



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

Indice	1
1 Introduzione	3
2 Specifica dei Requisiti	5
A Analisi Concettuale	7
A.1 Raffinamento dei Requisiti	9
A.1.1 Testo	9
A.1.2 Soluzione	10
A.2 Diagramma ER	13
A.2.1 Testo	13
A.2.2 Soluzione	14
A.3 Estensione del Diagramma ER	19
A.3.1 Testo	19
A.3.2 Soluzione	20
A.4 Estensione del Diagramma ER	27
A.4.1 Testo	27
A.4.2 Soluzione	28
A.5 Identificazione delle Entità e Definizione Informale di Vincoli Esterni	35
A.5.1 Testo	35
A.5.2 Soluzione	36

A.6	Diagramma UML degli Use-Case	45
A.6.1	Testo	45
A.6.2	Soluzione	46
A.7	Specifiche Concettuali Informali degli Use-Case	47
A.7.1	Testo	47
A.7.2	Soluzione	48
A.8	Vincoli Esterni	55
A.8.1	Testo	55
A.8.2	Soluzione	56
A.9	Specifiche Concettuali degli Use-Case	65
A.9.1	Testo	65
A.9.2	Soluzione	66
P	Progettazione della Base Dati e delle Funzionalità	73
P.1	Scelta del DBMS e Ristrutturazione del Diagramma ER e delle Specifiche dei Dati	75
P.1.1	Testo	75
P.1.2	Soluzione	76
	P.1.2.1 Scelta del DBMS	76
	P.1.2.2 Ristrutturazione del Diagramma ER e delle Specifiche dei Dati . .	77
P.2	Schema Relazionale della Base Dati	87
P.2.1	Testo	87
P.2.2	Soluzione	88
	P.2.2.1 Definizione delle Relazioni Derivanti da Entità e Relationship Accorpate ad Entità	88
	P.2.2.2 Definizione delle Relazioni Derivanti da Relationship non Accor- pate ad Entità	89
P.3	Progettazione dei Vincoli Esterni	91
P.3.1	Testo	91
P.3.2	Soluzione	92

1

Introduzione

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

Questo materiale è concesso a
Filippo Boni
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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.

Questo materiale è riservato a
Filippo Bonaccorsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

Parte A

Analisi Concettuale

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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.

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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.).



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

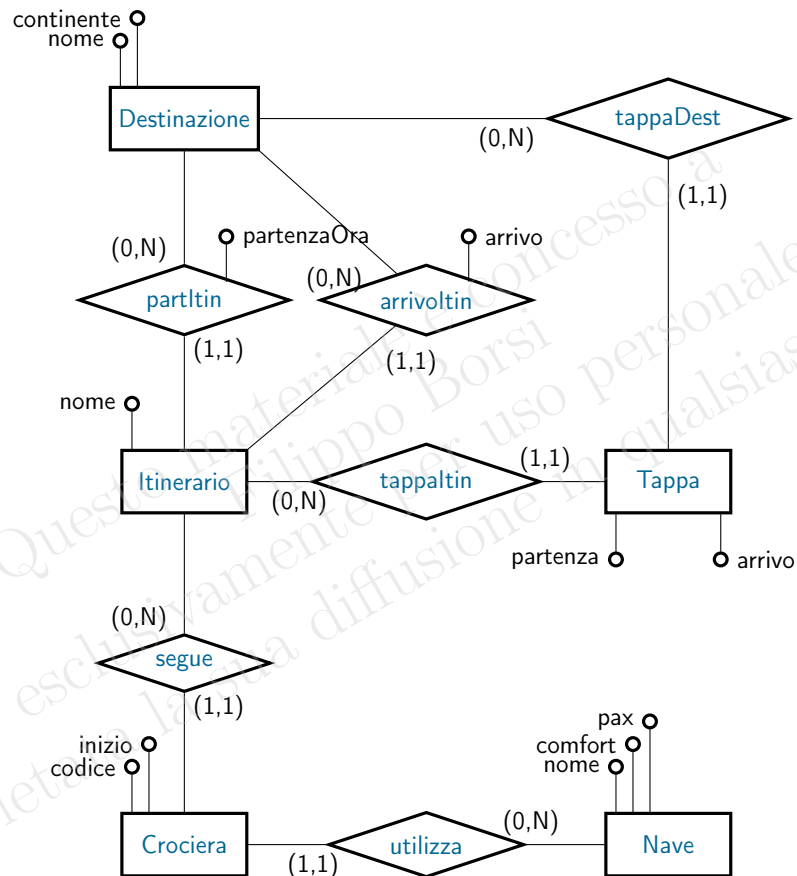
A.2

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 **tappaltin**

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 [arrivoltin](#)

