovano. Gli itinerari fissano, oltre che l'ordine delle destinazioni da visitare, anche la relativa data ed ora di arrivo e di partenza. Dato che, in generale, un itinerario può essere previsto da più di una crociera, le date di arrivo e partenza relative ad una destinazione vengono espresse come differenze rispetto la data di inizio della crociera stessa (ad es., l'itinerario "Panorami d'Oriente" prevede di raggiungere la destinazione x alle 16:00 del quinto giorno di crociera, e di ripartire alle 12:00 del giorno successivo, il sesto).

Inoltre, le destinazioni sono caratterizzate da un insieme di posti da vedere durante eventuali escursioni organizzate. Questi ultimi sono caratterizzati dal nome, dalla descrizione, e dalla fascia oraria consigliata per le visite. Il sistema deve permettere di risalire ai posti da vedere in ogni singola destinazione.

L'agenzia classifica le crociere in *crociere di luna di miele* e *crociere per famiglia* (di queste ultime interessa conoscere se sono adatte o meno ai bambini), e le destinazioni in *romantiche* e *divertenti*. Si noti che possono esistere destinazioni che sono sia romantiche che divertenti. Per venire incontro alle nuove tendenze delle giovani coppie, le crociere di luna di miele vengono ulteriormente classificate in *tradizionali* e *alternative*: sono definite tradizionali quelle che prevedono un numero di destinazioni romantiche maggiore o uguale al numero di destinazioni divertenti, alternative le altre.

Infine, il sistema deve anche permettere di gestire le prenotazioni di crociere effettuate

dai clienti. In particolare, dei clienti interessa nome, cognome, età ed indirizzo, mentre delle prenotazioni interessa l'istante di prenotazione, la crociera ed il numero di posti prenotati.

Le funzionalità richieste al sistema sono le seguenti:

1. Dato un cliente che desidera prenotare un certo numero di posti per una crociera c, il personale dell'Ufficio Prenotazioni deve poter effettuare la relativa prenotazione. La richiesta di prenotazione deve essere rifiutata nel caso il numero di posti disponibili, all'istante corrente, per la crociera c non sia sufficiente.

2. Dato un insieme di clienti, l'Ufficio Marketing deve poter calcolare l'età media di quelli che hanno prenotato almeno una crociera che prevede una destinazione esotica (ovvero che si trova in un continente diverso dall'Europa).

3. Dato un insieme di destinazioni, l'Ufficio Marketing deve poter calcolare la percentuale di quelle *gettonate*. Una destinazione si dice gettonata se è stata raggiunta da almeno dieci crociere di luna di miele, oppure da almeno quindici crociere per famiglie nel corso degli ultimi due anni.

Parte A

Analisi Concettuale

A.1 Raffinamento dei Requisiti

A.1.1 Testo

Raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e producendo un elenco numerato di requisiti il meno ambiguo possibile.

A.1.2 Soluzione

1. Delle crociere offerte interessa:
   1.1. codice
   1.2. data di inizio
   1.3. data di fine
   1.4. nave utilizzata (Req. 2.)
   1.5. itinerario (Req. 3.).
   1.6. Le crociere si classificano in *crociere di luna di miele* e *crociere per famiglia*:
      1.6.1. Le crociere di luna di miele si classificano in:
         1.6.1.1. tradizionali, se il loro itinerario (Req. 1.5.) prevede un numero di destinazioni romantiche (Req. 4.4.) maggiore o uguale al numero di destinazioni divertenti (Req. 4.5.)
         1.6.1.2. alternative, se il loro itinerario (Req. 1.5.) prevede un numero di destinazioni romantiche (Req. 4.4.) strettamente minore al numero di destinazioni divertenti (Req. 4.5.).
      1.6.2. Delle crociere per famiglia interessa sapere:
         1.6.2.1. se sono adatte ai bambini.
2. Delle navi utilizzate per le crociere interessa:
   2.1. nome
   2.2. grado di comfort (intero tra 3 e 5)
   2.3. numero massimo di passeggeri.
3. Degli itinerari delle crociere interessa:
   3.1. nome
   3.2. sequenza destinazioni. Per ogni destinazione interessa:
      3.2.1. numero d'ordine nell'itinerario
      3.2.2. data di arrivo, in termini di numero di giorni dall'inizio della crociera
      3.2.3. ora di arrivo
      3.2.4. data di partenza, in termini di numero di giorni dall'inizio della crociera
      3.2.5. ora di partenza.
4. Delle destinazioni interessa:
   4.1. nome
   4.2. continente
   4.3. posti da vedere

4.4. se sono *romantiche*

4.5. se sono *divertenti*

4.6. se sono *gettonate*, ovvero sono state raggiunte da almeno dieci crociere di luna di miele oppure da almeno quindici crociere per famiglie nel corso degli ultimi due anni

4.7. se sono *esotiche*, ovvero si trovano in un continente diverso dall'Europa.

5. Dei posti da vedere interessa:

   5.1. nome

   5.2. descrizione

   5.3. fascia oraria consigliata per le visite.

6. Dei clienti interessa:

   6.1. nome

   6.2. cognome

   6.3. data di nascita

   6.4. indirizzo.

7. Di ogni prenotazione effettuata da un cliente per una crociera interessa:

   7.1. cliente che ha effettuato la prenotazione

   7.2. istante di prenotazione

   7.3. crociera prenotata

   7.4. numero di posti prenotati.

8. Le funzionalità richieste al sistema sono:

   8.1. Dato un cliente che desidera prenotare un certo numero di posti per una crociera c, il personale dell'Ufficio Prenotazioni deve poter effettuare la relativa prenotazione. La richiesta di prenotazione deve essere rifiutata nel caso il numero di posti disponibili, all'istante corrente, per la crociera c non sia sufficiente.

   8.2. Dato un insieme di clienti, l'Ufficio Marketing deve poter calcolare l'età media di quelli che hanno prenotato almeno una crociera che prevede una destinazione esotica (Req. 4.7.).

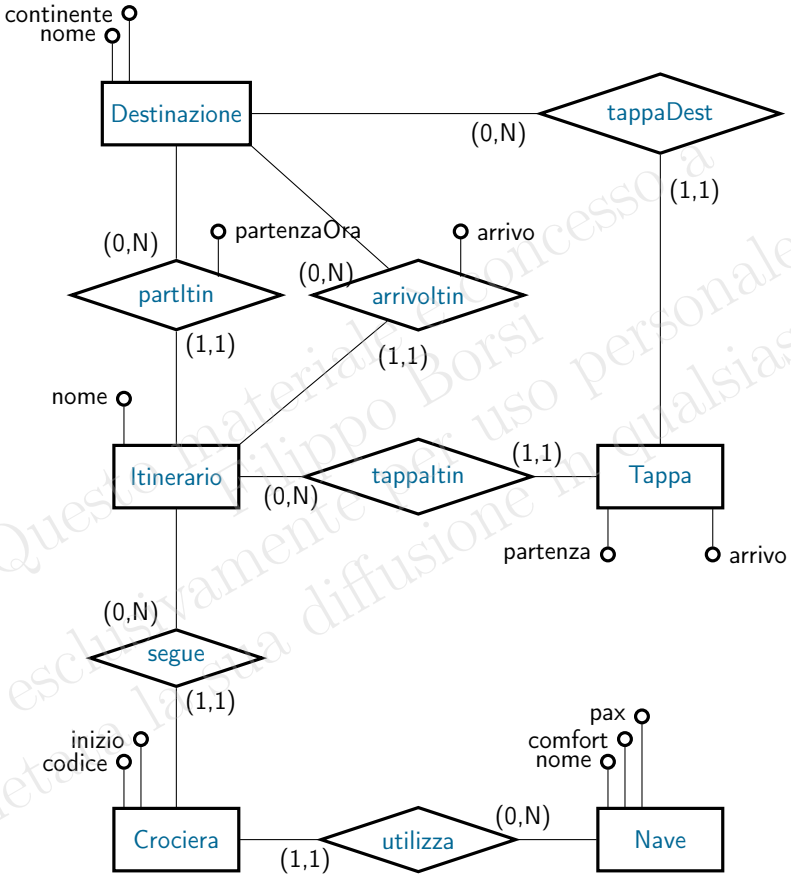
   8.3. Dato un insieme di destinazioni, l'Ufficio Marketing deve poter calcolare la percentuale di quelle gettonate (Req. 4.6.).

A.2 Diagramma ER

A.2.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti a partire dall'output della fase di raffinamento effettuata al passo A.1. In particolare, produrre il diagramma ER concettuale per l'applicazione ed il dizionario dei dati per modellare i seguenti requisiti: Req. 1.1., Req. 1.2., Req. 1.3., Req. 1.4., Req. 1.5., Req. 2., Req. 3., Req. 4.1., Req. 4.2.

A.2.2 Soluzione



Specifiche dei Dati

Entità Crociera

Ogni istanza di questa entità rappresenta una crociera (Req. 1.)

attributo	dominio	molteplicità	descrizione
codice	stringa		Il codice della crociera
inizio	data		La data di inizio della crociera

Entità Nave

Ogni istanza di questa entità rappresenta una nave (Req. 2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nave
comfort	[3,5]		Il grado di comfort della nave
pax	intero > 0		Il numero massimo di passeggeri per la nave

Entità Itinerario

Ogni istanza di questa entità rappresenta un'itinerario (Req. 3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'itinerario

Entità Destinazione

Ogni istanza di questa entità rappresenta una destinazione toccata da Itinerari (Req. 4.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della destinazione
continente	{E, AS, AMN, AMS, AF, O}		Il continente della destinazione (nell'ordine: Europa, Asia, America del Nord, America del Sud, Africa, Oceania)

Entità Tappa

Ogni istanza di questa entità rappresenta una Tappa toccata da un Itinerario (Req. 3.2.), escluse quelle di partenza e arrivo dell'Itinerario

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di arrivo alla Destinazione
partenza	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di ripartenza dalla Destinazione

Relationship utilizza

Ogni istanza di questa relationship lega una Crociera alla Nave utilizzata (Req. 1.4.)
Attributi: Nessuno

Relationship segue

Ogni istanza di questa relationship lega una Crociera al suo Itinerario (Req. 1.5.)
Attributi: Nessuno

Relationship tappaItin

Ogni istanza di questa relationship lega un Itinerario ad una sua Tappa (Req. 3.2.)
Attributi: Nessuno

Relationship tappaDest

Ogni istanza di questa relationship lega una Tappa di un Itinerario alla relativa Destinazione (Req. 3.2.)
Attributi: Nessuno

Relationship partItin

Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di partenza (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
partenzaOra	ora		L'ora di partenza della Crociera

Relationship arrivoItin

Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di arrivo (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (in termini di giorni dalla data di partenza dell'itinerario) e l'ora di arrivo della Crociera alla Tappa

Dominio DeltaDataOra

Il dominio è un record composto dai seguenti campi:

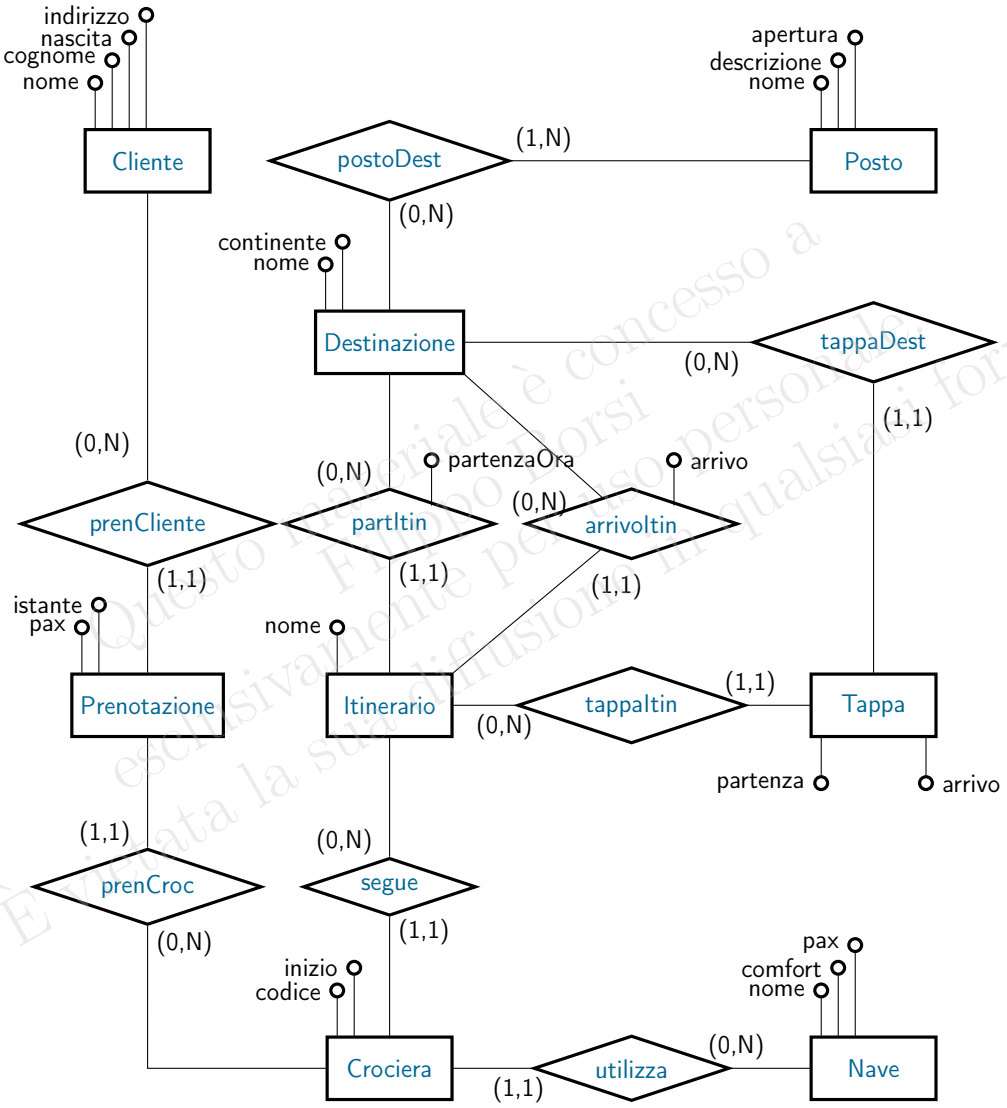
- giorni: intero > 0
- ora: ora

A.3 Estensione del Diagramma ER

A.3.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti estendendo il diagramma ER concettuale per l'applicazione ed il dizionario dei dati per modellare anche i seguenti requisiti: Req. 4.3., Req. 5., Req. 6., Req. 7.

