Matteo Lauro
Biagio Mazzella
Gruppo 2

Traccia 1: Compagnie di Navigazione- Basi di Dati

02-02-2024

UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Indice

1 Progettazione Concettuale

1.1 Analisi Dei Requisiti

Il primo step nella progettazione concettuale di un database riguarda **l'analisi dei requisiti**, ossia la fase nel quale l'obiettivo è, a partire dal definire le funzionalità che il database deve offrire (ovvero i requisiti e le esigenze che devono essere individuati e soddisfatti, indipendentemente da come lo farà), **una formalizzazione** *(in particolare astratta, ossia in modo indipendente dalla tecnologia, dall'implementazione e dal particolare linguaggio di programmazione)* **di un documento di una dettagliata specifica dei requisiti che descrive le funzionalità e caratteristiche.**

In particolare, nel nostro caso della progettazione di una base dati riguardante la gestione delle compagnie di navigazione via mare, si ricavano informazioni da memorizzare e gestire nel database. Si tratta quindi di trovare le entità e che cosa memorizzare, le relazioni tra essi, e di eventuali vincoli o regole in base a ciò che il sistema di database deve fare.

Prima di tutto, essendo un database per la gestione di compagnie di navigazione, dovrà essere formato da un insieme di compagnie, identificato univocamente dal suo nome, essendo che per motivi di copyright non esisteranno compagnie con gli stessi nomi. Le compagnie ovviamente potranno offrire diverse corse, ossia un viaggio, un insieme di percorsi, tra un porto a un altro (eventualmente con uno scalo) in un determinato periodo, orario di partenza e arrivo, con delle imbarcazioni che possono essere un traghetto, motonave e aliscafo, e un determinato prezzo. Inoltre le imbarcazionei avranno una capienza massima, ed anche una capienza massima per i veicoli, se si tratta di un traghetto.

Ad esempio, una compagnia potrebbe operare una corsa di motonave da Mu ad Atlantide soltanto il martedì e il giovedì e nel periodo tra il 15 giugno e il 15 settembre.

Invece nel caso di corse che abbiano uno scalo intermedio il sistema espone tra le sue corse tutte le singole tratte. Ad esempio, se esiste una corsa tra Mu e Atlantide con scalo a Tortuga, il sistema manterrà tutte e tre le corse da Mu a Tortuga, da Tortuga ad Atlantide e da Mu ad Atlantide.

Ogni *Compagnia* ha caratteristiche relative ai suoi contatti *(telefono, sito web, e-mail e indirizzi dei vari social)*. Quindi Compagnia avrà bisogno di un attributo composto per memorizzare i loro *Contatti*, con un attributo multivalore per i vari indirizzi *Social*. Invece per le corse, di utile avranno il suo codice, essendo che corsa è come se fosse un contenitore di più percorsi determinati dai porti, il suo costo standard, lo stato di una corsa che può essere regolare, annullato o in ritardo, e un avviso generale. Inoltre ovviamente una compagnia avrà bisogno delle sue imbarcazioni, che verranno utilizzate per le loro corse. Abbiamo trovato le prime quattro entità: ***Compagnia***, ***Porto***, ***Corsa*** e ***Imbarcazione***, con le eventuali tipologie. Quindi imbarcazione avrà 3 specializzazioni (gerarchia di tipo disgiunta, totale): *Traghetto, Motonave, Aliscafo*. Compagnia, Corsa e Imbarcazione verranno tra loro ovviamente associate, oltre all'associazione tra Corsa e Porto. Questa associazione sarà chiamata **Percorso**, e avrà attributi per memorizzare gli orari e date di partenza e arrivo.

Ovviamente dovremmo sapere che tipo percorso sia, ossia se quel porto, per quella corsa, sarà un percorso di partenza, di scalo o di destinazione. Per fare ciò dovremmo differenziare il tipo percorso, trasformando Percorso in una gerarchia (gerarchia di tipo disgiunta, totale) con figlie *Partenza,Scalo,Destinazione*.

Il sistema, oltre alle compagnie, deve gestire le informazioni relative ai passeggeri e ai biglietti, che quest'ultimi, acquistati dai passeggeri, possono essere ridotti o interi, ed eventualmente con un aumento di prezzo (derivato dal costo standard della corsa) per quelli prenotati e/o con bagagli. Quindi c'è bisogno di una entità *Biglietto* che potrà avere 2 specializzazioni (gerarchia di tipo disgiunta, totale): *Ridotto, Intero*. A sua volta *Intero* avrà come specializzazione di tipo Disgiunta, Parziale, il suo figlio *Sovrapprezzo*. Infine anche Sovrapprezzo sarà un'entità padre con le sue 2 specializzazioni: *Prenotazione, Bagagli* di tipo Overlapping, totale, ossia che può essere un sovrapprezzo per la prenotazione e/o per i bagagli.