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

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

A.3

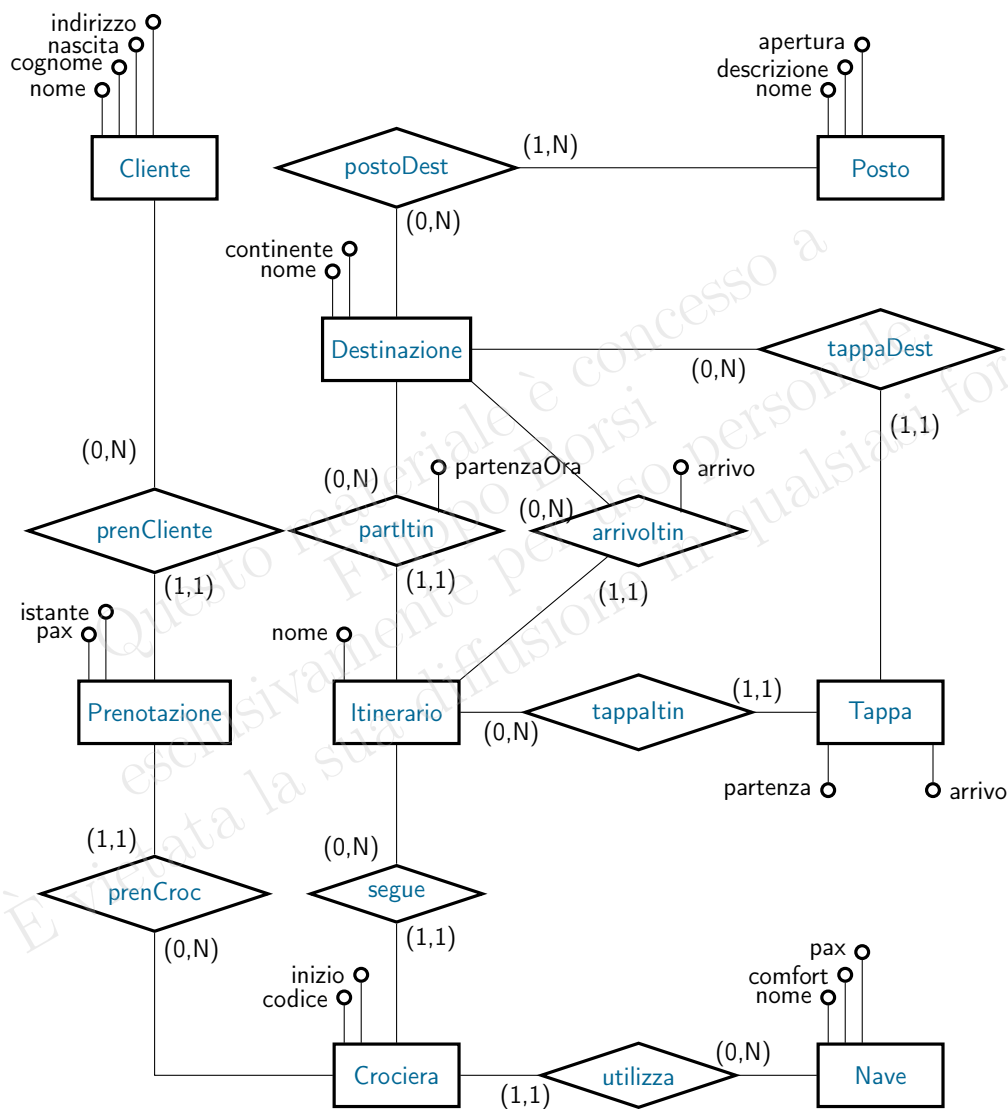
Estensione del Diagramma ER

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.

Questo materiale è concesso a
Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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'istante 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 [arrivoltin](#)