A.3.2 Soluzione



Specifiche dei Dati

Entità Crociera

Ogni istanza di questa entità rappresenta una crociera (Req. 1.)

attributo	dominio	molteplicità	descrizione
codice	stringa		Il codice della crociera
inizio	data		La data di inizio della crociera

Entità Nave

Ogni istanza di questa entità rappresenta una nave (Req. 2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nave
comfort	[3,5]		Il grado di comfort della nave
pax	intero > 0		Il numero massimo di passeggeri per la nave

Entità Itinerario

Ogni istanza di questa entità rappresenta un'itinerario (Req. 3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'itinerario

Entità Destinazione

Ogni istanza di questa entità rappresenta una destinazione toccata da Itinerari (Req. 4.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della destinazione
continente	{E, AS, AMN, AMS, AF, O}		Il continente della destinazione (nell'ordine: Europa, Asia, America del Nord, America del Sud, Africa, Oceania)

Entità Tappa

Ogni istanza di questa entità rappresenta una Tappa toccata da un Itinerario (Req. 3.2.), escluse quelle di partenza e arrivo dell'Itinerario

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di arrivo alla Destinazione
partenza	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di ripartenza dalla Destinazione

Entità Posto

Ogni istanza di questa entità rappresenta un posto da vedere (Req. 5.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del posto da vedere
descrizione	stringa		La descrizione del posto da vedere
apertura	FasciaOraria		La fascia oraria di apertura del posto da vedere

Entità Cliente

Ogni istanza di questa entità rappresenta un cliente (Req. 6.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del cliente
cognome	stringa		Il cognome del cliente
nascita	data		La data di nascita del cliente
indirizzo	Indirizzo		L'indirizzo del cliente

Entità Prenotazione
Ogni istanza di questa entità rappresenta una prenotazione di una Crociera da parte di un Cliente (Req. 7.)

attributo	dominio	molteplicità	descrizione
pax	intero > 0		Il numero di posti della prenotazione
istante	dataora		L'instante nel quale è stata effettuata la prenotazione

Relationship utilizza
Ogni istanza di questa relationship lega una Crociera alla Nave utilizzata (Req. 1.4.)
Attributi: Nessuno

Relationship segue
Ogni istanza di questa relationship lega una Crociera al suo Itinerario (Req. 1.5.)
Attributi: Nessuno

Relationship tappaItin
Ogni istanza di questa relationship lega un Itinerario ad una sua Tappa (Req. 3.2.)
Attributi: Nessuno

Relationship tappaDest
Ogni istanza di questa relationship lega una Tappa di un Itinerario alla relativa Destinazione (Req. 3.2.)
Attributi: Nessuno

Relationship partItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di partenza (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
partenzaOra	ora		L'ora di partenza della Crociera

Relationship arrivoItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di arrivo (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (in termini di giorni dalla data di partenza dell'itinerario) e l'ora di arrivo della Crociera alla Tappa

Relationship postoDest
Ogni istanza di questa relationship lega una Destinazione ad un Posto da vedere (Req. 4.3.)
Attributi: Nessuno

Relationship prenCliente
Ogni istanza di questa relationship lega una Prenotazione al relativo Cliente (Req. 7.1.)
Attributi: Nessuno

Relationship prenCroc
Ogni istanza di questa relationship lega una Prenotazione alla relativa Crociera (Req. 7.3.)
Attributi: Nessuno

Dominio DeltaDataOra
Il dominio è un record composto dai seguenti campi:

- giorni: intero > 0
- ora: ora

Dominio FasciaOraria
Il dominio è un record composto dai seguenti campi:

- da: ora
- a: ora

Dominio Indirizzo
Il dominio è un record composto dai seguenti campi:

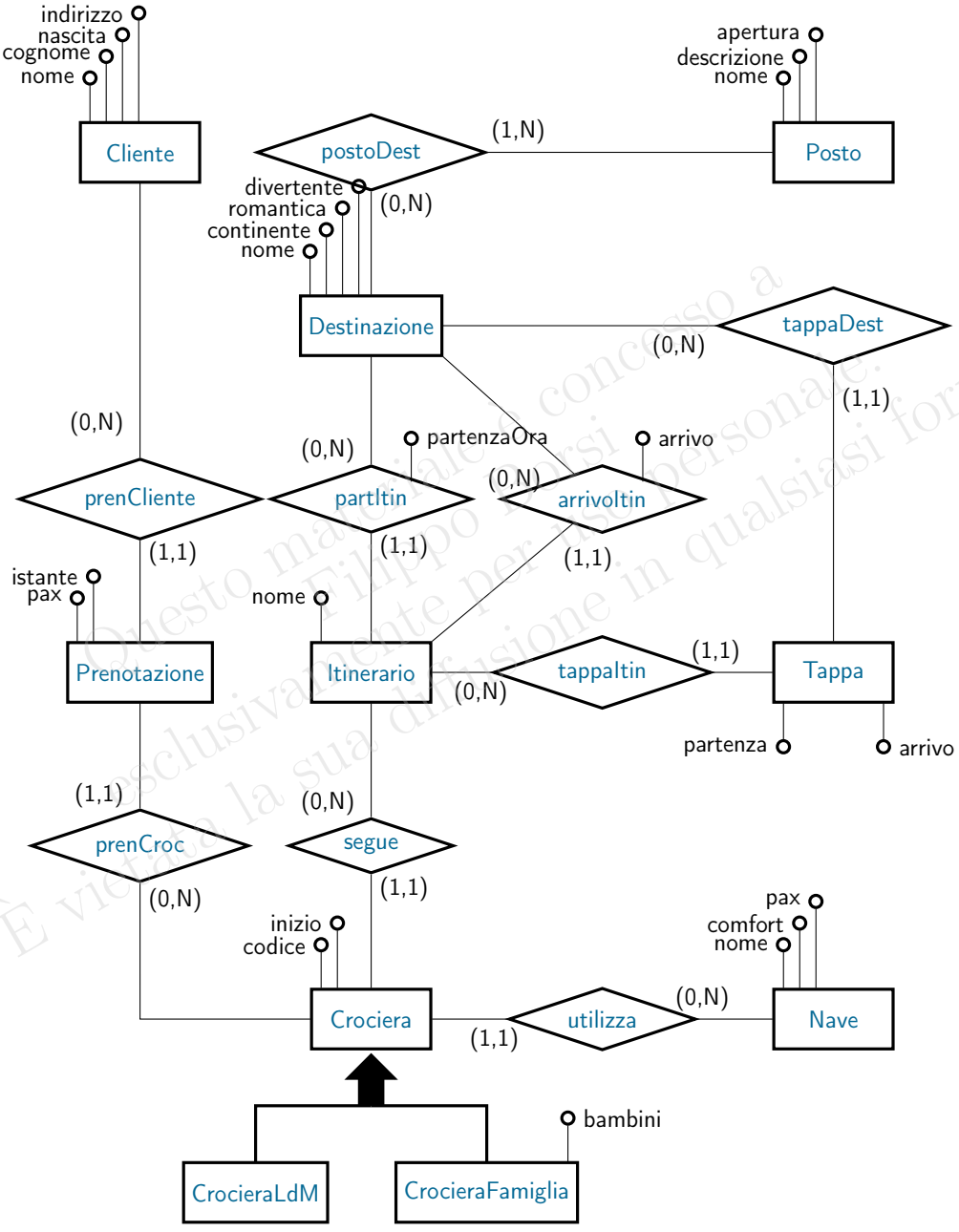
- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre numeriche
- città: stringa
- nazione: stringa

A.4 Estensione del Diagramma ER

A.4.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti estendendo il diagramma ER concettuale per l'applicazione ed il dizionario dei dati per modellare anche i seguenti requisiti: Req. 1.6. (con l'esclusione di Req. 1.6.1.), Req. 4.4., Req. 4.5.

A.4.2 Soluzione



Specifiche dei Dati

Entità Crociera

Ogni istanza di questa entità rappresenta una crociera (Req. 1.)

attributo	dominio	molteplicità	descrizione
codice	stringa		Il codice della crociera
inizio	data		La data di inizio della crociera

Entità Nave

Ogni istanza di questa entità rappresenta una nave (Req. 2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nave
comfort	[3,5]		Il grado di comfort della nave
pax	intero > 0		Il numero massimo di passeggeri per la nave

Entità Itinerario

Ogni istanza di questa entità rappresenta un'itinerario (Req. 3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'itinerario

Entità Destinazione

Ogni istanza di questa entità rappresenta una destinazione toccata da Itinerari (Req. 4.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della destinazione
continente	{E, AS, AMN, AMS, AF, O}		Il continente della destinazione (nell'ordine: Europa, Asia, America del Nord, America del Sud, Africa, Oceania)
romantica	booleano		true se la destinazione è classificata come romantica, false altrimenti
divertente	booleano		true se la destinazione è classificata come divertente, false altrimenti

Entità Tappa

Ogni istanza di questa entità rappresenta una Tappa toccata da un Itinerario (Req. 3.2.), escluse quelle di partenza e arrivo dell'Itinerario

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di arrivo alla Destinazione
partenza	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di ripartenza dalla Destinazione

Entità Posto

Ogni istanza di questa entità rappresenta un posto da vedere (Req. 5.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del posto da vedere
descrizione	stringa		La descrizione del posto da vedere
apertura	FasciaOraria		La fascia oraria di apertura del posto da vedere

Entità Cliente

Ogni istanza di questa entità rappresenta un cliente (Req. 6.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del cliente
cognome	stringa		Il cognome del cliente
nascita	data		La data di nascita del cliente
indirizzo	Indirizzo		L'indirizzo del cliente

Entità Prenotazione

Ogni istanza di questa entità rappresenta una prenotazione di una Crociera da parte di un Cliente (Req. 7.)

attributo	dominio	molteplicità	descrizione
pax	intero > 0		Il numero di posti della prenotazione
istante	dataora		L'instante nel quale è stata effettuata la prenotazione

Entità CrocieraLdM

Ogni istanza di questa entità rappresenta una crociera di luna di miele (Req. 1.6.1.)
Attributi: Nessuno

Entità CrocieraFamiglia

Ogni istanza di questa entità rappresenta una crociera per famiglia (Req. 1.6.2.)

attributo	dominio	molteplicità	descrizione
bambini	booleano		true se la crociera è adatta ai bambini, false altrimenti

Relationship utilizza

Ogni istanza di questa relationship lega una Crociera alla Nave utilizzata (Req. 1.4.)
Attributi: Nessuno

Relationship segue

Ogni istanza di questa relationship lega una Crociera al suo Itinerario (Req. 1.5.)
Attributi: Nessuno

Relationship tappaItin
Ogni istanza di questa relationship lega un Itinerario ad una sua Tappa (Req. 3.2.)
Attributi: Nessuno

Relationship tappaDest
Ogni istanza di questa relationship lega una Tappa di un Itinerario alla relativa Destinazione (Req. 3.2.)
Attributi: Nessuno

Relationship partItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di partenza (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
partenzaOra	ora		L'ora di partenza della Crociera

Relationship arrivoItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di arrivo (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (in termini di giorni dalla data di partenza dell'itinerario) e l'ora di arrivo della Crociera alla Tappa

Relationship postoDest
Ogni istanza di questa relationship lega una Destinazione ad un Posto da vedere (Req. 4.3.)
Attributi: Nessuno

Relationship prenCliente
Ogni istanza di questa relationship lega una Prenotazione al relativo Cliente (Req. 7.1.)
Attributi: Nessuno

Relationship prenCroc
Ogni istanza di questa relationship lega una Prenotazione alla relativa Crociera (Req. 7.3.)
Attributi: Nessuno

Dominio DeltaDataOra

Il dominio è un record composto dai seguenti campi:

- giorni: intero > 0
- ora: ora

Dominio FasciaOraria

Il dominio è un record composto dai seguenti campi:

- da: ora
- a: ora

Dominio Indirizzo

Il dominio è un record composto dai seguenti campi:

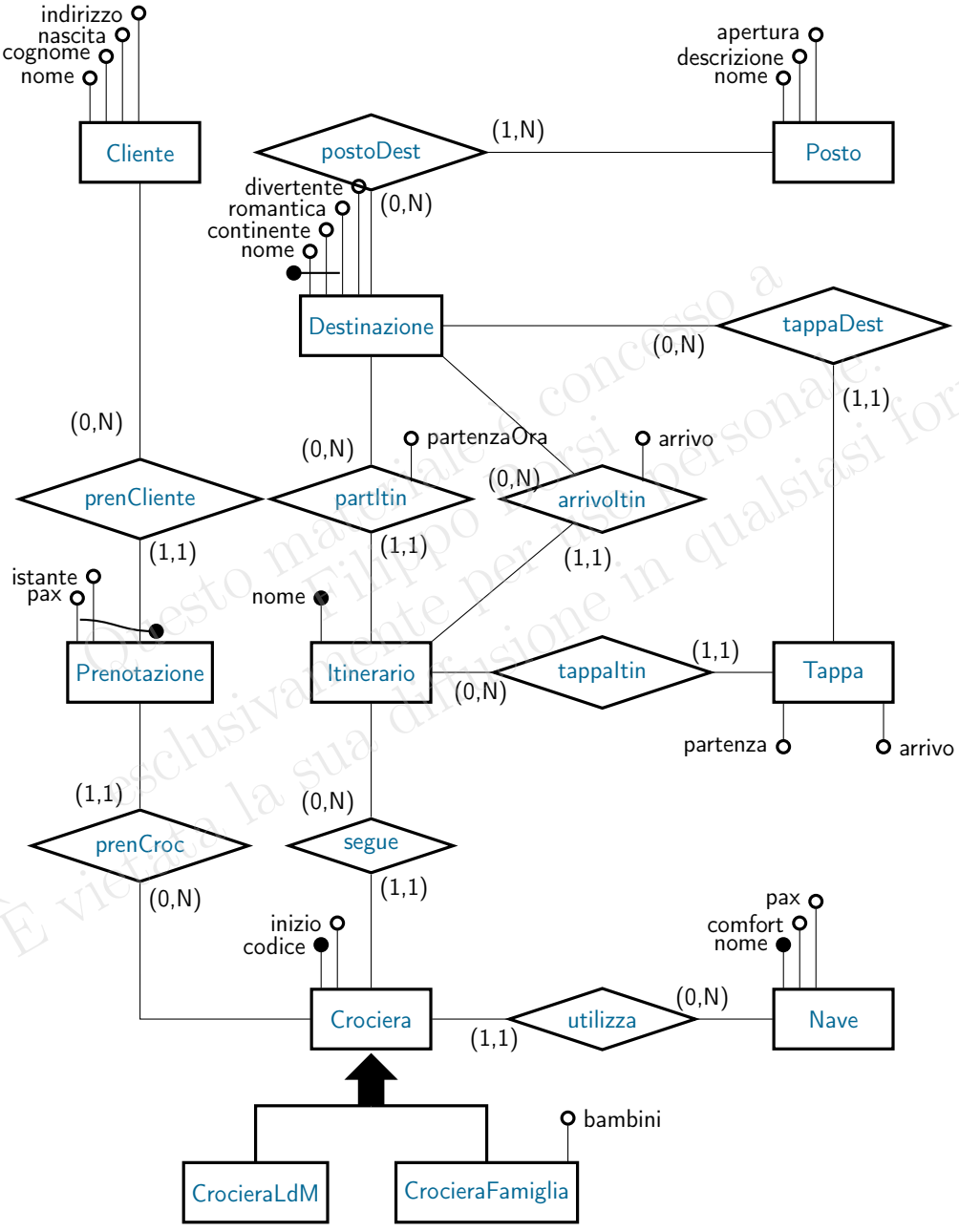
- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre numeriche
- città: stringa
- nazione: stringa

A.5 Identificazione delle Entità e Definizione Informale di Vincoli Esterni

A.5.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti estendendo il diagramma ER concettuale per l'applicazione ed il dizionario dei dati con eventuali vincoli di identificazione delle entità che provengono dai requisiti e la definizione informale di eventuali ulteriori vincoli esterni.

A.5.2 Soluzione



Specifiche dei Dati

Entità Crociera

Ogni istanza di questa entità rappresenta una crociera (Req. 1.)

attributo	dominio	molteplicità	descrizione
codice	stringa		Il codice della crociera
inizio	data		La data di inizio della crociera

Vincoli:

[V.Crociera.posti] Il numero di posti prenotati per ogni crociera non può mai eccedere il numero di posti disponibili nella nave utilizzata.

Per ogni istanza cr dell'entità Crociera, siano:

- postiOccupati il numero totale di posti prenotati per la crociera cr, ovvero la somma dei valori dell'attributo pax di tutte le istanze z dell'entità Prenotazione tali per cui (z, cr) è un'istanza della relationship prenCrociera
- totPosti il numero totale di posti disponibili sulla nave utilizzata dalla crociera cr, ovvero il valore dell'attributo pax dell'istanza n dell'entità Nave per la quale (cr, n) è un'istanza della relationship utilizza.

Deve essere totPosti − postiOccupati ≥ 0.

Entità Nave

Ogni istanza di questa entità rappresenta una nave (Req. 2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nave
comfort	[3,5]		Il grado di comfort della nave
pax	intero > 0		Il numero massimo di passeggeri per la nave

Vincoli:

[V.Nave.crociere.disj] Le crociere che utilizzano la stessa nave devono avere periodi tutti disgiunti.

Per ogni coppia di istanze diverse c e c' dell'entità Crociera che utilizzano la stessa Nave n (ovvero tali per cui sia (c,n) che (c',n) sono istanze della relationship utilizza), siano:

- c_in e c'_in i valori dell'attributo inizio per, rispettivamente, c e c'

- itin_c e itin_c' le istanze dell'entità Itinerario per cui si ha che (c, itin_c) e (c', itin_c') sono istanze della relationship segue
- dest_arrivo_c e dest_arrivo_c' le istanze dell'entità Destinazione per cui si ha che (dest_arrivo_c, itin_c) e (dest_arrivo_c', itin_c') sono istanze della relationship arrivoltin
- arrivo_c e arrivo_c' i valori (di dominio DeltaDataOra) dell'attributo arrivo, rispettivamente, delle istanze (dest_arrivo_c, itin_c) e (dest_arrivo_c', itin_c') della relationship arrivoltin
- giorno_arrivo_c e giorno_arrivo_c' i valori del campo giorni, rispettivamente, delle istanze arrivo_c e arrivo_c'.

Deve essere:

$$c'\_in > c\_in + giorno\_arrivo\_c \ \textit{giorni}$$
oppure
$$c\_in > c'\_in + giorno\_arrivo\_c' \ \textit{giorni}.$$

Entità Itinerario
Ogni istanza di questa entità rappresenta un'itinerario (Req. 3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'itinerario

Entità Destinazione
Ogni istanza di questa entità rappresenta una destinazione toccata da Itinerari (Req. 4.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della destinazione
continente	{E, AS, AMN, AMS, AF, O}		Il continente della destinazione (nell'ordine: Europa, Asia, America del Nord, America del Sud, Africa, Oceania)
romantica	booleano		true se la destinazione è classificata come romantica, false altrimenti
divertente	booleano		true se la destinazione è classificata come divertente, false altrimenti

Entità Tappa
Ogni istanza di questa entità rappresenta una Tappa toccata da un Itinerario (Req. 3.2.), escluse quelle di partenza e arrivo dell'Itinerario

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di arrivo alla Destinazione
partenza	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di ripartenza dalla Destinazione

Vincoli:

[V.Tappa.date] La data di partenza da ogni tappa segue la relativa data di arrivo.

Per ogni istanza t dell'entità Tappa avente a e p come valori, rispettivamente, dei suoi attributi arrivo e partenza, deve essere: $a < p$.

[V.Tappa.succPart] Le istanze dell'entità Tappa con giorno di arrivo pari ad 1 devono avere valore per l'ora di arrivo successivo all'orario di partenza del relativo itinerario. In altri termini, eventuali tappe di un'itinerario previste nel giorno di viaggio 1 devono essere previste in orari successivi a quello di inizio dell'itinerario.

Per ogni istanza t dell'entità Tappa, siano:

- i l'itinerario relativo a t, ovvero l'istanza di Itinerario tale che (t, i) è un'istanza della relationship tappaItin
- p la destinazione di partenza di i, ovvero l'istanza dell'entità Destinazione tale che (p, i) è una istanza della relationship partItin
- o_p l'ora di partenza dell'itinerario dalla destinazione di partenza p, ovvero il valore dell'attributo partenzaOra dell'istanza (p, i) della relationship partItin
- a_t il momento di arrivo previsto alla tappa t, ovvero il valore dell'attributo arrivo (di dominio DeltaDataOra) dell'istanza t
- g_t e o_t i valori, rispettivamente, dei campi giorni e ora di a_t.

Se il campo g_t è pari ad 1, allora deve essere: $o_p < o_t$.

[V.Tappa.primaDiArrivo] Le istanze dell'entità Tappa devono avere un valore per l'attributo partenza precedente al valore dell'attributo arrivo della destinazione finale del relativo itinerario.

Per ogni istanza t dell'entità Tappa, siano:

- i l'itinerario relativo a t, ovvero l'istanza di Itinerario tale che (t, i) è un'istanza della relationship tappaItin
- dest_arr la destinazione di arrivo di i, ovvero l'istanza dell'entità Destinazione tale che (dest_arr, i) è una istanza della relationship arrivoItin
- arr l'istante di arrivo dell'itinerario i alla destinazione di arrivo dest_arr, ovvero il valore (di dominio DeltaDataOra) dell'attributo arrivo dell'istanza (dest_arr, i) della relationship arrivoItin
- t_part il momento di partenza previsto dalla tappa t, ovvero il valore dell'attributo partenza (di dominio DeltaDataOra) dell'istanza t

Deve essere: t_part $<$ arr.

Entità Posto
Ogni istanza di questa entità rappresenta un posto da vedere (Req. 5.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del posto da vedere
descrizione	stringa		La descrizione del posto da vedere
apertura	FasciaOraria		La fascia oraria di apertura del posto da vedere

Entità Cliente
Ogni istanza di questa entità rappresenta un cliente (Req. 6.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del cliente
cognome	stringa		Il cognome del cliente
nascita	data		La data di nascita del cliente
indirizzo	Indirizzo		L'indirizzo del cliente

Entità Prenotazione
Ogni istanza di questa entità rappresenta una prenotazione di una Crociera da parte di un Cliente (Req. 7.)

attributo	dominio	molteplicità	descrizione
pax	intero > 0		Il numero di posti della prenotazione
istante	dataora		L'instante nel quale è stata effettuata la prenotazione

Vincoli:

[V.Prenotazione.istante] Le prenotazioni devono essere effettuate prima del giorno di inizio della relativa crociera.

Per ogni istanza p di entità Prenotazione, siano:

- i il valore dell'attributo istante di p
- cr l'istanza dell'entità Crociera per la quale (p, cr) è un'istanza della relationship prenCroc
- cr_i il valore dell'attributo inizio di cr.

Deve essere i $<$ cr_i.

Entità CrocieraLdM

Ogni istanza di questa entità rappresenta una crociera di luna di miele (Req. 1.6.1.)
Attributi: Nessuno

Entità CrocieraFamiglia

Ogni istanza di questa entità rappresenta una crociera per famiglia (Req. 1.6.2.)

attributo	dominio	molteplicità	descrizione
bambini	booleano		true se la crociera è adatta ai bambini, false altrimenti

Relationship utilizza

Ogni istanza di questa relationship lega una Crociera alla Nave utilizzata (Req. 1.4.)
Attributi: Nessuno

Relationship segue

Ogni istanza di questa relationship lega una Crociera al suo Itinerario (Req. 1.5.)
Attributi: Nessuno

Relationship tappaItin

Ogni istanza di questa relationship lega un Itinerario ad una sua Tappa (Req. 3.2.)
Attributi: Nessuno

Vincoli:

[V.tappaItin.soste.disj] I periodi di sosta di ogni itinerario nelle diverse tappe sono tutti disgiunti.

Per ogni coppia di istanze diverse t e t' dell'entità Tappa relative allo stesso Itinerario i (ovvero tali per cui sia (t,i) che (t',i) sono istanze della relationship tappaItin), siano:

- t_arr e t_part i valori degli attributi, rispettivamente, arrivo e partenza di t
- t'_arr e t'_part i valori degli attributi, rispettivamente, arrivo e partenza di t'.

Deve essere: t'_arr $>$ t_part oppure t_arr $>$ t'_part.

Relationship tappaDest
Ogni istanza di questa relationship lega una Tappa di un Itinerario alla relativa Destinazione (Req. 3.2.)
Attributi: Nessuno

Relationship partItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di partenza (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
partenzaOra	ora		L'ora di partenza della Crociera

Relationship arrivoItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di arrivo (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (in termini di giorni dalla data di partenza dell'itinerario) e l'ora di arrivo della Crociera alla Tappa

Relationship postoDest
Ogni istanza di questa relationship lega una Destinazione ad un Posto da vedere (Req. 4.3.)
Attributi: Nessuno

Relationship prenCliente
Ogni istanza di questa relationship lega una Prenotazione al relativo Cliente (Req. 7.1.)
Attributi: Nessuno

Relationship prenCroc
Ogni istanza di questa relationship lega una Prenotazione alla relativa Crociera (Req. 7.3.)
Attributi: Nessuno

Dominio DeltaDataOra
Il dominio è un record composto dai seguenti campi:

- giorni: intero > 0
- ora: ora

Dominio FasciaOraria
Il dominio è un record composto dai seguenti campi:

- da: ora
- a: ora

Dominio Indirizzo
Il dominio è un record composto dai seguenti campi:

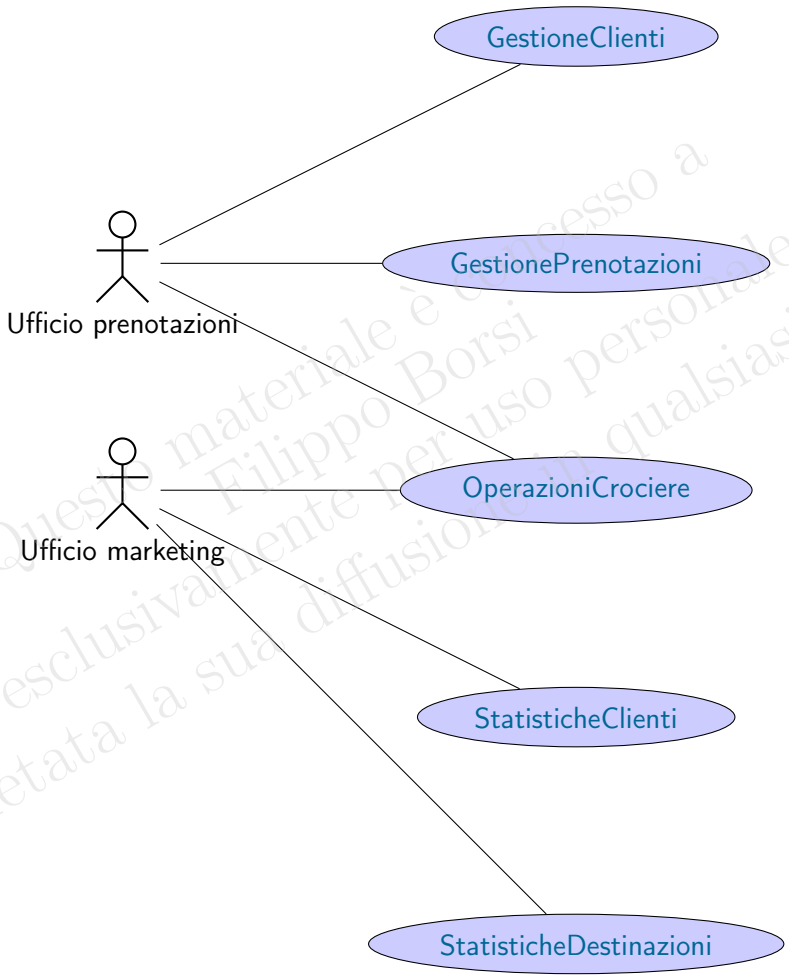
- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre numeriche
- città: stringa
- nazione: stringa

A.6 Diagramma UML degli Use-Case

A.6.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti producendo un diagramma UML degli use-case.