Per quanto riguarda i passeggeri, potranno acquistare/prenotare il biglietto in modo normale o per prenotazione, dando le sue credenziali, con la sua email e password. Inoltre se si tratta di un viaggio tramite traghetti, dovranno indicare se intendono trasportare anche il loro veicolo a bordo, insieme alla data acquisto.

Quindi abbiamo bisogno di altre due entità:*Passeggero e Biglietto* e l'entità associativa *Acquista* il quale avrà come attributi la data di acquisto e un flag per sapere se è presente o meno il veicolo. L'entità biglietto avrà bisogno, oltre agli attributi quali il prezzo (ridotto per i bambini), il tipo di biglietto (eventualmente prenotato) e il codice, anche le informazioni sulla corsa. Alla fine anche biglietto sarà relazionato con *Corsa*. Ovviamente *Biglietto* sarà relazionato anche con *Passeggero* attraverso *Acquista*.

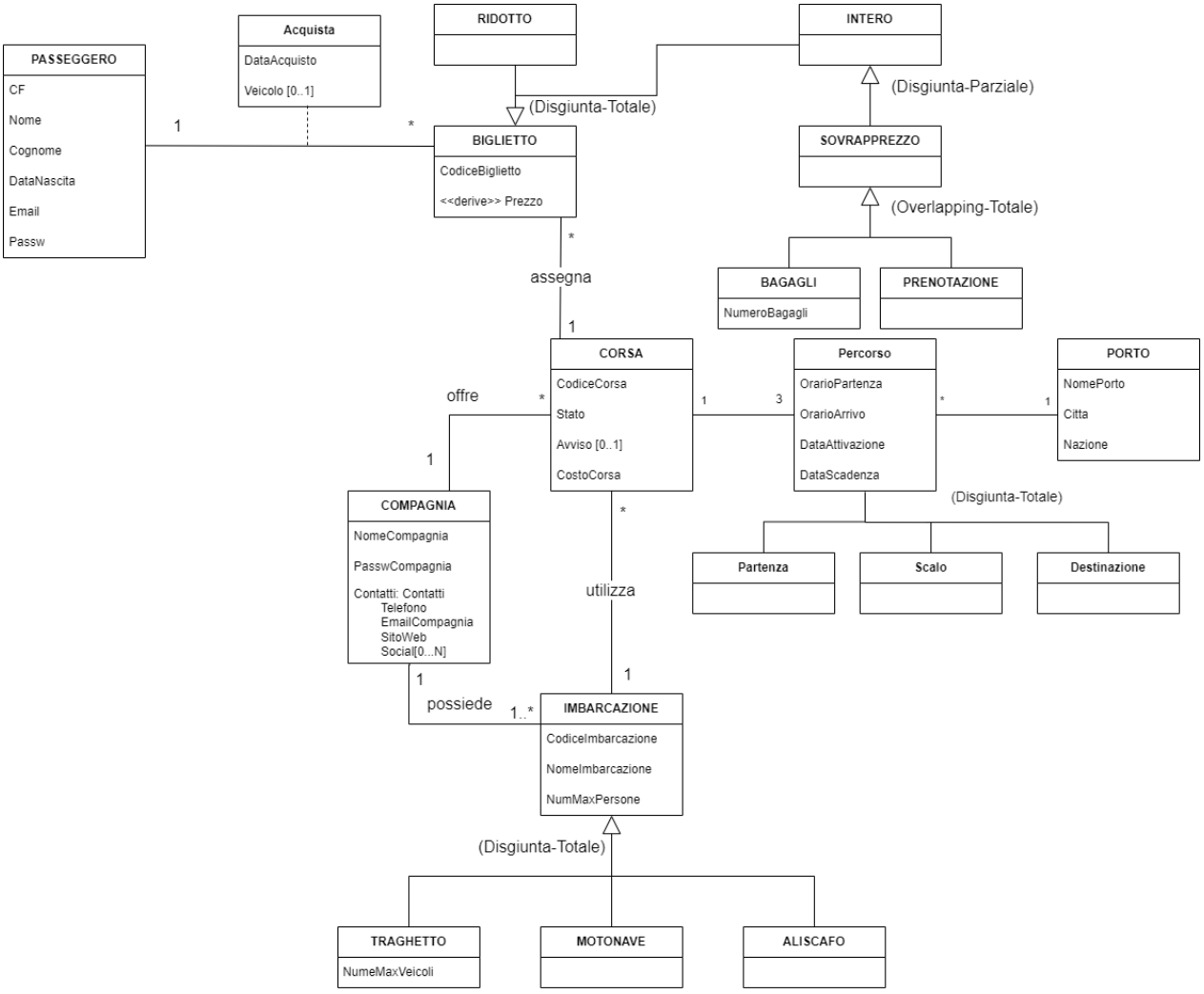
Infine il sistema dovrà permettere l'accesso o la registrazioni di una compagnia o di un passeggero, in modo che, per le compagnie, potranno modificare le proprie corse, aggiungerle, inserire nuovi imbarcazioni ecc. Anche i passeggeri dovranno prima avere un account, in modo che possono acquistare i biglietti, avere una cronologia dei biglietti acquistati, vedere una tabella con i tutti i corsi fino alle prossime 48 ore, eventualmente filtrato in base alla compagnia o prezzo scelto;

In questa descrizione, si evince che abbiamo bisogno di 8 entità (+11 classe figlie):

Compagnia, Corsa, Percorso (Partenza,Scalo,Destinazione), Imbarcazione (Traghetto, Motonave, Aliscafo), Acquista, Porto, Passeggero, Biglietto (Ridotto,Intero(Sovrapprezzo(Prenotazione,Bagagli))).

e 7 associazioni. Ottenute queste informazioni, possiamo descrivere il modello Class Diagram di UML (Unified Modeling Language).