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

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

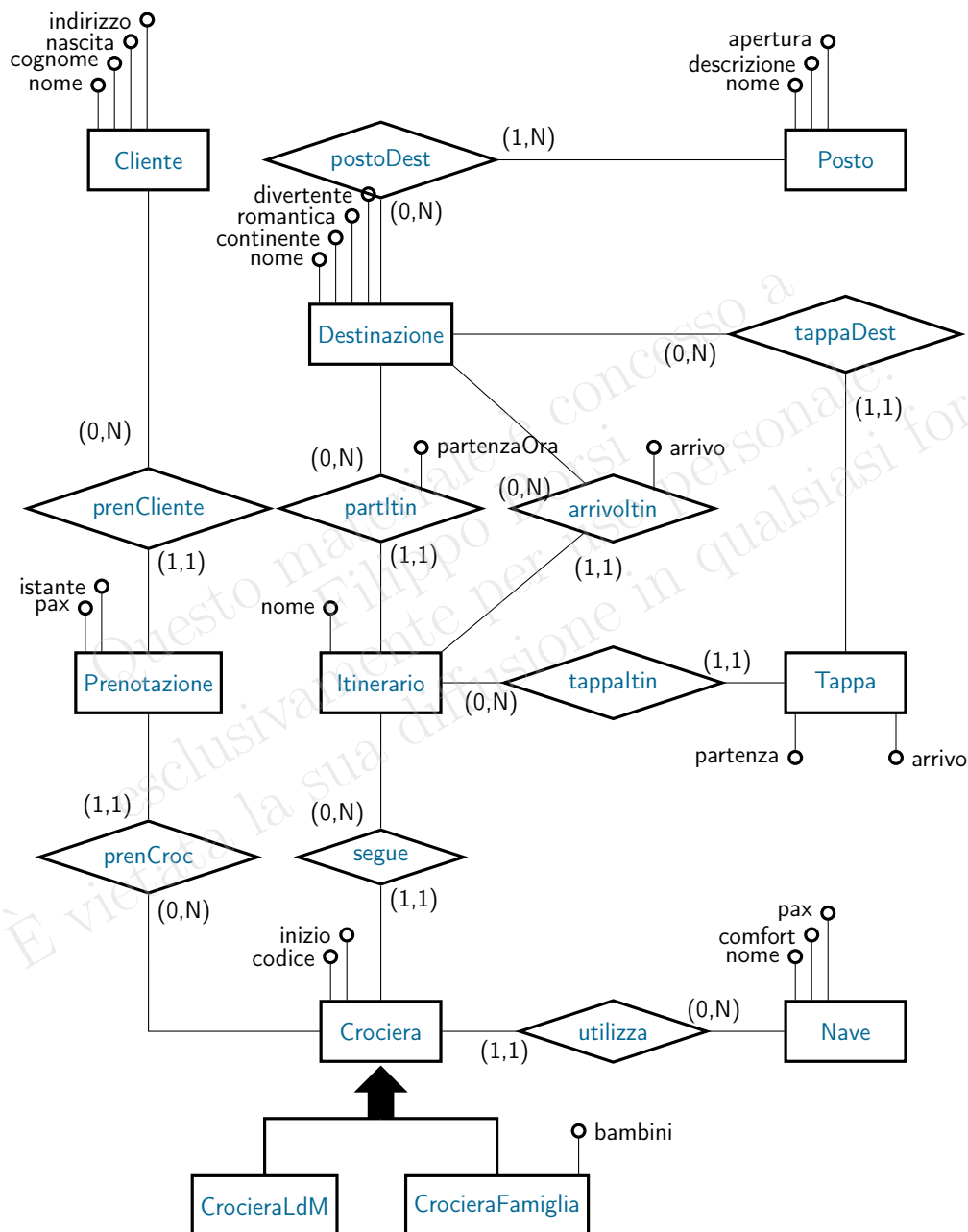
A.4

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'istante 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 `tappaltin`