A.6.2 Soluzione



A.7 Specifiche Concettuali Informali degli Use-Case

A.7.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti producendo le specifiche concettuali informali degli use-case.

A.7.2 Soluzione

Specifica Use-Case GestionePrenotazioni

- **prenota(cl : Cliente, cr : Crociera, posti: intero > 0) : Prenotazione** (Req. 8.1.)

  precondizioni: La data di inizio della crociera cr deve essere successiva all'istante corrente.

  Inoltre, la crociera cr deve avere almeno posti posti liberi, ovvero: GestionePrenotazioni.postiDisponibili(cr) $\geq$ posti.

  postcondizioni: Il livello estensionale dei dati viene modificato creando le seguenti nuove istanze:

  - α dell'entità Prenotazione, con valore posti per l'attributo pax e valore per l'attributo istante pari all'istante corrente
  - (α, cl) della relationship prenCliente
  - (α, cr) della relationship prenCrociera

  Viene restituita l'istanza α.

- **postiDisponibili(cr : Crociera) : intero**

  precondizioni: nessuna

  postcondizioni: La funzione non modifica il livello estensionale dei dati.

  Siano:

  - postiOccupati il numero totale di posti prenotati per la crociera cr, ovvero la somma dei valori dell'attributo pax di tutte le istanze z dell'entità Prenotazione tali per cui (z, cr) è un'istanza della relationship prenCrociera
  - totPosti il numero totale di posti disponibili sulla nave utilizzata dalla crociera cr, ovvero il valore dell'attributo pax dell'istanza n dell'entità Nave per la quale (cr, n) è un'istanza della relationship utilizza.

  Viene restituito totPosti − postiOccupati.

Specifica Use-Case GestioneClienti

- **nuovo(no : stringa, c : stringa, na: data, i: Indirizzo) : Cliente**

  precondizioni: nessuna

  postcondizioni: Il livello estensionale dei dati viene modificato creando la seguente nuova istanza:

  – α dell'entità Cliente, con valori no, c, na ed i rispettivamente per gli attributi nome, cognome, nascita, indirizzo.

  Viene restituita l'istanza α.

- **cerca(no : stringa (0,1), co : stringa (0,1), na: data (0,1)) : Cliente (0,N)**

  precondizioni: nessuna

  postcondizioni: La funzione non modifica il livello estensionale dei dati.

  Viene restituito l'insieme delle istanze cl dell'entità Cliente che soddisfano tutti i seguenti criteri:

  – se no è definito, allora il valore dell'attributo nome di cl deve essere uguale a no

  – se c è definito, allora il valore dell'attributo cognome di cl deve essere uguale a no

  – se na è definito, allora il valore dell'attributo nascita di cl deve essere uguale a na.

Specifica Use-Case OperazioniCrociere

- **tipoCrociera(c : CrocieraLdM) : {tradizionale, alternativa} (Req. 1.6.1.)**

precondizioni: nessuna

postcondizioni: La funzione non modifica il livello estensionale dei dati.

Sia i l'itinerario della crociera c, ovvero l'istanza dell'entità Itinerario tale per cui (c, i) è istanza della relationship segue.

Siano:

- D_r l'insieme delle destinazioni romantiche della crociera c, ovvero l'insieme delle istanze d dell'entità Destinazione tali che:
  - esiste un'istanza t dell'entità Tappa tale che (t, i) è istanza della relationship tappaItin e (t, d) è istanza della relationship tappaDest
  - il valore dell'attributo romantica di d è true
- D_d l'insieme delle destinazioni divertenti della crociera c, ovvero l'insieme delle istanze d dell'entità Destinazione tali che:
  - esiste un'istanza t dell'entità Tappa tale che (t, i) è istanza della relationship tappaItin e (t, d) è istanza della relationship tappaDest
  - il valore dell'attributo divertente di d è true.

Viene restituito il valore 'tradizionale' se $|D_r| \geq |D_d|$ e 'alternativa' altrimenti.

Specifica Use-Case StatisticheClienti

- **etaMediaEsotica(C : Cliente (0,N)) : reale ≥ 0 (Req. 8.2.)**

  precondizioni: Sia C_e il sottoinsieme dei clienti in C che hanno prenotato almeno una crociera il cui itinerario tocca una destinazione esotica, ovvero l'insieme contenente tutti e soli gli elementi c $\in$ C che soddisfano tutti i seguenti criteri:

  - esiste una istanza p di entità Prenotazione tale per cui (p,c) è istanza della relationship prenCliente
  - esiste una istanza cr di entità Crociera tale per cui (p,cr) è istanza della relationship prenCroc
  - esiste una istanza i di entità Itinerario tale per cui (cr,i) è istanza della relationship segue
  - esiste una istanza t di entità Tappa tale per cui (t,i) è istanza della relationship tappaItin
  - esiste una istanza d di entità Destinazione tale per cui (t,d) è istanza della relationship tappaDest
  - StatisticheDestinazioni.esotica(d).

  Deve essere $|C_e| \geq 1$.

  postcondizioni: La funzione non modifica il livello estensionale dei dati.

  Viene restituito:

$$\text{result} = \frac{\sum_{c \in C_e} \text{StatisticheClienti.eta}(c)}{|C_e|}.$$

- **eta(c : Cliente) : intero ≥ 0**

  precondizioni: nessuna

  postcondizioni: La funzione non modifica il livello estensionale dei dati.

  Sia na il valore (di dominio data) dell'attributo nascita di c e sia diff la differenza (di dominio data) tra la data corrente e il valore na.

  Viene restituito il valore del campo anno di diff.

Specifica Use-Case StatisticheDestinazioni

- **esotica(d : Destinazione) : booleano**

  precondizioni: nessuna

  postcondizioni: La funzione non modifica il livello estensionale dei dati.
  Sia cont il valore dell'attributo continente di d.
  Viene restituito 'true' se cont $\neq$ 'E' e 'false' altrimenti.

- **gettonata(d : Destinazione) : booleano**

  precondizioni: nessuna

  postcondizioni: La funzione non modifica il livello estensionale dei dati.
  Siano L e F l'insieme, rispettivamente, delle crociere di luna di miele e per famiglia degli ultimi due anni che toccano la destinazione d:
  - L è l'insieme delle istanze ldm dell'entità CrocieraLdM che soddisfano tutti i seguenti criteri:
    * ldm ha un valore in per l'attributo inizio (di dominio data) il cui campo anno ha un valore a tale per cui la differenza tra il valore dell'anno corrente e a è al più 2
    * esiste una istanza it di entità Itinerario tale per cui (ldm, it) è una istanza della relationship segue
    * esiste una istanza t di entità Tappa tale per cui (it, t) è una istanza della relationship tappaItin
    * (t, d) è una istanza della relationship tappaDest.
  - F è l'insieme delle istanze fam dell'entità CrocieraFamiglia che soddisfano tutti i seguenti criteri:
    * fam ha un valore in per l'attributo inizio (di dominio data) il cui campo anno ha un valore a tale per cui la differenza tra il valore dell'anno corrente e a è al più 2
    * esiste una istanza it di entità Itinerario tale per cui (fam, it) è una istanza della relationship segue
    * esiste una istanza t di entità Tappa tale per cui (it, t) è una istanza della relationship tappaItin
    * (t, d) è una istanza della relationship tappaDest.

  Viene restituito 'true' se $|L| \geq 10$ oppure se $|F| \geq 15$, e 'false' altrimenti.

- **percentualeGettonate(D: Destinazione (1,N)): reale in [0,1] (Req. 8.3.)**

  precondizioni: nessuna

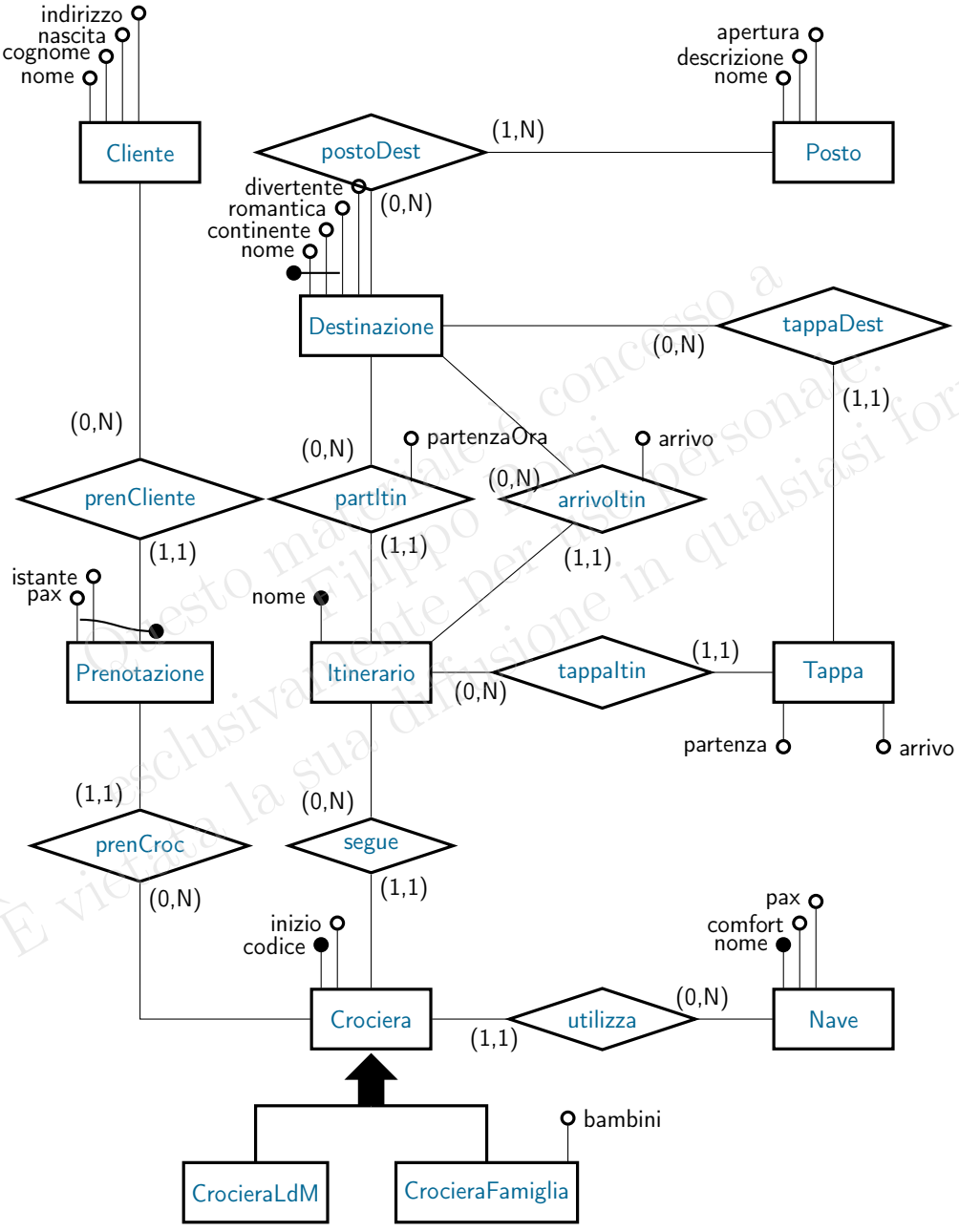
postcondizioni: La funzione non modifica il livello estensionale dei dati.
Sia D' l'insieme composto da tutte e sole le destinazioni d $\in$ D gettonate, ovvero per cui vale: StatisticheDestinazioni.gettonata(d) = true.
Viene restituito $\frac{|D'|}{|D|}$.

A.8 Vincoli Esterni

A.8.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti formalizzando, in logica del primo ordine, i vincoli esterni già definiti informalmente.

A.8.2 Soluzione



Specifiche dei Dati

Entità Crociera

Ogni istanza di questa entità rappresenta una crociera (Req. 1.)

attributo	dominio	molteplicità	descrizione
codice	stringa		Il codice della crociera
inizio	data		La data di inizio della crociera

Vincoli:

[V.Crociera.posti] Il numero di posti prenotati per ogni crociera non può mai eccedere il numero di posti disponibili nella nave utilizzata.

Formalmente:

$$\forall \text{cr Crociera(cr)} \rightarrow \text{GestionePrenotazioni.postiDisponibili (cr)} \geq 0$$

Entità Nave

Ogni istanza di questa entità rappresenta una nave (Req. 2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nave
comfort	[3,5]		Il grado di comfort della nave
pax	intero > 0		Il numero massimo di passeggeri per la nave

Vincoli:

[V.Nave.crociere.disj] Le crociere che utilizzano la stessa nave devono avere periodi tutti disgiunti.