1.2 Schema Concettuale - Schema UML



1.3 Dizionario delle entità e delle associazioni

1.3.1 Dizionario delle entità

Entità	Descrizione	Attributi
Compagnia	Entità che descrive le aziende per le compagnie di navigazione che offrono corsi.	**NomeCompagnia**: Denominazione dell'azienda. **PasswCompagnia**: password utilizzato per accedere al sistema **Contatti (Contatti)**: attributo composto che descrive le informazioni per contattare la compagnia (tutti gli attributi di sotto): **Telefono**: telefono numerico (fisso) della compagnia **Email**: posta elettronica della compagnia **Social**: indirizzi per i diversi social. E' un attributo multivalore **SitoWeb**: il sito web ufficiale della compagnia
Corsa	Entità legata ai dati relative a un percorso dato da una compagnia. Funziona come aggregante per i porti, i biglietti e le imbarcazioni.	**CodiceCorsa**: Codice alfanumerico per riconoscere e verificare la validità di una corsa **Stato**: lo stato di una corsa (es. annullato, rinviato) **Avviso**: messaggio eventuale di avviso della corsa **CostoCorsa**: costo standard senza tener conto di riduzioni o sovrapprezzi dovuto a prenotazioni e/o trasporto di bagagli
Imbarcazione	Rappresenta le varie imbarcazioni con le sue relative informazioni. E' l'entità padre della generalizzazione di tipo: Disgiunta, Totale.	**CodiceImbarcazione**: codice alfanumerico per riconoscere e verificare la validità di un'imbarcazione **NomeImbarcazione**: denominazione dell'imbarcazione **NumMaxPersone**: indica il numero massimo di persone trasportabili da una determinata imbarcazione (il numero varia in base alla dimensione e al tipo di imbarcazione)
Traghetto	Specializzazione di Imbarcazione. Rappresenta un traghetto generico.	***NumMaxVeicoli****: indica il numero massimo di veicoli trasportabili da un determinato traghetto (il numero varia in base alla dimensione del traghetto).*

Motonavi	Specializzazione di Imbarcazione. Rappresenta un motonave generico.	*non sono presenti attributi per Motonave.*
Aliscafo	Specializzazione di Imbarcazione. Rappresenta un aliscafo generico.	*non sono presenti attributi per Aliscafo.*
Passeggero	descrive e generalizza un passeggero quando. Può viaggiare nella corsa specificata nel biglietto quando ha il biglietto	**CF**: rappresenta il codice fiscale del passeggero **Nome**: rappresenta il nome del passeggero **Cognome**: rappresenta il cognome del passeggero **DataNascita**: rappresenta la data di nascita del passeggero **Passw**: password dell'account del passeggero **Email**: e-mail dell'account del passeggero
Biglietto	Entità che descrive il biglietto della corsa. Può essere un biglietto con prezzo intero (eventualmente sovrapprezzo per quelli prenotati, oppure con i bagagli) e ridotto. Oltre al nominativo del passeggero, ha bisogno anche delle informazioni relative alla corsa. E' l'entità padre della generalizzazione di tipo: Disgiunta, Totale.	**CodiceBiglietto**: il codice a barre del biglietto. **Prezzo**: attributo *derivato* il quale rappresenta il prezzo del biglietto, che varia in base alla corsa e alla tipologia del biglietto.
Ridotto	Specializzazione di Biglietto. Rappresenta un biglietto con prezzo ridotto generico.	*non sono presenti attributi per Ridotto.*
Intero	Specializzazione di Biglietto. Rappresenta un biglietto con prezzo ridotto generico. E' anche l'entità padre della generalizzazione di tipo: Disgiunta, Parziale, con la specializzazione Sovrapprezzo	*non sono presenti attributi per Intero.*
Sovrapprezzo	Specializzazione di Intero. Rappresenta un biglietto intero con prezzo aumentato. E' anche l'entità padre della generalizzazione di tipo: Overlapping, Totale, con le specializzazioni Prenotazione e Bagagli	*non sono presenti attributi per Sovrapprezzo.*
Prenotazione	Specializzazione di Sovrapprezzo. Rappresenta un biglietto intero avendo il prezzo con sovrapprezzo, ma prenotato.	*non sono presenti attributi per Prenotazione.*