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 <code>Crociera</code>

Relationship `arrivoltin`

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

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

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



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

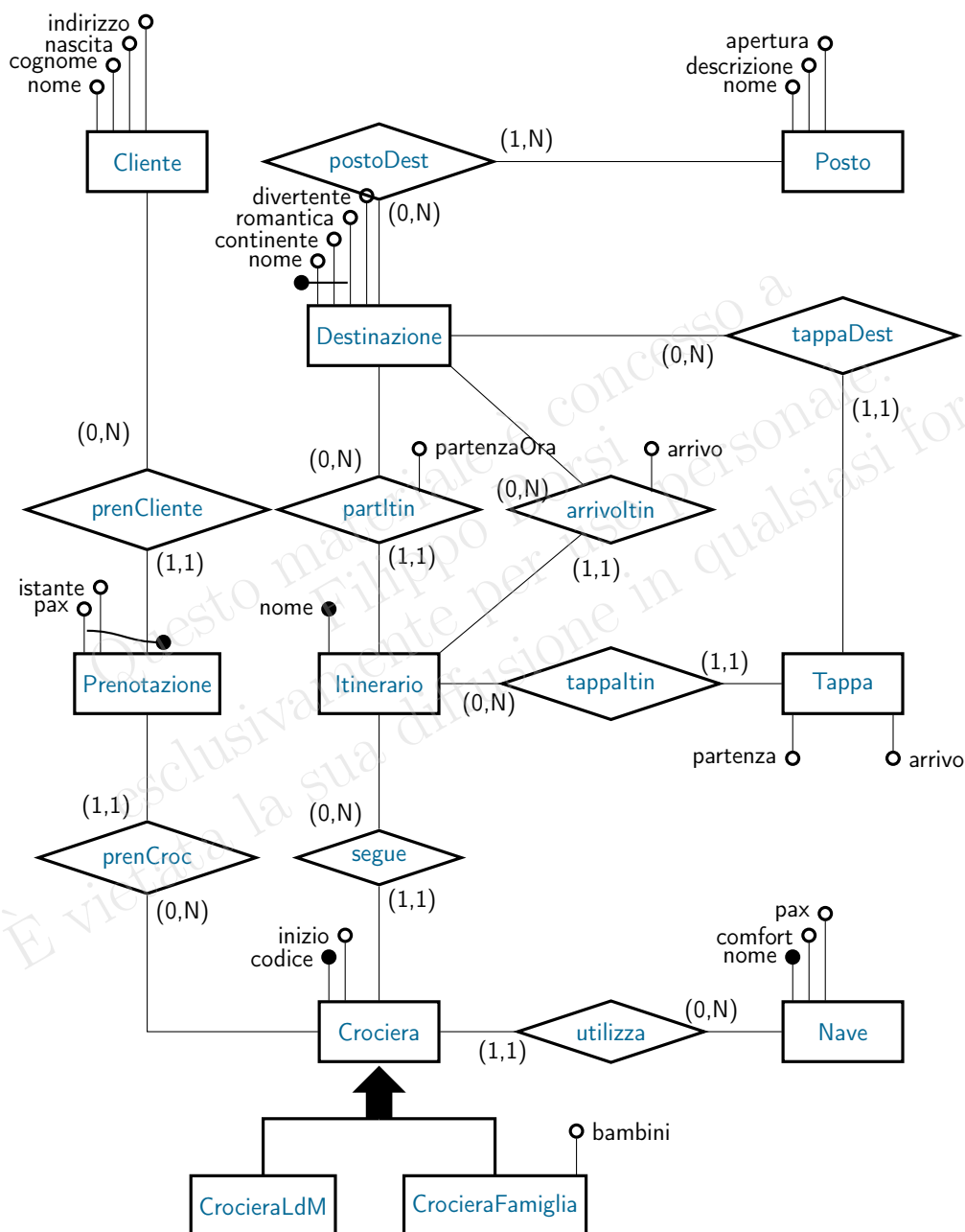
A.5

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 $\text{totPosti} - \text{postiOccupati} \geq 0$.

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.

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:

$$c_in > c_in + giorno_arrivo_c \text{ giorni}$$

oppure

$$c_in > c'_in + giorno_arrivo_c' \text{ giorni.}$$

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 tappaltin
- p la destinazione di partenza di i , ovvero l'istanza dell'entità Destinazione tale che (p, i) è una istanza della relationship partltin
- 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 partltin
- 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 `tappatin`
- `dest_arr` la destinazione di arrivo di i , ovvero l'istanza dell'entità `Destinazione` tale che $(dest_arr, i)$ è una istanza della relationship `arrivoltin`
- `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 `arrivoltin`
- `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'istante 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 [tappaltin](#)

Ogni istanza di questa relationship lega un [Itinerario](#) ad una sua [Tappa](#) (Req. [3.2.](#))

Attributi: Nessuno

Vincoli:

[V.tappaltin.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 *tappaltin*), 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 <i>Crociera</i>