Formalmente:

$$\forall \text{c, c',n, c\_in, i\_c, i\_c', dest\_arr\_c, dest\_arr\_c', arr\_c, arr\_c', arr\_g\_c, arr\_g\_c'}$$
$$[\text{Crociera(c)} \wedge \text{Crociera(c')} \wedge \text{c} \neq \text{c'} \wedge \text{utilizza(c, n)} \wedge \text{utilizza(c', n)} \wedge$$
$$\text{inizio(c, c\_in)} \wedge \text{inizio(c', c'\_in)} \wedge \text{segue(c, i\_c)} \wedge \text{segue(c', i\_c')} \wedge$$
$$\text{arrivoltin(dest\_arr\_c, i\_c)} \wedge \text{arrivoltin(dest\_arr\_c', i\_c')} \wedge$$
$$\text{arrivo(dest\_arr\_c, i\_c, arr\_c)} \wedge \text{arrivo(dest\_arr\_c', i\_c', arr\_c')} \wedge$$
$$\text{giorni(arr\_c, arr\_g\_c)} \wedge \text{giorni(arr\_c', arr\_g\_c')}] \rightarrow$$
$$[\text{c'\_in} > \text{c\_in} + \text{arr\_g\_c} \vee \text{c\_in} > \text{c'\_in} + \text{arr\_g\_c'}].$$

Entità Itinerario

Ogni istanza di questa entità rappresenta un'itinerario (Req. 3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'itinerario

Entità Destinazione

Ogni istanza di questa entità rappresenta una destinazione toccata da Itinerari (Req. 4.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della destinazione
continente	{E, AS, AMN, AMS, AF, O}		Il continente della destinazione (nell'ordine: Europa, Asia, America del Nord, America del Sud, Africa, Oceania)
romantica	booleano		true se la destinazione è classificata come romantica, false altrimenti
divertente	booleano		true se la destinazione è classificata come divertente, false altrimenti

Entità Tappa

Ogni istanza di questa entità rappresenta una Tappa toccata da un Itinerario (Req. 3.2.), escluse quelle di partenza e arrivo dell'Itinerario

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di arrivo alla Destinazione
partenza	DeltaDataOra		La data (relativa alla data di partenza della crociera) e l'ora di ripartenza dalla Destinazione

Vincoli:

[V.Tappa.date] La data di partenza da ogni tappa segue la relativa data di arrivo.

Formalmente:

$$\forall t,p,a\ \text{Tappa}(t) \wedge \text{partenza}(t,p) \wedge \text{arrivo}(t,a) \rightarrow (a < p)$$

[V.Tappa.succPart] Le istanze di questa entità con giorno di arrivo pari ad 1 devono avere valore per l'ora di arrivo successivo all'orario di partenza del relativo itinerario.

Formalmente:

$$\begin{aligned}\forall i, p, t, a_t, o_t, o_p \ & \text{Itinerario}(i) \wedge \text{partItin}(p, i) \wedge \text{partenzaOra}(p, i, o_p) \wedge \\ & \text{Tappa}(t) \wedge \text{tappaItin}(t, i) \wedge \text{arrivo}(t, a_t) \wedge \\ & \text{giorno}(a_t, 1) \wedge \text{ora}(a_t, o_t) \ \rightarrow \\ & o_p < o_t\end{aligned}$$

[V.Tappa.primaDiArrivo] Le istanze dell'entità Tappa devono avere un valore per l'attributo partenza precedente al valore dell'attributo arrivo della destinazione finale del relativo itinerario.

Formalmente:

$$\begin{aligned}\forall i, \text{dest\_arr}, \text{arr}, \text{t\_part}, t \ & \text{Itinerario}(i) \wedge \text{arrivoItin}(\text{dest\_arr}, i) \wedge \text{arrivo}(\text{dest\_arr}, i, \text{arr}) \wedge \\ & \text{Tappa}(t) \wedge \text{tappaItin}(t, i) \wedge \text{partenza}(t, \text{t\_part}) \ \rightarrow \\ & \text{t\_part} < \text{arr}\end{aligned}$$

Entità Posto

Ogni istanza di questa entità rappresenta un posto da vedere (Req. 5.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del posto da vedere
descrizione	stringa		La descrizione del posto da vedere
apertura	FasciaOraria		La fascia oraria di apertura del posto da vedere

Entità Cliente

Ogni istanza di questa entità rappresenta un cliente (Req. 6.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del cliente
cognome	stringa		Il cognome del cliente
nascita	data		La data di nascita del cliente
indirizzo	Indirizzo		L'indirizzo del cliente

Entità Prenotazione

Ogni istanza di questa entità rappresenta una prenotazione di una Crociera da parte di un Cliente (Req. 7.)

attributo	dominio	molteplicità	descrizione
pax	intero > 0		Il numero di posti della prenotazione
istante	dataora		L'instante nel quale è stata effettuata la prenotazione

Vincoli:

[V.Prenotazione.istante] Le prenotazioni devono essere effettuate prima del giorno di inizio della relativa crociera.

Formalmente:

$$\forall p,i,cr,cr\_i \quad Prenotazione(p) \wedge istante(p,i) \wedge prenCroc(p,cr) \wedge Crociera(cr) \wedge inizio(cr,cr\_i) \rightarrow i < cr\_i.$$

Entità CrocieraLdM

Ogni istanza di questa entità rappresenta una crociera di luna di miele (Req. 1.6.1.)
Attributi: Nessuno

Entità CrocieraFamiglia

Ogni istanza di questa entità rappresenta una crociera per famiglia (Req. 1.6.2.)

attributo	dominio	molteplicità	descrizione
bambini	booleano		true se la crociera è adatta ai bambini, false altrimenti

Relationship utilizza

Ogni istanza di questa relationship lega una Crociera alla Nave utilizzata (Req. 1.4.)
Attributi: Nessuno

Relationship segue

Ogni istanza di questa relationship lega una Crociera al suo Itinerario (Req. 1.5.)
Attributi: Nessuno

Relationship tappaItin
Ogni istanza di questa relationship lega un Itinerario ad una sua Tappa (Req. 3.2.)
Attributi: Nessuno

Vincoli:

[V.tappaItin.soste.disj] I periodi di sosta di ogni itinerario nelle diverse tappe sono tutti disgiunti.

Formalmente:

$$\forall i, t, t', t\_part, t\_arr, t'\_part, t'\_arr$$
$$Itinerario(i) \land Tappa(t) \land tappaItin(t, i) \land arrivo(t, t\_arr) \land partenza(t, t\_part) \land$$
$$Tappa(t') \land tappaItin(t', i) \land arrivo(t', t'\_arr) \land partenza(t, t'\_part) \land$$
$$t \neq t' \rightarrow$$
$$(t'\_arr > t\_part) \lor (t\_arr > t'\_part)$$

Relationship tappaDest
Ogni istanza di questa relationship lega una Tappa di un Itinerario alla relativa Destinazione (Req. 3.2.)
Attributi: Nessuno

Relationship partItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di partenza (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
partenzaOra	ora		L'ora di partenza della Crociera

Relationship arrivoItin
Ogni istanza di questa relationship lega un Itinerario alla relativa Destinazione di arrivo (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
arrivo	DeltaDataOra		La data (in termini di giorni dalla data di partenza dell'itinerario) e l'ora di arrivo della Crociera alla Tappa

Relationship postoDest
Ogni istanza di questa relationship lega una Destinazione ad un Posto da vedere (Req. 4.3.)
Attributi: Nessuno

Relationship prenCliente
Ogni istanza di questa relationship lega una Prenotazione al relativo Cliente (Req. 7.1.)
Attributi: Nessuno

Relationship prenCroc
Ogni istanza di questa relationship lega una Prenotazione alla relativa Crociera (Req. 7.3.)
Attributi: Nessuno

Dominio DeltaDataOra

Il dominio è un record composto dai seguenti campi:

- giorni: intero > 0
- ora: ora

Le istanze del dominio sono totalmente ordinate. A tal fine verranno utilizzati i simboli di predicato binari $\leq$, $=$ e $<$ in forma infissa, definiti per coppie di istanze del dominio come segue:

$$\forall d', d'', g', o', g'', o'' \left(\begin{array}{l} \text{DeltaDataOra}(d') \wedge \text{DeltaDataOra}(d'') \wedge \\ \text{giorni}(d', g') \wedge \text{ore}(d', o') \wedge \\ \text{giorni}(d'', g'') \wedge \text{ore}(d'', o'') \end{array} \right) \rightarrow [(d' \leq d'') \leftrightarrow (g' < g'' \vee (g' = g'' \wedge o' \leq o''))]$$

$$\forall d', d'' \left(\text{DeltaDataOra}(d') \wedge \text{DeltaDataOra}(d'') \right) \rightarrow [(d' = d'') \leftrightarrow (d' \leq d'' \wedge d'' \leq d')]$$

$$\forall d', d'' \left(\text{DeltaDataOra}(d') \wedge \text{DeltaDataOra}(d'') \right) \rightarrow [(d' < d'') \leftrightarrow (d' \leq d'' \wedge \neg(d' = d''))]$$

Dominio FasciaOraria

Il dominio è un record composto dai seguenti campi:

- da: ora

- a: ora

Dominio Indirizzo

Il dominio è un record composto dai seguenti campi:

- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre numeriche
- città: stringa
- nazione: stringa

A.9 Specifiche Concettuali degli Use-Case

A.9.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti formalizzando, in logica del primo ordine, le specifiche concettuali degli use-case già definite informalmente.

A.9.2 Soluzione

Specifica Use-Case GestionePrenotazioni

- **prenota(cl : Cliente, cr : Crociera, posti: intero > 0) : Prenotazione** (Req. 8.1.)

  precondizioni: $\forall$in inizio(cr, inizio) $\rightarrow$ in $<$ adesso $\wedge$ GestionePrenotazioni.postiDisponibili (cr) $\geq$ posti

  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: un nuovo elemento α

  Variazioni nelle ennuple di predicati: Prenotazione(α), prenCliente(cl, α), prenCrociera(α, cr), pax(α, posti), istante(α, adesso).

  Valore di ritorno: result $= \alpha$

- **postiDisponibili(cr : Crociera) : intero ≥ 0**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno: Detto:

  $$Z = \{(z, p) \mid \text{Prenotazione}(z) \wedge \text{prenCrociera}(z, cr) \wedge \text{pax}(z, p)\}$$

  sia

  $$\text{postiOccupati} = \sum_{(z,p) \in Z} p.$$

  Sia inoltre totPosti il valore dell'attributo pax della nave utilizzata per la crociera cr, ovvero tale da soddisfare la seguente formula:

  $$\exists n \ \text{utilizza}(cr, n) \wedge \text{pax}(n, \text{totPosti}).$$

  result $=$ totPosti $-$ postiOccupati.

Specifica Use-Case GestioneClienti

- **nuovo(no : stringa, c : stringa, na: data, i: Indirizzo) : Cliente**

  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: un nuovo elemento α

  Variazioni nelle ennuple di predicati: Cliente(α), nome(α, no), cognome(α, c), nascita(α, na), indirizzo(α, i)

  Valore di ritorno: result $= \alpha$

- **cerca(no : stringa (0,1), co : stringa (0,1), na: data (0,1)) : Cliente (0,N)**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno: Siano definite le seguenti formule (aperte in c):

  - φ_{nome} : nome(c, no) se no è definito, e true altrimenti
  - φ_{cognome} : cognome(c, co) se co è definito, e true altrimenti
  - φ_{nascita} : nascita(c, na) se na è definito, e true altrimenti.

  result $= \{c \mid \text{Cliente(c)} \wedge \varphi_{\text{nome}}(\text{c}) \wedge \varphi_{\text{cogn}}(\text{c}) \wedge \varphi_{\text{nascita}}(\text{c})\}$.

Specifica Use-Case OperazioniCrociere

- **tipoCrociera(c : CrocieraLdM) : {tradizionale, alternativa} (Req. 1.6.1.)**

precondizioni: nessuna

postcondizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: Siano:

$$D_r = \left\{ \mathsf{d} \;\middle|\; \begin{array}{l} \exists \mathsf{i}, \mathsf{t}\ \mathsf{Itinerario(i)} \wedge \mathsf{segue(c,i)} \wedge \\ \mathsf{tappaltin(t,i)} \wedge \mathsf{tappaDest(t,d)} \wedge \\ \mathsf{Destinazione(d)} \wedge \mathsf{romantica(d,true)} \end{array} \right\}$$

$$D_d = \left\{ \mathsf{d} \;\middle|\; \begin{array}{l} \exists \mathsf{i}, \mathsf{t}\ \mathsf{Itinerario(i)} \wedge \mathsf{segue(c,i)} \wedge \\ \mathsf{tappaltin(t,i)} \wedge \mathsf{tappaDest(t,d)} \wedge \\ \mathsf{Destinazione(d)} \wedge \mathsf{divertente(d,true)} \end{array} \right\}$$

i sottoinsiemi (potenzialmente non disgiunti) delle destinazioni della crociera di luna di miele c rispettivamente romantiche e divertenti.

$$\mathit{result} = \begin{cases} \text{tradizionale} & \text{se } |D_r| \geq |D_d| \\ \text{alternativa} & \text{altrimenti} \end{cases}$$

Specifica Use-Case StatisticheClienti

- **etaMediaEsotica(C : Cliente (0,N)) : reale ≥ 0 (Req. 8.2.)**

  precondizioni: Sia

$$C_e = \left\{ c \in C \;\middle|\; \begin{array}{ll} \exists p, cr, i, t, d & prenCliente(c, p) \wedge Prenotazione(p) \wedge \\ & prenCrociera(p, cr) \wedge Crociera(cr) \wedge \\ & segue(cr, i) \wedge Itinerario(i) \wedge tappaltin(t, i) \wedge \\ & Tappa(t) \wedge tappaDest(t, d) \wedge \\ & Destinazione(d) \wedge \\ & StatisticheDestinazioni.esotica(d) \end{array} \right\}$$

  il sottoinsieme dei clienti in C che hanno prenotato almeno una crociera il cui itinerario tocca una destinazione esotica.

  Deve essere $|C_e| \geq 1$.

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno:

$$result = \frac{\sum_{c \in C_e} StatisticheClienti.eta(c)}{|C_e|}.$$

- **eta(c : Cliente) : intero ≥ 0**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno: Siano na, anni e diff tali da soddisfare la seguente formula (esistono e sono unici):

$$nascita(c, na) \wedge diff = adesso - na \wedge anno(diff, anni).$$

  result = anni.