Bagagli	Specializzazione di Sovrapprezzo. Rappresenta un biglietto intero avendo il prezzo con sovrapprezzo, specificando anche la possibilità di trasporto di bagagli.	***NumeroBagagli****: indica il numero di bagagli portati da un passeggero che può variare da 0 a 3.*
Porto	Descrive le informazioni di un porto	**NomePorto**: il nome del porto. **Citta**: la città in cui è presente il porto. **Nazione**: il paese dove è presente il porto nella determinata città.
Percorso	Rappresenta un'entità associativa tra *Corsa* e *Porto*. E' un'entità padre della generalizzazione di tipo: Disgiunta, Totale.	**OrarioPartenza**: Rappresenta l'orario di partenza da un porto in una corsa **OrarioArrivo**: Orario che rappresenta l'orario di arrivo al porto di destinazione. **DataAttivazione**: Data di Partenza da un porto in una corsa **DataScadenza**: Data che rappresenta la data di arrivo al porto di destinazione
Partenza	Specializzazione di Percorso Rappresenta il percorso di partenza in base alla corsa e al porto (di partenza)	*non sono presenti attributi per Partenza.*
Scalo	Specializzazione di Percorso Rappresenta il percorso di scalo in base alla corsa e al porto (di scalo)	*non sono presenti attributi per Scalo.*
Arrivo	Specializzazione di Percorso Rappresenta il percorso di arrivo in base alla corsa e al porto (di arrivo) essendo il percorso di arrivo, le date e gli orari devono essere uguali, essendo l'ultima tappa	*non sono presenti attributi per Arrivo.*
Acquista	Rappresenta un'entità associativa tra *Passeggero* e *Biglietto*.	***DataAcquisto****: rappresenta la data d'acquisto del biglietto da parte di un passeggero.* ***Veicolo****:flag che mi permette di sapere se il passeggero vuole trasportare anche il suo veicolo*

1.3.2 Dizionario delle associazioni

Associazioni	Descrizione
Acquista	Associazioni *uno-a-molti* tra ***Passeggero*** e ***Biglietto***. Questa associazione dice che un passeggero può acquistare nessuno o più biglietti, e viceversa, un biglietto dev'essere acquistato da un solo passeggero. Questo perché il biglietto è unico, ed assegnato all'unico acquirente. Mentre il passeggero, al momento della registrazione non ha ancora comprato nessun biglietto. Inoltre l'associazione memorizza anche la data d'acquisto, se il passeggero è munito di veicolo o meno, e se trasporta con se bagagli
Assegna	Associazione *molti-a-uno* tra ***Biglietto*** e ***Corsa***. Questa associazione dice che un Biglietto deve assegnare una sola corsa, e viceversa, una corsa può essere assegnata da nessuna o più biglietti. Per il motivo in cui biglietto è assegnato a una corsa sola ovviamente, mentre un biglietto della corsa può non essere comprato da nessuno, quindi non esisteranno biglietti per quella corsa.
Percorso	Associazione *molti-a-molti(3)* tra ***Corsa*** e ***Porto***. Questa associazione dice che una corsa deve percorrere da 1 a tre porti, e viceversa, un porto può essere percorso da nessuna o più corse. Questo perché una corsa ha bisogno di almeno 2 porti per essere completo, ossia quello di partenza e destinazione, al massimo 3 aggiungendo quello di scalo. Invece un porto può non essere indicato in nessun porto per un determinato periodo.
Utilizza	Associazione *molti-a-uno* tra ***Corsa*** e ***Imbarcazione***. Questa associazione dice che una corsa deve utilizzare una imbarcazione, e viceversa, un'imbarcazione può essere utilizzata da nessuna o più corse. Questo perchè una corsa ha bisogno obbligatoriamente un'imbarcazione, mentre può capitare che un'imbarcazione può non essere utilizzata da nessuna corsa.
Possiede	Associazione *uno-a-molti* tra ***Compagnia*** e ***Imbarcazione***. Questa associazione descrive che una Compagnia deve possedere una o più imbarcazioni, e viceversa, un'imbarcazione dev'essere posseduta da una sola compagnia. Questo perchè una compagnia per esistere, e offrire corse, deve almeno avere un'imbarcazione
Offre	Associazione *uno-a-molti* tra ***Compagnia*** e ***Corsa***. Questa associazione descrive che una Compagnia può offrire una o più corse, e viceversa, una corsa dev'essere offerta da una sola Compagnia.

2 Ristrutturazione dello schema concettuale

In questa seconda fase andremo ad ottimizzare e semplificare lo schema ER/UML, per renderlo conforme a una corretta formattazione al modello logico, con i dovuti step.