Relationship *arrivoltin*

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

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

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



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

A.6

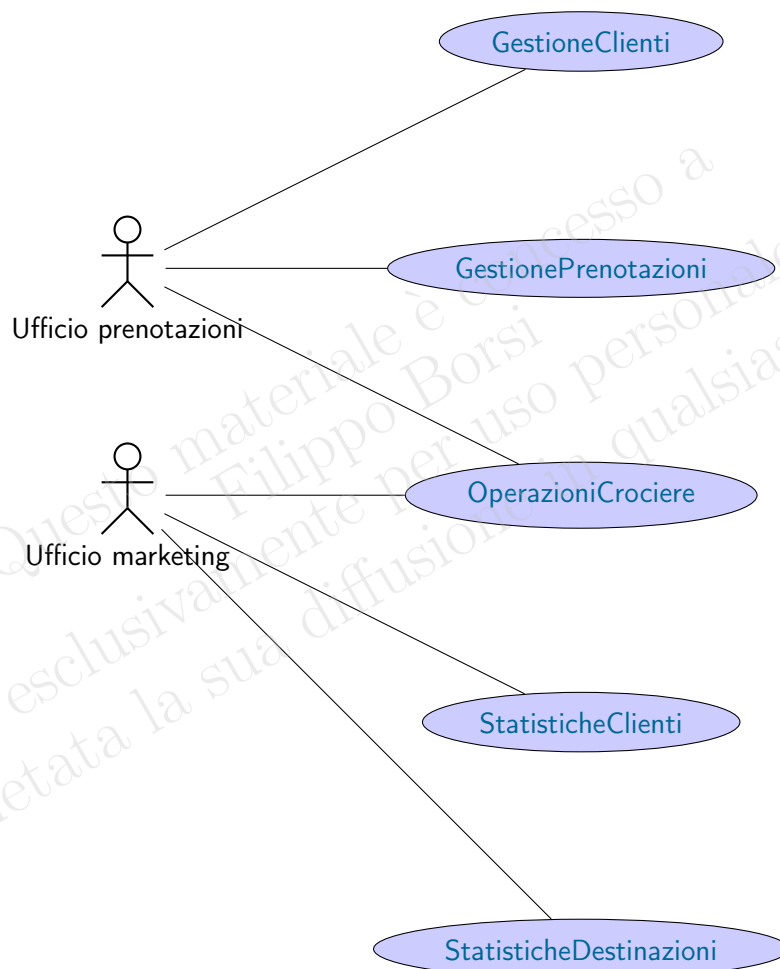
Diagramma UML degli Use-Case

A.6.1 Testo

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

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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.

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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 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 **OperazioniCrociera**

- **tipoCrociera**($c : \text{CrocieraLdM}$) : {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 tappaIn 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 tappaIn 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 tappatin
- esiste una istanza d di entità Destinazione tale per cui (t,d) è istanza della relationship tappaDest
- $\text{StatisticheDestinazioni.esotica}(d)$.

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 tappaltin
 - * (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 tappaltin
 - * (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 $\frac{|D'|}{|D|}$.

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

A.8

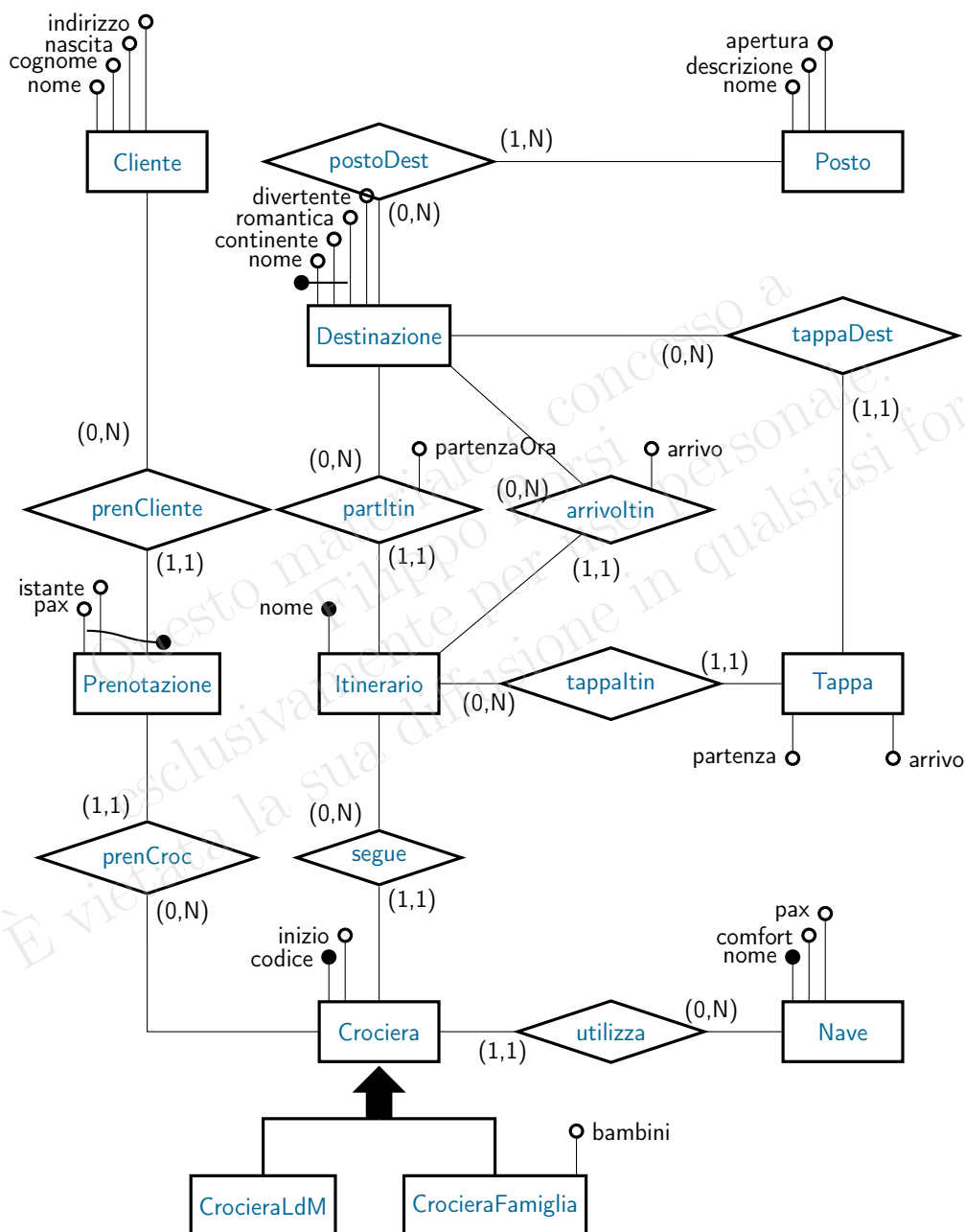
Vincoli Esterni

A.8.1 Testo

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

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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:

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

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:

$$\begin{aligned} &\forall c, c', n, c_in, i_c, i_c', dest_arr_c, dest_arr_c', arr_c, arr_c', arr_g_c, arr_g_c' \\ &[Crociera(c) \wedge Crociera(c') \wedge c \neq c' \wedge utilizza(c, n) \wedge utilizza(c', n) \wedge \\ &inizio(c, c_in) \wedge inizio(c', c'_in) \wedge segue(c, i_c) \wedge segue(c', i_c') \wedge \\ &arrivoltin(dest_arr_c, i_c) \wedge arrivoltin(dest_arr_c', i_c') \wedge \\ &arrivo(dest_arr_c, i_c, arr_c) \wedge arrivo(dest_arr_c', i_c', arr_c') \wedge \\ &giorni(arr_c, arr_g_c) \wedge giorni(arr_c', arr_g_c')] \rightarrow \\ &[c'_in > c_in + arr_g_c \vee c_in > c'_in + arr_g_c']. \end{aligned}$$

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:

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

[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:

$$\begin{aligned} \forall i, \text{dest_arr}, \text{arr}, t_part, t \quad & \text{Itinerario}(i) \wedge \text{arrivoltin}(\text{dest_arr}, i) \wedge \text{arrivo}(\text{dest_arr}, i, \text{arr}) \wedge \\ & \text{Tappa}(t) \wedge \text{tappaltin}(t, i) \wedge \text{partenza}(t, t_part) \rightarrow \\ & t_part < \text{arr} \end{aligned}$$

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'istante 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:

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

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 `tappaltin`

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

Attributi: Nessuno

Vincoli:

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

Formalmente:

$$\begin{aligned} &\forall i, t, t', t_part, t_arr, t'_part, t'_arr \\ &\quad Itinerario(i) \wedge Tappa(t) \wedge tappaltin(t, i) \wedge arrivo(t, t_arr) \wedge partenza(t, t_part) \wedge \\ &\quad Tappa(t') \wedge tappaltin(t', i) \wedge arrivo(t', t'_arr) \wedge partenza(t', t'_part) \wedge \\ &\quad t \neq t' \rightarrow \\ &\quad (t'_arr > t_part) \vee (t_arr > t'_part) \end{aligned}$$

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	ora		L'ora di partenza della <code>Crociera</code>

Relationship `arrivoltin`

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

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

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

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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.

Questo materiale è concesso a
Federico Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

A.9.2 Soluzione

Specifica Use-Case **GestionePrenotazioni**

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

precondizioni: $\forall \text{inizio}(\text{cr}, \text{inizio}) \rightarrow \text{in} < \text{adesso} \wedge$
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 = α

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

precondizioni: nessuna

postcondizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: Detto:

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

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:

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

$$\text{result} = \text{totPosti} - \text{postiOccupati}.$$

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 = α

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

Specifica Use-Case **OperazioniCrociera**

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

precondizioni: nessuna

postcondizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: Siano:

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

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

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

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

Specifica Use-Case **StatisticheClienti**

- $\text{etaMediaEsotica}(C : \text{Cliente } (0,N)) : \text{reale} \geq 0$ (Req. 8.2.)

precondizioni: Sia

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

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:

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

- $\text{eta}(c : \text{Cliente}) : \text{intero} \geq 0$

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):

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

$\text{result} = \text{anni}.$

Specifica Use-Case **StatisticheDestinazioni**

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

precondizioni: nessuna

postcondizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: $\text{result} = \exists \text{cont} \text{ continente}(\text{d}, \text{cont}) \wedge \text{cont} \neq \text{'E'}$.

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

precondizioni: nessuna

postcondizioni:

Modifica del livello estensionale dei dati: nessuna

Valore di ritorno: Siano

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

e

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

rispettivamente il numero di crociere di luna di miele e per famiglia degli ultimi due anni che toccano la destinazione d.

$\text{result} = (N_{ldm} \geq 10) \vee (N_{fam} \geq 15)$.

- **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:

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

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

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

Parte P

Progettazione della Base Dati e delle Funzionalità

Questo materiale è concesso a
Filippo Persi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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.