Specifica Use-Case StatisticheDestinazioni

- **esotica(d : Destinazione) : booleano**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna
  Valore di ritorno: result $= \exists$cont continente(d, cont) $\wedge$ cont $\neq$ 'E'.

- **gettonata(d : Destinazione) : booleano**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna
  Valore di ritorno: Siano

$$N_{ldm} = \left| \left\{ \text{ldm} \;\middle|\; \exists \text{in, it, t, diff, anni} \quad \begin{array}{l} \text{CrocieraLdM(ldm)} \wedge \\ \text{inizio(ldm, in)} \wedge \\ \text{diff} = \text{adesso} - \text{in} \wedge \\ \text{anno(diff, anni)} \wedge \\ \text{anni} \leq 2 \wedge \\ \text{segue(ldm, it)} \wedge \\ \text{Itinerario(it)} \wedge \\ \text{tappaItin(it, t)} \wedge \\ \text{tappaDest(t, d)} \end{array} \right\} \right|$$

e

$$N_{fam} = \left| \left\{ \text{fam} \;\middle|\; \exists \text{in, it, t, diff, anni} \quad \begin{array}{l} \text{CrocieraFamiglia(fam)} \wedge \\ \text{inizio(fam, in)} \wedge \\ \text{diff} = \text{adesso} - \text{in} \wedge \\ \text{anno(diff, anni)} \wedge \\ \text{anni} \leq 2 \wedge \\ \text{segue(fam, it)} \wedge \\ \text{Itinerario(it)} \wedge \\ \text{tappaItin(it, t)} \wedge \\ \text{tappaDest(t, d)} \end{array} \right\} \right|$$

rispettivamente il numero di crociere di luna di miele e per famiglia degli ultimi due anni che toccano la destinazione d.
result $= (N_{ldm} \geq 10) \vee (N_{fam} \geq 15)$.

- **percentualeGettonate(D: Destinazione (1,N)): reale in [0,1] (Req. 8.3.)**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno: Sia D' il sottoinsieme di D composto da tutte e sole le destinazioni gettonate, ovvero:

  $$D' = \{d \in D \mid StatisticheDestinazioni.gettonata(d) = true\}.$$

  result $= \frac{|D'|}{|D|}$.

Parte P

Progettazione della Base Dati e delle Funzionalità

P.1 Scelta del DBMS e Ristrutturazione del Diagramma ER e delle Specifiche dei Dati

P.1.1 Testo

Iniziare la fase di progettazione logica della base di dati decidendo il DBMS da utilizzare e ristrutturando il diagramma ER concettuale e le specifiche dei dati.

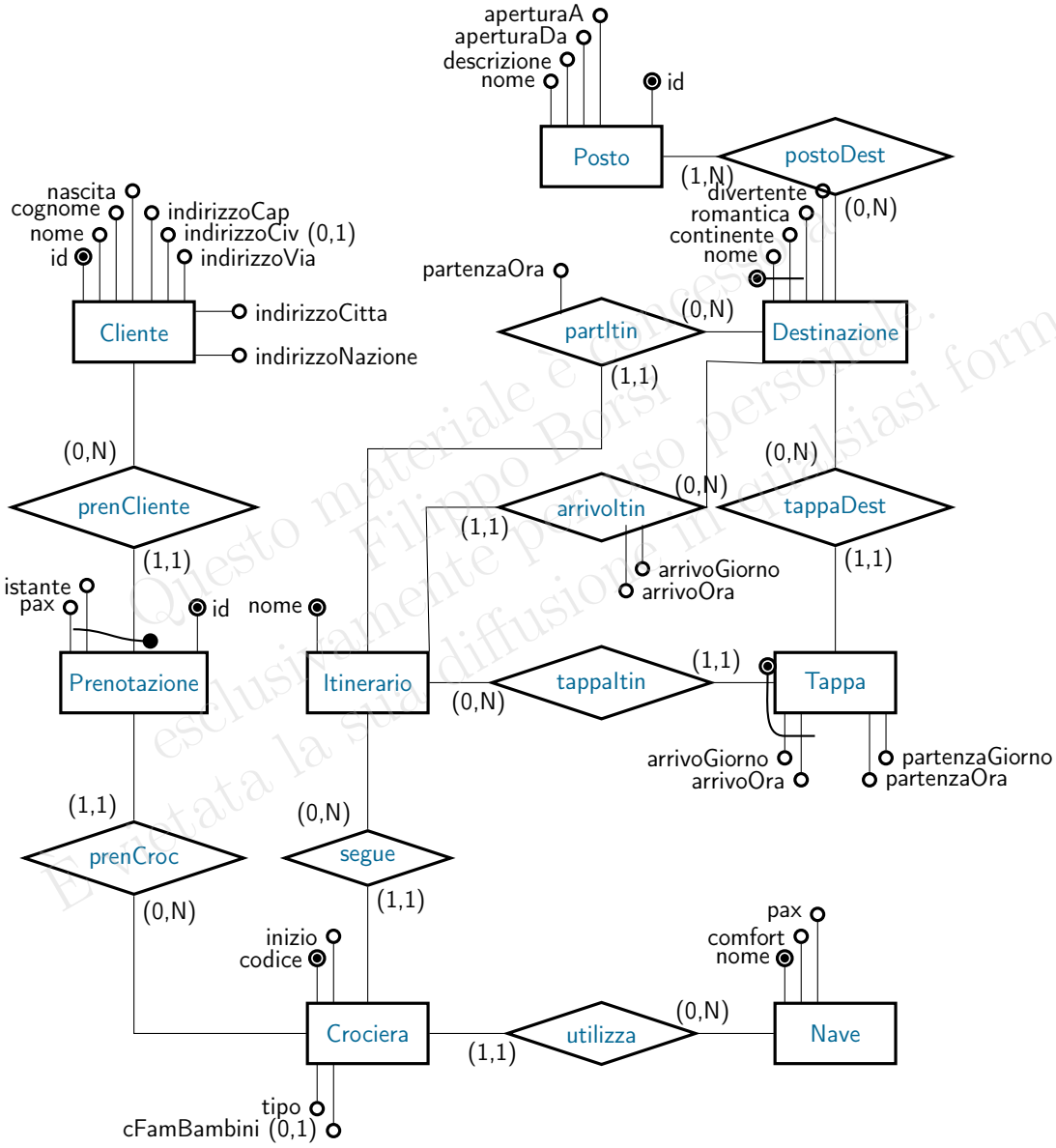
P.1.2 Soluzione

P.1.2.1 Scelta del DBMS

Si decide di utilizzare il DBMS PostgreSQL.

Nota: La scelta del DBMS è importante, in quanto può avere un impatto sul modo in cui vengono progettati i domini, i vincoli e le operazioni di use-case. In una fase di progetto completa, andrebbero prese anche altre decisioni, come l'architettura dell'applicazione ed il linguaggio di programmazione per quest'ultima. Tuttavia, dato lo scopo di questo corso, non prenderemo decisioni in tal senso.

P.1.2.2 Ristrutturazione del Diagramma ER e delle Specifiche dei Dati



Specifiche dei Dati

Entità Crociera

Ogni istanza di questa entità rappresenta una crociera (Req. 1.)

attributo	dominio	molteplicità	descrizione
codice	StringS		Il codice della crociera
inizio	date		La data di inizio della crociera
tipo	TipoCrociera		Il tipo della crociera (di luna di miele o per famiglia)
cFamBambini	boolean	(0,1)	definito se e solo se la crociera è per famiglia; true se la crociera per famiglia è adatta ai bambini, false altrimenti

Vincoli:

[V.Crociera.isa.bambini] Per ogni istanza dell'entità Crociera, il valore dell'attributo crocieraFamigliaBambini è definito se e solo se il valore dell'attributo tipo è 'PerFamiglia':

$$\forall c \text{ Crociera}(c) \rightarrow [\text{tipo}(c, \text{PerFamiglia}) \leftrightarrow \exists b \text{ cFamBambini}(c, b)]$$

[V.Crociera.posti] Il numero di posti prenotati per ogni crociera non può mai eccedere il numero di posti disponibili nella nave utilizzata.

Formalmente:

$$\forall \text{cr Crociera(cr)} \rightarrow \text{GestionePrenotazioni.postiDisponibili (cr)} \geq 0$$

Entità Nave

Ogni istanza di questa entità rappresenta una nave (Req. 2.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della nave
comfort	TipoComfort		Il grado di comfort della nave
pax	PosInteger		Il numero massimo di passeggeri per la nave

Vincoli:

[V.Nave.crociere.disj] Le crociere che utilizzano la stessa nave devono avere periodi tutti disgiunti.

Formalmente:

$$\forall c, c', n, c\_in, i\_c, i\_c', dest\_arr\_c, dest\_arr\_c', arr\_g\_c, arr\_g\_c'$$
$$[Crociera(c) \land Crociera(c') \land c \neq c' \land utilizza(c, n) \land utilizza(c', n) \land$$
$$inizio(c, c\_in) \land inizio(c', c'\_in) segue(c, i\_c) \land segue(c', i\_c') \land$$
$$arrivoItin(dest\_arr\_c, i\_c) \land arrivoItin(dest\_arr\_c', i\_c') \land$$
$$arrivoGiorno(dest\_arr\_c, i\_c, arr\_g\_c) \land arrivoGiorno(dest\_arr\_c', i\_c', arr\_g\_c')] \rightarrow$$
$$[c'\_in > c\_in + arr\_g\_c \lor c\_in > c'\_in + arr\_g\_c'].$$

Entità Itinerario
Ogni istanza di questa entità rappresenta un'itinerario (Req. 3.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome dell'itinerario

Entità Destinazione
Ogni istanza di questa entità rappresenta una destinazione toccata da itinerari (Req. 4.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della destinazione
continente	Continente		Il continente della destinazione
romantica	boolean		true se la destinazione è classificata come romantica, false altrimenti
divertente	boolean		true se la destinazione è classificata come divertente, false altrimenti

Entità Tappa
Ogni istanza di questa entità rappresenta una tappa toccata da un itinerario (Req. 3.2.), escluse quelle di partenza e arrivo dell'itinerario

attributo	dominio	molteplicità	descrizione
arrivoGiorno	PosInteger		La data (relativa alla data di partenza della crociera) di arrivo alla destinazione
arrivoOra	time		L'ora di arrivo alla destinazione
partenzaGiorno	PosInteger		La data (relativa alla data di partenza della crociera) di partenza dalla destinazione
partenzaOra	time		L'ora di partenza dalla destinazione

Vincoli:

[V.Tappa.date] Per ogni istanza di questa entità, l'istante di partenza (codificato dai valori della coppia di attributi partenzaGiorno e partenzaOra) deve essere successivo a quello di arrivo (codificato dai valori della coppia di attributi arrivoGiorno e arrivoOra):

$$\forall t, p\_g, p\_h, a\_g, a\_h \; [\mathsf{Tappa}(t) \land \mathsf{partenzaGiorno}(t, p\_g) \land \mathsf{partenzaOra}(t, p\_h) \land \mathsf{arrivoGiorno}(t, a\_g) \land \mathsf{arrivoOra}(t, a\_h)] \rightarrow ((a\_g < p\_g) \lor (a\_g = p\_g \land a\_h < p\_h))$$

[V.Tappa.succPart] Le istanze di questa entità con giorno di arrivo pari ad 1 devono avere valore per l'ora di arrivo successivo all'orario di partenza del relativo itinerario. Formalmente:

$$\forall i, p, t, a_t, o_t, o_p \; \mathsf{Itinerario}(i) \land \mathsf{partItin}(p, i) \land \mathsf{partenzaOra}(p, i, o_p) \land \mathsf{Tappa}(t) \land \mathsf{tappaItin}(t, i) \land \mathsf{arrivoGiorno}(t, 1) \land \mathsf{arrivoOra}(t, o_t) \rightarrow o_p < o_t$$

[V.Tappa.primaDiArrivo] Le istanze di questa entità devono avere valore per l'istante di partenza (codificato dai valori della coppia di attributi partenzaGiorno e partenzaOra) precedente al valore di arrivo del relativo itinerario alla destinazione

finale. Formalmente:

$$\forall i, dest\_arr, dest\_arr\_g, dest\_arr\_h, t, t\_part\_g, t\_part\_h$$
$$Itinerario(i) \wedge arrivoItin(dest\_arr, i) \wedge$$
$$arrivoGiorno(dest\_arr, i, dest\_arr\_g) \wedge arrivoOra(dest\_arr, i, dest\_arr\_h) \wedge$$
$$Tappa(t) \wedge tappaItin(t, i) \wedge$$
$$partenzaGiorno(t, t\_part\_g) \wedge partenzaOra(t, t\_part\_h) \rightarrow$$
$$[(t\_part\_g < dest\_arr\_g) \vee$$
$$(t\_part\_g = dest\_arr\_g \wedge t\_part\_h < dest\_arr\_h)]$$

Entità Posto
Ogni istanza di questa entità rappresenta un posto da vedere (Req. 5.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome del posto da vedere
descrizione	StringL		La descrizione del posto da vedere
aperturaDa	time		L'orario di apertura del posto da vedere
aperturaA	time		L'orario di chiusura del posto da vedere
id	integer		L'identificatore univoco del posto

Entità Cliente
Ogni istanza di questa entità rappresenta un cliente (Req. 6.)

attributo	dominio	molteplicità	descrizione
id	integer		L'identificatore univoco del cliente
nome	StringM		Il nome del cliente
cognome	StringM		Il cognome del cliente
nascita	date		La data di nascita del cliente
indirizzoVia	StringM		Il campo 'via' dell'indirizzo del cliente
indirizzoCiv	PosInteger	(0,1)	Il campo 'civico' dell'indirizzo del cliente
indirizzoCap	Cap		Il campo 'cap' dell'indirizzo del cliente
indirizzoCitta	StringM		Il campo 'citta' dell'indirizzo del cliente
indirizzoNazione	StringM		Il campo 'nazione' dell'indirizzo del cliente

Entità Prenotazione
Ogni istanza di questa entità rappresenta una prenotazione di una crociera da parte di un cliente (Req. 7.)

attributo	dominio	molteplicità	descrizione
pax	PosInteger		Il numero di posti della prenotazione
istante	timestamp		L'instante nel quale è stata effettuata la prenotazione
id	integer		L'identificatore univoco della prenotazione

Vincoli:

[V.Prenotazione.istante] Le prenotazioni devono essere effettuate prima del giorno di inizio della relativa crociera.