2.1 Analisi Delle Ridondanze

Analizzando lo schema UML non ci sono ridondanze di ogni genere. Ci potrebbe essere una ridondanza tra *NumMaxPersone* in **Imbarcazione** e il numero di biglietti venduti, ma sono due cose diverse essendo che *NumMaxPersone* indica la capienza massima di quella specifica imbarcazione e non il numero effettivo di passeggeri in quella data. Contiene quindi lasciarlo poichè raramente si andrà ad aggiornarlo, andando a risparmiare tempo di calcolo per ogni interrogazione, qualora ci servisse. Stessa costa tra *CostoCorsa* in **Corsa** e *Prezzo* in **Biglietto**. Il costo della corsa indica il prezzo standard senza aumenti o riduzioni del genere, il quale biglietto poi dovrà calcolare il prezzo specifico in base alla percentuale applicata sul costo standard.

2.2 Eliminazione delle generalizzazioni

Ci sono 4 generalizzazioni, ossia:

1. ***Imbarcazione***, con le sue specializzazioni: ***Traghetto***, ***Motonave*** e ***Aliscafo***. E' una generalizzazione disgiunta, totale. Si è deciso di ristrutturarlo con il metodo dell'accorpamento delle entità figlie nell'entità padre, aggiungendo come attributo *TipoImbarcazione*, in modo da specificare se sono dei traghetti, motonave e aliscafo. Si è deciso di utilizzare questo metodo, essendo che se usassimo il metodo dell'entità padre nelle figlie, andremo ad aumentare la complessità dello schema per la gestione delle relazioni. Stessa cosa per le sostituzione delle generalizzazioni con le sostituzioni. In generale abbiamo minimizzato gli accessi alla memoria anche se abbiamo uno spreco maggiore della stessa essendo che verranno memorizzati valori nulli.

2. ***Sovrapprezzo***, con le sue specializzazioni: ***Bagagli***, ***Prenotazione***. E' una generalizzazione di tipo overlapping, totale. Si è deciso di ristrutturarlo con il metodo dell'accorpamento delle entità figlie nell'entità padre, utilizzando due attributi: *Prenotazione* e NumeroBagagli, il quale *Prenotazione* funge da flag, il quale se è true, allora si tratta di una prenotazione. Invece NumeroBagagli è un tipo intero che descrive sia se si tratta che di un biglietto con bagagli, e nel caso anche il numero di bagagli. Se è 0 allora vuol dire che non si sta trasportando nessun bagaglio, ovvero è semplicemente sovrapprezzo. Si è deciso di utilizzare questo metodo per lo stesso motivo della generalizzazione precedente.

3. ***Intero***, con le sua specializzazione: ***Sovrapprezzo***. E' una generalizzazione di tipo disgiunta, parziale. Si è deciso di ristrutturarlo con il metodo dell'accorpamento delle entità figlie nell'entità padre, rimanendo gli attributi così come sono. Si è deciso di utilizzare questo metodo per lo stesso motivo della generalizzazione precedente. Per il prezzo con sovrapprezzo, sarà memorizzato nell'attributo prezzo, il quale ovviamente sarà più alto di quello intero.

4. ***Biglietto***, con le sue specializzazioni: ***Intero*** e ***Ridotto***. E' una generalizzazione di tipo disgiunta, totale. Si è deciso di ristrutturarlo con il metodo dell'accorpamento delle entità figlie nell'entità padre, aggiungendo come attributo *TipoBiglietto*, sostituendolo a *TipoIntero* in modo da specificare se è un biglietto intero o ridotto, con eventualmente dei bagagli e/o prenotato, specificato negli altri due attributi (bagagli e prenotati saranno con un sovrapprezzo). Si è deciso di utilizzare questo metodo per lo stesso motivo delle generalizzazioni precedenti.

5. ***Percorso***, con le sue specializzazioni: ***Partenza***, ***Scalo*** e ***Arrivo***. E' una generalizzazione disgiunta, totale. Si è deciso di ristrutturarlo con il metodo dell'accorpamento delle entità figlie nell'entità padre, aggiungendo come attributo *TipoPercorso*, in modo da specificare il tipo di percorso in base al porto, agli orari, date e alla corsa. Si è deciso di utilizzare questo metodo, essendo che se usassimo il metodo dell'entità padre nelle figlie, andremo ad aumentare la complessità dello schema per la gestione delle associazioni. Stessa cosa per le sostituzione delle generalizzazioni con le sostituzioni. In generale abbiamo minimizzato gli accessi alla memoria anche se abbiamo uno spreco maggiore della stessa essendo che verranno memorizzati valori nulli.

2.3 Eliminazione Attributi Strutturati

L'unico attributo strutturato è *Contatti* in ***Compagnia***. Si è deciso di scomporlo in base ai suoi attributi, sempre nell'entità ***Compagnia***.

2.4 Eliminazione Attributi Multivalore

E' presente solo attributi multivalore: *Social* in ***Contatti*** (che prima era anche composto). Quindi Social diventerà un'entità, con i suoi dovuti attributi: *TipoSocial* e *URL*, e relazionato con molteplicità 1:N con Compagnia.

2.5 Partizionamento/accorpamento di entità e associazioni

Non ci sono associazioni di molteplicità 1:1 e in generale elementi in cui si effettua questa fase.