Questo materiale è concesso a
gruppo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



P.1.2 Soluzione

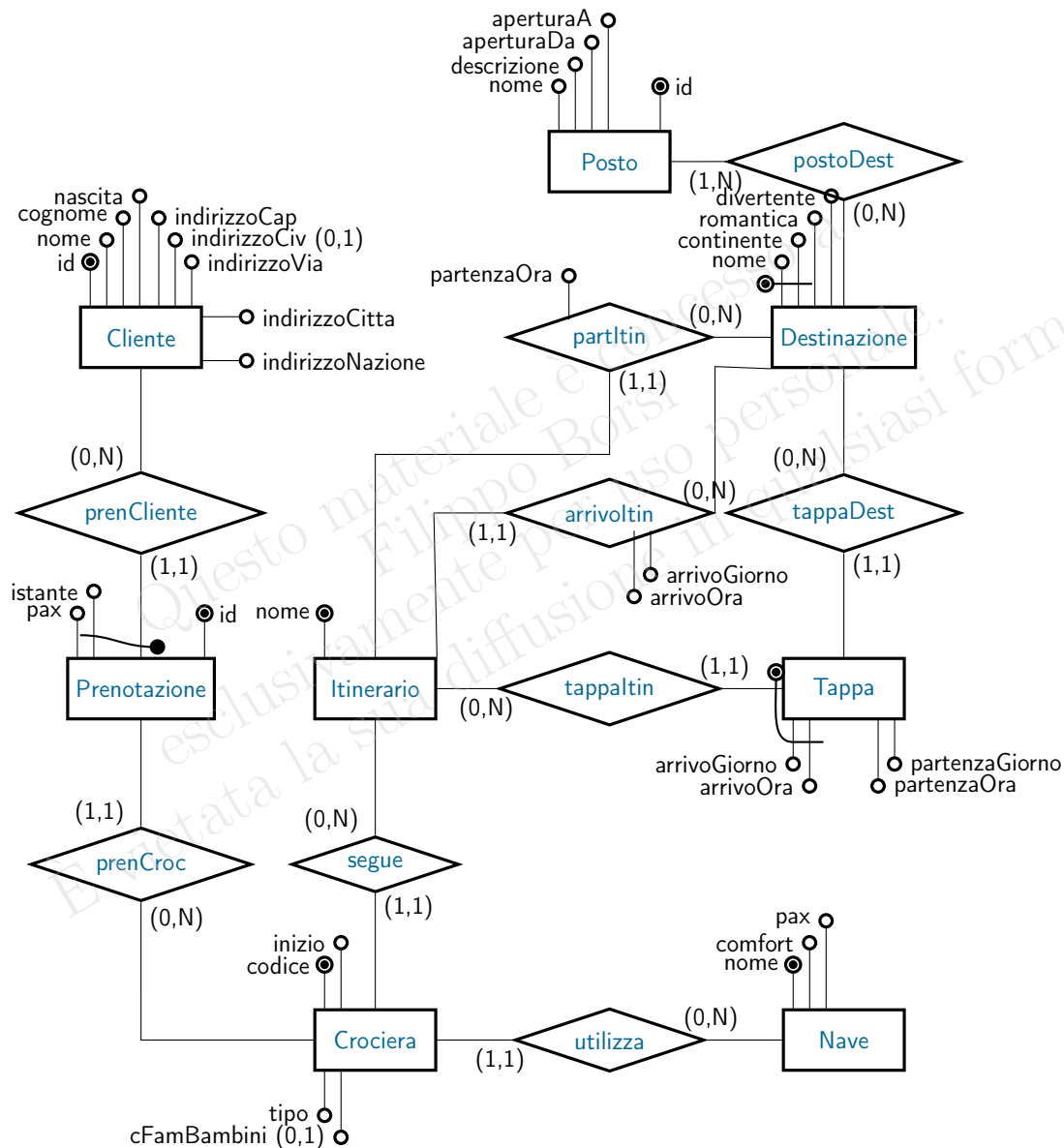
P.1.2.1 Scelta del DBMS

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.

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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:

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

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:

$$\begin{aligned} \forall c, c', n, c_in, i_c, i_c', dest_arr_c, dest_arr_c', arr_g_c, arr_g_c' \\ [Crociera(c) \wedge Crociera(c') \wedge c \neq c' \wedge utilizza(c, n) \wedge utilizza(c', n) \wedge \\ inizio(c, c_in) \wedge inizio(c', c'_in) segue(c, i_c) \wedge segue(c', i_c') \wedge \\ arrivoltin(dest_arr_c, i_c) \wedge arrivoltin(dest_arr_c', i_c') \wedge \\ arrivoGiorno(dest_arr_c, i_c, arr_g_c) \wedge arrivoGiorno(dest_arr_c', i_c', arr_g_c')] \rightarrow \\ [c'_in > c_in + arr_g_c \vee c_in > c'_in + arr_g_c']. \end{aligned}$$