Formalmente:

$$\forall p,i,cr,cr\_i \quad Prenotazione(p) \wedge istante(p,i) \wedge prenCroc(p,cr) \wedge Crociera(cr) \wedge inizio(cr,cr\_i) \rightarrow i < cr\_i.$$

Relationship utilizza
Ogni istanza di questa relationship lega una crociera alla nave utilizzata (Req. 1.4.)
Attributi: Nessuno

Relationship segue
Ogni istanza di questa relationship lega una crociera al suo itinerario (Req. 1.5.)
Attributi: Nessuno

Relationship tappaItin
Ogni istanza di questa relationship lega un itinerario ad una sua tappa (Req. 3.2.)
Attributi: Nessuno

Vincoli:

[V.tappaItin.soste.disj] I periodi di sosta di ogni itinerario nelle diverse tappe sono tutti disgiunti. Formalmente:

$$
\begin{aligned}
&\forall i, t, t', \text{t\_part\_g}, \text{t\_part\_h}, \text{t\_arr\_g}, \text{t\_arr\_h}, \text{t'\_part\_g}, \text{t'\_part\_h}, \\
&\quad \text{t'\_arr\_g}, \text{t'\_arr\_h} \\
&\quad \text{Itinerario}(i) \wedge \text{Tappa}(t) \wedge \text{tappaItin}(t, i) \wedge \\
&\quad \text{arrivoGiorno}(t, \text{t\_arr\_g}) \wedge \text{arrivoOra}(t, \text{t\_arr\_h}) \wedge \\
&\quad \text{partenzaGiorno}(t, \text{t\_part\_g}) \wedge \text{partenzaOra}(t, \text{t\_part\_h}) \wedge \\
&\quad \text{Tappa}(t') \wedge \text{tappaItin}(t', i) \wedge \\
&\quad \text{arrivoGiorno}(t, \text{t\_arr\_g}) \wedge \text{arrivoOra}(t, \text{t\_arr\_h}) \wedge \\
&\quad \text{partenzaGiorno}(t, \text{t\_part\_g}) \wedge \text{partenzaOra}(t, \text{t\_part\_h}) \wedge \\
&\quad t \neq t' \rightarrow \\
&\qquad \left[\begin{array}{l} \left(\begin{array}{l} (\text{t'\_arr\_g} > \text{t\_part\_g}) \vee \\ (\text{t'\_arr\_g} = \text{t\_part\_g} \wedge (\text{t'\_arr\_h} > \text{t\_part\_h})) \end{array} \right) \\ \vee \\ \left(\begin{array}{l} (\text{t\_arr\_g} > \text{t'\_part\_g}) \vee \\ (\text{t\_arr\_g} = \text{t'\_part\_g} \wedge (\text{t\_arr\_h} > \text{t'\_part\_h})) \end{array} \right) \end{array} \right]
\end{aligned}
$$

Relationship tappaDest
Ogni istanza di questa relationship lega una tappa di un itinerario alla relativa destinazione (Req. 3.2.)
Attributi: Nessuno

Relationship partItin
Ogni istanza di questa relationship lega un itinerario alla relativa destinazione di partenza (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
partenzaOra	time		L'ora di partenza della crociera

Relationship arrivoItin
Ogni istanza di questa relationship lega un itinerario alla relativa destinazione di arrivo (Req. 3.2.)

attributo	dominio	molteplicità	descrizione
arrivoGiorno	PosInteger		La data (in termini di giorni dalla data di partenza dell'itinerario) di arrivo della crociera alla tappa
arrivoOra	time		L'ora di arrivo della crociera alla tappa

Relationship postoDest
Ogni istanza di questa relationship lega una destinazione ad un posto da vedere (Req. 4.3.)
Attributi: Nessuno

Relationship prenCliente
Ogni istanza di questa relationship lega una prenotazione al relativo cliente (Req. 7.1.)
Attributi: Nessuno

Relationship prenCroc
Ogni istanza di questa relationship lega una prenotazione alla relativa crociera (Req. 7.3.)
Attributi: Nessuno

Dominio PosInteger
Il dominio è dato dal sottoinsieme del dominio integer formato dai valori > 0.

Dominio TipoComfort
Il dominio è dato dal sottoinsieme del dominio integer formato dai valori tra 3 e 5.

Dominio Cap
Il dominio è dato dall'insieme dalle stringhe di esattamente 5 caratteri numerici.

Dominio Continente
Il dominio è dato dall'insieme dei seguenti elementi:

$$\{\text{E}, \text{AS}, \text{AMN}, \text{AMS}, \text{AF}, \text{O}\}$$

(nell'ordine: Europa, Asia, America del Nord, America del Sud, Africa, Oceania)

Dominio TipoCrociera
Il dominio è dato dall'insieme dei seguenti elementi:

$$\{\text{LunaDiMiele}, \text{PerFamiglia}\}$$

Dominio StringS
Il dominio è dato dall'insieme delle stringhe di al più 20 caratteri.

Dominio StringM
Il dominio è dato dall'insieme delle stringhe di al più 100 caratteri.

Dominio StringL
Il dominio è dato dall'insieme delle stringhe di al più 1000 caratteri.

P.2 Schema Relazionale della Base Dati

P.2.1 Testo

Proseguire la fase di progettazione logica della base di dati producendo lo schema relazionale della base dati e i relativi vincoli (vincoli di chiave, di chiave primaria, di foreign key, di inclusione, di ennupla, di dominio) a partire dallo schema ER ristrutturato.

P.2.2 Soluzione

P.2.2.1 Definizione delle Relazioni Derivanti da Entità e Relationship Accorpate ad Entità

Crociera (codice:StringS, inizio:date, tipo:TipoCrociera, cFamBambini*:boolean, nave:StringM, itinerario:StringM)
La relazione accorpa le relationship utilizza e segue
[VincoloDB.1] *foreign key:* nave references Nave(nome)
[VincoloDB.2] *foreign key:* itinerario references Itinerario(nome)
[VincoloDB.3] *ennupla:* tipo $=$ PerFamiglia $\leftrightarrow$ crocieraFamigliaBambini $\neq$ NULL (implementa [V.Crociera.isa.bambini])

Nave (nome:StringM, comfort:TipoComfort, pax:PosInteger)

Destinazione (nome:StringM, continente:Continente, romantica:boolean, divertente:boolean)

Itinerario (nome:StringM, partenzaItinNome:StringM, partenzaItinCont:Continente, partenzaOra:time, arrivoItinNome:StringM, arrivoItinCont:Continente, arrivoGiorno:PosInteger, arrivoOra:time)
La relazione accorpa le relationship partItin e arrivoItin.
[VincoloDB.4] *foreign key:* (partenzaItinNome, partenzaItinCont) references Destinazione(nome, continente)
[VincoloDB.5] *foreign key:* (arrivoItinNome, arrivoItinCont) references Destinazione(nome, continente)

Tappa (itinerario:StringM, arrivoGiorno:PosInteger, arrivoOra:time, partenzaGiorno:PosInteger, partenzaOra:time, destinazioneNome:StringM, destinazioneContinente:Continente)
La relazione accorpa le relationship tappaItin e tappaDest.
[VincoloDB.6] *foreign key:* itinerario references Itinerario(nome)
[VincoloDB.7] *foreign key:* (destinazioneNome, destinazioneContinente) references Destinazione(nome, continente)
[VincoloDB.8] *ennupla:* arrivoGiorno $<$ partenzaGiorno $\vee$ (arrivoGiorno $=$ partenzaGiorno $\wedge$ arrivoOra $<$ partenzaOra) (implementa [V.Tappa.date])

Posto (id:integer, nome:StringM, descrizione:StringL, aperturaDa:time, aperturaA:time)
[VincoloDB.9] *inclusione:* id $\subseteq$ postoDest(posto)
[VincoloDB.10] *seriale:* i valori della colonna id sono generati automaticamente dal DBMS

Cliente (id:integer, nome:StringM, cognome:StringM, nascita:date, indirizzoVia:StringM, indirizzoCiv*:PosInteger, indirizzoCap:Cap, indirizzoCitta:StringM, indirizzoNazione:StringM)
[VincoloDB.11] *seriale:* i valori della colonna id sono generati automaticamente dal DBMS

Prenotazione (id:integer, cliente:integer, crociera:StringS, pax:PosInteger, istante:timestamp)
La relazione accorpa le relationship prenCliente e prenCroc.

[VincoloDB.12] *foreign key:* cliente references Cliente(id)
[VincoloDB.13] *chiave:* (cliente, istante)
[VincoloDB.14] *foreign key:* crociera references Crociera(codice)
[VincoloDB.15] *seriale:* i valori della colonna id sono generati automaticamente dal DBMS

P.2.2.2 Definizione delle Relazioni Derivanti da Relationship non Accorpate ad Entità

postoDest (posto:integer, destinazioneNome:StringM, destinazioneContinente:Continente)
[VincoloDB.16] *foreign key:* posto references Posto(id)
[VincoloDB.17] *foreign key:* (destinazioneNome, destinazioneContinente) references Destinazione(nome, continente)

P.3 Progettazione dei Vincoli Esterni

P.3.1 Testo

Proseguire la fase di progettazione logica della base di dati progettando come imporre i vincoli di integrità sui dati non esprimibili come vincoli di chiave, di chiave primaria, di foreign key, di inclusione, di ennupla, di dominio.

P.3.2 Soluzione

Vincolo V.Crociera.posti

Si definisce la seguente funzione nel DBMS. La funzione sarà utilizzata per verificare il vincolo e sarà invocata dall'operazione di use-case GestionePrenotazioni.prenota ().

DB.postiDisponibili(cr : StringS) : integer

algoritmo:

1 $Q \leftarrow$ risultato della query SQL ottenuta dallo scheletro seguente sostituendo a ':cr' il valore del parametro attuale *cr*:

```
select n.pax - x.postiPrenotati as postiLiberi
from Crociera c, Nave n,
  ( select sum(z.pax) as postiPrenotati
    from Prenotazione z
    where z.crociera = :cr
  ) x
where c.codice = :cr and c.nave = n.nome
```

2 **if** $Q == NULL$ **then**
3 generare l'errore 'Crociera non trovata';
4 **else**
5 **return** *il valore della colonna 'postiLiberi' dell'unica ennupla in Q*;

[VincoloDB.18] *trigger:*

Operazioni: modifica di una ennupla nella relazione Nave

Istante di Invocazione: dopo l'operazione intercettata

Funzione:

1 *old* $\leftarrow$ l'ennupla prima della modifica;
2 *new* $\leftarrow$ l'ennupla risultato della modifica;
3 *isError* $\leftarrow$ false;
4 **if** *new.pax* $<$ *old.pax* **then**
5 *isError* $\leftarrow$

```
exists (
  select *
  from Crociera cr
  where nave = new.nome
    and DB.postiDisponibili(cr.codice) < 0
)
```

6 **if** *isError* **then** blocca l'operazione;
7 **else** permetti l'operazione;

[VincoloDB.19] *trigger:*

Operazioni: modifica di una ennupla nella relazione Crociera

Istante di Invocazione: dopo l'operazione intercettata

Funzione:

```
1 old ← l'ennupla prima della modifica;
2 new ← l'ennupla risultato della modifica;
3 isError ← false;
4 if new.nave ≠ old.nave then
5   isError ← DB.postiDisponibili(new.codice) < 0;
6 if isError then blocca l'operazione;
7 else permetti l'operazione;
```

[VincoloDB.20] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Prenotazione

Istante di Invocazione: dopo l'operazione intercettata

Funzione:

```
1 new ← l'ennupla inserita o risultato della modifica;
2 isError ← DB.postiDisponibili(new.codice) < 0;
3 if isError then blocca l'operazione;
4 else permetti l'operazione;
```


Vincolo V.Nave.crociere.disj

[VincoloDB.21] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Crociera

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *new* ← l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
  select *
  from Crociera cr, Itinerario cr_i, Itinerario new_i
  where cr.nave = new.nave
    and cr.itinerario = cr_i.nome
    and new.itinerario = new_i.nome
    and cr.inizio <= new.inizio + new_i.arrivoGiorno
    and cr.inizio + cr_i.arrivoGiorno >= new.inizio
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

Vincolo V.Tappa.succPart

[VincoloDB.22] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Tappa

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *isError* ← false;
2 *new* ← l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica;
3 **if** *new.arrivoGiorno* == *1* **then**
4 *isError* ←

```
exists (
  select *
  from Itinerario i
  where nome = new.itinerario
    and i.partenzaOra >= new.arrivoOra
);
```

5 **if** *isError* **then** blocca l'operazione;
6 **else** permetti l'operazione;

[VincoloDB.23] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Itinerario

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *new* ← l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
  select *
  from Tappa t
  where t.itinerario = new.nome
    and t.arrivoGiorno = 1
    and t.arrivoOra < new.partenzaOra
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

Vincolo V.Tappa.primaDiArrivo

[VincoloDB.24] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Tappa

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *new* ← l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
  select *
  from Itinerario i
  where nome = new.itinerario
    and (
      (i.arrivoGiorno < new.partenzaGiorno)
      or
      (i.arrivoGiorno = new.partenzaGiorno and
        i.arrivoOra < new.partenzaOra)
    )
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

[VincoloDB.25] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Itinerario

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *new* ← l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
  select *
  from Tappa t
  where t.itinerario = new.nome
    and (
      (t.partenzaGiorno > new.arrivoGiorno)
      or
      (t.partenzaGiorno = new.arrivoGiorno
        and t.partenzaOra >= new.arrivoOra)
    )
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