2.6 Scelta degli identificatori primari

In questa fase andremo ad trovare uno o più attributi per Entità in modo identificarli univocamente:

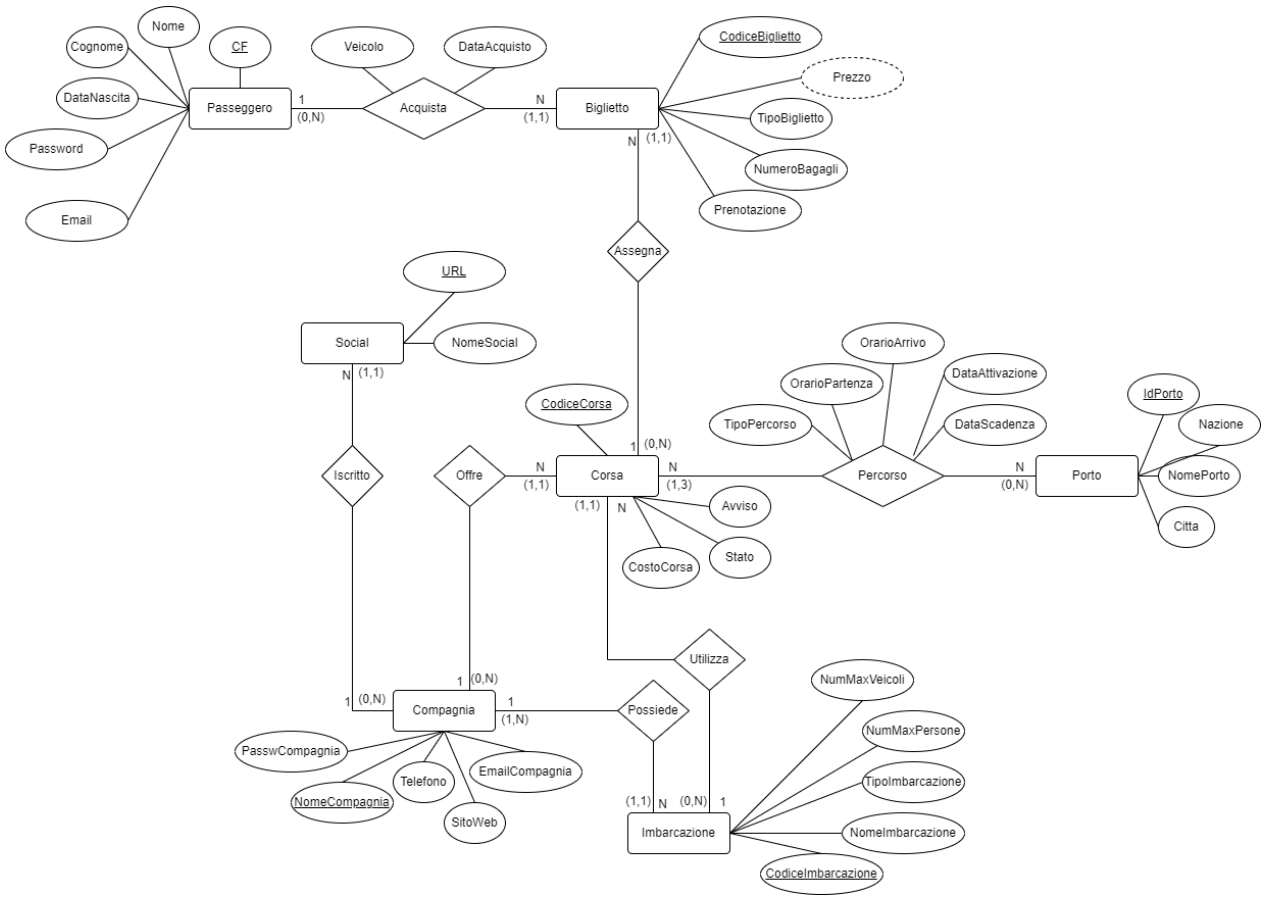
- **Compagnia**: un attributo che può essere potenzialmente una chiave primaria è *NomeCompagnia*, essendo che per copyright, non possono esserci Compagnie con lo stesso nome.

- **Social**: un attributo che può essere potenzialmente una chiave primaria è *URL*, essendo che un sito web o in generale la pagina di un social è identificato in modo univoco dalle altre.