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):

$$\forall t, p_g, p_h, a_g, a_h \\ [Tappa(t) \wedge partenzaGiorno(t, p_g) \wedge partenzaOra(t, p_h) \wedge \\ arrivoGiorno(t, a_g) \wedge arrivoOra(t, a_h)] \rightarrow \\ ((a_g < p_g) \vee (a_g = p_g \wedge a_h < p_h))$$

[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:

$$\forall i, p, t, a_t, o_t, o_p \text{ Itinerario}(i) \wedge partItin(p, i) \wedge partenzaOra(p, i, o_p) \wedge \\ Tappa(t) \wedge tappaltin(t, i) \wedge \\ arrivoGiorno(t, 1) \wedge arrivoOra(t, o_t) \rightarrow \\ o_p < o_t$$

[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:

$$\begin{aligned}
 &\forall i, \text{dest_arr}, \text{dest_arr_g}, \text{dest_arr_h}, t, t_part_g, t_part_h \\
 &\quad \text{Itinerario}(i) \wedge \text{arrivoltin}(\text{dest_arr}, i) \wedge \\
 &\quad \text{arrivoGiorno}(\text{dest_arr}, i, \text{dest_arr_g}) \wedge \text{arrivoOra}(\text{dest_arr}, i, \text{dest_arr_h}) \wedge \\
 &\quad \text{Tappa}(t) \wedge \text{tappaltin}(t, i) \wedge \\
 &\quad \text{partenzaGiorno}(t, t_part_g) \wedge \text{partenzaOra}(t, t_part_h) \rightarrow \\
 &\quad [(t_part_g < \text{dest_arr_g}) \vee \\
 &\quad (t_part_g = \text{dest_arr_g} \wedge t_part_h < \text{dest_arr_h})]
 \end{aligned}$$

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'istante 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:

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

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 tappaltin

Ogni istanza di questa relationship lega un itinerario ad una sua tappa (Req. 3.2.)

Attributi: Nessuno

Vincoli:

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

$$\begin{aligned}
 &\forall i, t, t', t_part_g, t_part_h, t_arr_g, t_arr_h, t'_part_g, t'_part_h, \\
 &\quad t'_arr_g, t'_arr_h \\
 &\quad Itinerario(i) \wedge Tappa(t) \wedge tappaltin(t, i) \wedge \\
 &\quad arrivoGiorno(t, t_arr_g) \wedge arrivoOra(t, t_arr_h) \wedge \\
 &\quad partenzaGiorno(t, t_part_g) \wedge partenzaOra(t, t_part_h) \wedge \\
 &\quad Tappa(t') \wedge tappaltin(t', i) \wedge \\
 &\quad arrivoGiorno(t, t_arr_g) \wedge arrivoOra(t, t_arr_h) \wedge \\
 &\quad partenzaGiorno(t, t_part_g) \wedge partenzaOra(t, t_part_h) \wedge \\
 &\quad t \neq t' \rightarrow \\
 &\quad \left[\begin{aligned} &\left((t'_arr_g > t_part_g) \vee \right. \\ &\quad \left. (t'_arr_g = t_part_g \wedge (t'_arr_h > t_part_h)) \right) \\ &\vee \\ &\left((t_arr_g > t'_part_g) \vee \right. \\ &\quad \left. (t_arr_g = t'_part_g \wedge (t_arr_h > t'_part_h)) \right) \end{aligned} \right]
 \end{aligned}$$

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 *arrivoltin*

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 delle stringhe di esattamente 5 caratteri numerici.

Dominio Continente

Il dominio è dato dall'insieme dei seguenti elementi:

{E, AS, AMN, AMS, AF, O}

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

Dominio TipoCrociera

Il dominio è dato dall'insieme dei seguenti elementi:

{LunaDiMiele, 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.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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, partenzaltin:StringM, partenzaOra:time, arrivoltin:StringM, arrivoGiorno:PosInteger, arrivoOra:time)

La relazione accorpa le relationship partltin e arrivoltin.

[VincoloDB.4] *foreign key*: partenzaltin references Destinazione(nome)

[VincoloDB.5] *foreign key*: arrivoltin references Destinazione(nome)

Tappa (itinerario:StringM, arrivoGiorno:PosInteger, arrivoOra:time, partenzaGiorno:PosInteger, partenzaOra:time, destinazioneNome:StringM, destinazioneContinente:Continente)

La relazione accorpa le relationship tappaltin 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)

Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.



Questo materiale è concesso a
Filippo Borsi
esclusivamente per uso personale.
È vietata la sua diffusione in qualsiasi forma.

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 $Q == \text{NULL}$ 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 già 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 già
        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 già
        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;
```