Vincolo V.tappaltin.soste.disj

[VincoloDB.26] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Tappa

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *new* ← l'ennupla che si sta inserendo oppure l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
-- esiste gia' una tappa t dello stesso itinerario di new
select *
from Tappa t
where new.itinerario = t.itinerario
-- tale che:
  and (
    -- [1.] si prevede di arrivare in new durante la sosta gia'
       prevista in t,
    -- ovvero:
    -- [1.a] si arriva a new dopo l'arrivo in t
    (new.arrivoGiorno > t.arrivoGiorno or
      (new.arrivoGiorno = t.arrivoGiorno and t.arrivoOra > t.
         arrivoOra)
    )
    and
    -- [1.b] e si arriva a new prima della ripartenza da t
    (new.arrivoGiorno < t.partenzaGiorno or
      (new.arrivoGiorno = t.partenzaGiorno and new.arrivoOra < t
         .partenzaOra)
    )
  -- oppure
  ) or (
    -- [2.] si prevede di ripartire da new durante la sosta gia'
       prevista in t,
    -- ovvero:
    -- [2.a] si parte da new dopo l'arrivo in t
    (new.partenzaGiorno > t.arrivoGiorno or
      (new.partenzaGiorno = t.arrivoGiorno and new.partenzaOra >
         t.arrivoOra)
    )
    and
    -- [2.b] e si parte da new prima della ripartenza da t
    (new.partenzaGiorno < t.partenzaGiorno or
      (new.partenzaGiorno = t.partenzaGiorno and new.partenzaOra
         < t.partenzaOra)
    )
  )
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

Vincolo V.Prenotazione.istante

[VincoloDB.27] *trigger:*

Operazioni: inserimento o modifica di una ennupla nella relazione Prenotazione

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 *new* ← l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
  select *
  from Crociera cr
  where cr.codice = new.crociera
    and new.istante > cr.inizio
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

[VincoloDB.28] *trigger:*

Operazioni: modifica di una ennupla nella relazione Crociera

Istante di Invocazione: prima l'operazione intercettata

Funzione:

1 *new* ← l'ennupla risultato della modifica;
2 *isError* ←

```
exists (
  select *
  from Prenotazione p
  where p.crociera = new.codice
    and p.istante > new.inizio
)
```

3 **if** *isError* **then** blocca l'operazione;
4 **else** permetti l'operazione;

P.4 Specifiche Realizzative degli Use-Case

P.4.1 Testo

Proseguire la fase di progettazione dell'applicazione producendo le specifiche realizzative delle operazioni di use-case.

P.4.2 Soluzione

Specifica Use-Case GestionePrenotazioni

- **prenota(*cl* : integer, *cr* : StringS, *posti*: PosInteger) : integer (Req. 8.1.)**

  algoritmo:

  1 **if** *GestionePrenotazioni.postiDisponibili(cr) restituisce un errore* **then**
  2 inoltra l'errore;
  3 **else**
  4 **if** *GestionePrenotazioni.postiDisponibili(cr)* $<$ *posti* **then**
  5 termina con l'errore 'Posti disponibili insufficienti';
  6 *I* $\leftarrow$ risultato del comando SQL ottenuto dallo scheletro seguente sostituendo a ':cl', ':cr' e ':posti' i valori, rispettivamente dei parametri attuali *cl*, *cr*, *posti*:

  ```
  insert into Prenotazione(cliente, crociera, pax, istante)
  values (:cl, :cr, :posti, CURRENT_TIMESTAMP)
  returning id
  ```

  7 **if** *I rappresenta un errore* **then**
  8 inoltra l'errore;
  9 **else**
  10 **return** *il valore della colonna 'id' dall'unica ennupla di I*;

  / Nota: l'opzione returning id, che permette ad un comando* ***insert*** *di restituire uno o più valori della ennupla inserita, è una estensione di PostgreSQL e* non *è standard SQL (come del resto la possibilità di definire sequenze). */*

- **postiDisponibili(*cr* : StringS) : integer**

  algoritmo:

  1 *Q* $\leftarrow$ risultato della query SQL ottenuta dallo scheletro seguente sostituendo a ':cr' il valore del parametro attuale *cr*:

  ```
  select DB.postiDisponibili(:cr) as postiLiberi
  ```

  2 **if** *Q rappresenta un errore* **then**
  3 inoltra l'errore;
  4 **else**
  5 **return** *il valore della colonna 'postiLiberi' dell'unica ennupla in Q*;

Specifica Use-Case GestioneClienti

- **nuovo(*no* : StringM, *c* : StringM, *na*: date, *indVia*:StringM, *indCiv**:PosInteger, *indCap*:Cap, *indCitta*:StringM, *indNazione*:StringM) : integer**

  algoritmo:

  1 *I* ← risultato del comando SQL ottenuto dallo scheletro seguente sostituendo ad ogni segnaposto denotato da ':' il valore dell'omonimo parametro attuale:

  ```
  insert into Cliente(nome, cognome, nascita,
                      indirizzoVia, indirizzoCiv, indirizzoCap,
                      indirizzoCitta, indirizzoNazione)
  values (:no, :co, :na, :indVia, :indCiv, :indCap, :indCitta, :
      indNazione)
  returning id
  ```

  2 **return** *il valore della colonna 'id' dalla unica ennupla in I*;

- **cerca(*no** : StringM, *co** : StringM, *na**: date) : Collezione(⟨id, nome, cognome, nascita, indirizzoVia, indirizzoCiv, indirizzoCap, indirizzoCitta, indirizzoNazione⟩)**

  algoritmo:

  1 *PHI_nome* ← 'true' se *no* = NULL e 'nome = :no' altrimenti;
  2 *PHI_cognome* ← 'true' se *co* = NULL e 'cognome = :co' altrimenti;
  3 *PHI_nasc* ← 'true' se *na* = NULL e 'nascita = :na' altrimenti;
  4 *Q* ← risultato della query SQL ottenuto dallo scheletro seguente sostituendo ad ogni segnaposto denotato da ':' il valore dell'omonima variabile locale o parametro attuale:

  ```
  select id, nome, cognome, nascita, indirizzoVia, indirizzoCiv,
         indirizzoCap, indirizzoCitta, indirizzoNazione
  from Cliente
  where :PHI_nome and :PHI_cognome and :PHI_nascita
  ```

  5 **return** *Q*;

Specifica Use-Case OperazioniCrociere

- **tipoCrociera(c : StringS) : {tradizionale, alternativa} (Req. 1.6.1.)**

  algoritmo:

  1 $Q \leftarrow$ risultato della query SQL ottenuto dallo scheletro seguente sostituendo ad ogni segnaposto denotato da ':' il valore dell'omonimo parametro attuale:

```
select
  sum(case when d.romantica then 1 else 0 end) -
    sum(case when d.divertente then 1 else 0 end) as delta

from
   Crociera c, Tappa t, Destinazione d

where c.codice = :cod
  and c.tipo = 'LunaDiMiele'
  and c.itinerario = t.itinerario
  and t.destinazioneNome = d.nome
  and t.destinazioneContinente = d.continente
```

  2 **if** Q = *NULL* **then**
  3 genera l'errore 'La crociera c non esiste oppure non è una crociera di Luna di miele';
  4 **else**
  5 $t \leftarrow$ l'unica ennupla di Q;
  6 **if** *t.delta >= 0* **then return** *tradizionale* ;
  7 **else return** *alternativa* ;

Specifica Use-Case StatisticheClienti

- **etaMediaEsotica(*C* : Insieme(integer)) : real (Req. 8.2.)**

algoritmo:

1 Creare la tabella temporanea (visibile solo all'invocazione; corrente di questa operazione):

TmpClienti(<u>cliente</u>)
Vincolo *foreign key:* cliente references Cliente(id)

2 $I \leftarrow$ risultato del seguente comando SQL dopo aver sostituito ai segnaposti (...) gli elementi $c \in C$:

```
insert into TmpClienti(cliente)
  values (...),
  (...),
  ...
```

3 **if** *I rappresenta un errore di foreign key per un certo* $c \in C$ **then**
4 genera l'errore 'il cliente con ID *c* non esiste';
5 **return**;
6 $Q \leftarrow$

```
select avg(floor((CURRENT_DATE - cl.nascita)/365)) as etaMedia
from ( select distinct pr.cliente as id
       from Prenotazione pr, Crociera cr, Tappa t,
            TmpClienti tmp
       where pr.cliente = tmp.cliente
         and pr.crociera = cr.codice
         and cr.itinerario = t.itinerario
         and DB.isDestinazioneEsotica(t.destinazioneNome, t.
             destinazioneContinente) )
             codice_cliente,
     Cliente cl
where codice_cliente.id = cl.id
```

7 Eliminare la tabella temporanea TmpClienti(cliente);
8 **if** *Q rappresenta un errore* **then**
9 inoltra l'errore;
10 **else**
11 **return** *Q*;

- **eta(*cl* : integer) : integer**

algoritmo:

1 $Q \leftarrow$ risultato della query SQL ottenuto dallo scheletro seguente sostituendo a ':cl' il valore del parametro attuale *cl*:

```
select DB.etaCliente(:cl)
```

2 **if** *Q rappresenta un errore* **then**
3 inoltra l'errore';
4 **else**
5 **return** *il valore della colonna 'eta' dell'unica ennupla di Q*;

Specifica Use-Case StatisticheDestinazioni

- **esotica(*nome* : StringM, *continente*: Continente) : boolean**

  algoritmo:

  1 $Q \leftarrow$ risultato della query SQL otteuta dallo scheletro seguente dopo aver sostituito i segnaposti denotati da ':' con i valori degli omonimi parametri attuali:

  ```
  select DB.isDestinazioneEsotica(:nome, :continente)
  ```

  2 **if** *Q rappresenta un errore* **then**
  3 inoltra l'errore;
  4 **else**
  5 **return** *il valore booleano della componente 'esotica' dell'unica ennupla in Q*;

- **gettonata(*nome* : StringM, *continente*: Continente) : boolean**

  algoritmo:

  1 $Q \leftarrow$ risultato della query SQL otteuta dallo scheletro seguente dopo aver sostituito i segnaposti denotati da ':' con i valori degli omonimi parametri attuali:

  ```
  select DB.isDestinazioneGettonata(:nome, :continente)
  ```

  2 **return** *Q*;

- **percentualeGettonate(*D*: Insieme(⟨nome: StringM, cont: Continente⟩)): real (Req. 8.3.)**

  algoritmo:

1 **if** $|D| = 0$ **then**
2 genera l'errore 'L'insieme di destinazioni in input è vuoto';
3 Crea la tabella temporanea (visibile solo all'invocazione corrente di questa operazione):

TmpDest(<u>nome</u>, <u>continente</u>)
Vincolo *foreign key:* (nome, continente) references Destinazione(nome, continente)

4 $I \leftarrow$ risultato del comando SQL ottenuto dallo scheletro seguente dopo aver sostituito ai segnaposti $(\ldots)$ gli elementi $d \in D$:

```
insert into TmpDest(nome, continente)
  values (...),
  (...),
  ...
```

5 **if** *I rappresenta un errore di foreign key per un certo* $d = (n, c) \in D$ **then**
6 genera l'errore 'la destinazione di nome *n* nel continente *c* non esiste';
7 $Q \leftarrow$ risultato della seguente query SQL:

```
select sum(WHEN
       DB.isDestinazioneGettonata(t.nome, t.continente) THEN 1
       ELSE 0 END)/count(*) as result
from TmpDest t
```

8 elimina la tabella temporanea TmpDest;
9 **return** *il valore della colonna 'result' dell'unica ennupla di Q*;

Si definiscono le seguenti funzioni nel DBMS:

DB.etaCliente(*cl* : integer) : integer

algoritmo:

1 $Q \leftarrow$ risultato della query SQL ottenuto dallo scheletro seguente sostituendo a ':cl' il valore del parametro attuale *cl*:

```
select floor( (CURRENT_DATE - nascita)/365) as eta
from Cliente
where id = :cl
```

2 **if** *Q è l'insieme vuoto* **then**
3 genera l'errore 'Cliente con id *cl* non trovato';
4 **else**
5 **return** *il valore della colonna 'eta' dell'unica ennupla di Q*;

DB.isDestinazioneEsotica(*nome* : StringM, *continente*: Continente) : boolean

algoritmo:

1 $Q \leftarrow$ risultato della query SQL otteuta dallo scheletro seguente dopo aver sostituito i segnaposti denotati da ':' con i valori degli omonimi parametri attuali:

```
select (continente <> 'E') as esotica
from Destinazione
where nome = :nome and continente = :continente
```

2 **if** *Q è l'insieme vuoto* **then**
3 genera l'errore 'Destinazione non trovata';
4 **else**
5 **return** *il valore booleano della componente 'esotica' dell'unica ennupla in Q*;

DB.isDestinazioneGettonata(*nome* : StringM, *continente*: Continente) : boolean

algoritmo:

1 $Q \leftarrow$ risultato della query SQL otteuta dallo scheletro seguente dopo aver sostituito i segnaposti denotati da ':' con i valori degli omonimi parametri attuali:

```
select cr.tipo as tipoCrociere, count(*) as numero
from Crociera cr, Tappa t
where cr.itinerario = t.itinerario
  and t.destinazioneNome = :nome
  and t.destinazioneContinente = :continente
  and (CURRENT_DATE - cr.inizio) <= 365*2
group by cr.tipo
```

// La query Q restituisce al più due ennuple, una per tipo di crociera (valori della colonna tipoCrociere pari a 'LunaDiMiele' e 'PerFamiglia'). Il valore della colonna 'numero' di ognuna delle ennuple è il numero di crociere di quel tipo che hanno toccato la destinazione data negli ultimi due anni. Se l'ennupla per un certo tipo di crociera non è restituita, allora tale numero si intende essere pari a 0.

2 **if** *Q contiene due ennuple* **then**

3 *ldm* $\leftarrow$ il valore della colonna numero della ennupla con valore 'LunaDiMiele' per la colonna tipoCrociere;

4 *fam* $\leftarrow$ il valore della colonna numero della ennupla con valore 'PerFamiglia' per la colonna tipoCrociere;

5 **return** *ldm* ≥ 10 *and fam* ≥ 15;

6 **else return** *false* ;