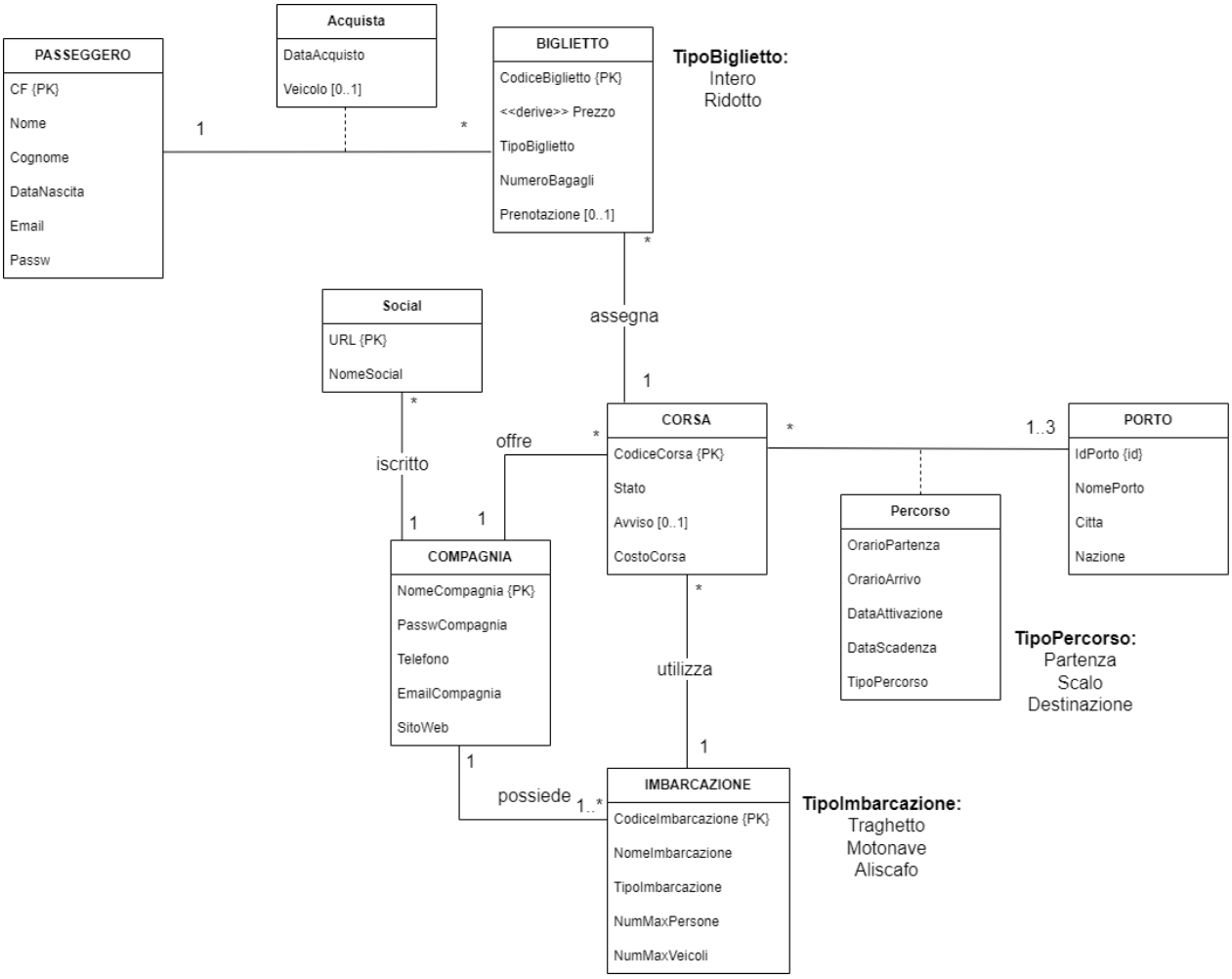
- **Porto**: due attributi che, uniti, possono essere potenzialmente una chiave primaria è ***Città*** e ***Nazione*** essendo che è quasi impossibile, che ci sono città con lo stesso nome e nello stesso paese. Tuttavia, per velocizzare le operazioni e l'accesso all'indice, si va a creare un nuovo attributo ***IDPorto***, che diventerà la chiave primaria.
- **Corsa**: un attributo che può essere potenzialmente una chiave primaria è *CodiceCorsa*, essendo che per legge una corsa, quindi un percorso di una imbarcazione dev'essere riconosciuto in modo da identificare anche la tratta.
- **Imbarcazione**: la chiave primaria per questa entità è ***CodiceImbarcazione***, per lo stesso motivo di Corsa. Per legge una imbarcazione dev'essere riconosciuta tramite un codice per tracciamenti, controlli ecc.
- **Biglietto**: la chiave primaria per questa entità è ***CodiceBiglietti***, poiché ogni biglietto, anche quelli online, avranno un codice per essere controllati all'ingresso durante le imbarcazioni.
- **Passeggero**: la chiave primaria per questa entità è CF, ossia il codice fiscale.

2.7 Schema Concettuale Ristrutturato

2.7.1 Schema ER (Entity-Relationship)



2.7.2 Schema UML



2.7.3 Dizionario delle entità Ristrutturato

Entità	Descrizione	Attributi
Compagnia	Entità che descrive le aziende per le compagnie di navigazione che offrono corsi.	**NomeCompagnia**: Denominazione dell'azienda. **PasswCompagnia**: password utilizzato per accedere al sistema **Telefono**: telefono numerico (fisso) della compagnia **Email**: posta elettronica della compagnia **Social**: indirizzi per i diversi social. E' un attributo multivalore **SitoWeb**: il sito web ufficiale della compagnia
Corsa	Entità legata ai dati relative a un percorso dato da una compagnia. Funziona come aggregante per i porti, i biglietti e le imbarcazioni.	**CodiceCorsa**: Codice alfanumerico per riconoscere e verificare la validità di una corsa **Stato**: lo stato di una corsa (es. annullato, rinviato) **Avviso**: messaggio eventuale di avviso della corsa **CostoCorsa**: costo standard senza tener conto di riduzioni o sovrapprezzi dovuto a prenotazioni e/o trasporto di bagagli
Imbarcazione	Rappresenta le varie imbarcazioni con le sue relative informazioni e tipologie.	**CodiceImbarcazione**: codice alfanumerico per riconoscere e verificare la validità di un'imbarcazione **NomeImbarcazione**: denominazione dell'imbarcazione **TipoImbarcazione**: il tipo dell'imbarcazione, ossia traghetto, motonave o aliscafo **NumMaxPersone**: indica il numero massimo di persone trasportabili da una determinata imbarcazione (il numero varia in base alla dimensione e al tipo di imbarcazione) **NumMaxVeicoli**: indica il numero massimo di veicoli trasportabili da un determinato traghetto (il numero varia in base alla dimensione del traghetto).

Passeggero	descrive e generalizza un passeggero quando. Può viaggiare nella corsa specificata nel biglietto quando ha il biglietto	**CF**: rappresenta il codice fiscale del passeggero **Nome**: rappresenta il nome del passeggero **Cognome**: rappresenta il cognome del passeggero **DataNascita**: rappresenta la data di nascita del passeggero **Passw**: password dell'account del passeggero **Email**: e-mail dell'account del passeggero
Biglietto	Entità che descrive il biglietto della corsa. Può essere un biglietto con prezzo intero (eventualmente sovrapprezzo per quelli prenotati, oppure con i bagagli) e ridotto. Oltre al nominativo del passeggero, ha bisogno anche delle informazioni relative alla corsa.	**CodiceBiglietto**: il codice a barre del biglietto. **Prezzo**: attributo *derivato* il quale rappresenta il prezzo del biglietto, che varia in base alla corsa e alla tipologia del biglietto. **TipoBiglietto**: il tipo di biglietto, ossia ridotto o intero **NumeroBagagli**: il numero di bagagli trasportati **Prenotazione**: flag in cui indica se il biglietto è acquistato il giorno stesso della corsa o prenotato per il giorno dopo
Porto	Descrive le informazioni di un porto	**NomePorto**: il nome del porto. **Citta**: la città in cui è presente il porto. **Nazione**: il paese dove è presente il porto nella determinata città.
Percorso	Rappresenta un'entità associativa tra *Corsa* e *Porto*.	**OrarioPartenza**: Rappresenta l'orario di partenza da un porto in una corsa **OrarioArrivo**: Orario che rappresenta l'orario di arrivo al porto di destinazione. **DataAttivazione**: Data di Partenza da un porto in una corsa **DataScadenza**: Data che rappresenta la data di arrivo al porto di destinazione **TipoPercorso**: indica il tipo di percorso, ossia partenza,scalo o destinazione

Acquista	Rappresenta un'entità associativa tra *Passeggero* e *Biglietto*.	***DataAcquisto****: rappresenta la data d'acquisto del biglietto da parte di un passeggero.* ***Veicolo****:flag che mi permette di sapere se il passeggero vuole trasportare anche il suo veicolo*
Social	Rappresenta il link della compagnia di un social. Ovviamente è unico essendo che la pagina url è unica.	***URL****: rappresenta la pagina web della compagnia* ***NomeSocial****:Il nome del social il quale Compagnia è iscritto*

2.7.4 Dizionario delle associazioni Ristrutturato

Associazioni	Descrizione
Acquista	Associazioni *uno-a-molti* tra ***Passeggero*** e ***Biglietto***. Questa associazione dice che un passeggero può acquistare nessuno o più biglietti, e viceversa, un biglietto dev'essere acquistato da un solo passeggero. Questo perché il biglietto è unico, ed assegnato all'unico acquirente. Mentre il passeggero, al momento della registrazione non ha ancora comprato nessun biglietto. Inoltre l'associazione memorizza anche la data d'acquisto, se il passeggero è munito di veicolo o meno, e se trasporta con se bagagli
Assegna	Associazione *molti-a-uno* tra ***Biglietto*** e ***Corsa***. Questa associazione dice che un Biglietto deve assegnare una sola corsa, e viceversa, una corsa può essere assegnata da nessuna o più biglietti. Per il motivo in cui biglietto è assegnato a una corsa sola ovviamente, mentre un biglietto della corsa può non essere comprato da nessuno, quindi non esisteranno biglietti per quella corsa.
Percorso	Associazione *molti-a-molti(3)* tra ***Corsa*** e ***Porto***. Questa associazione dice che una corsa deve percorrere da 1 a tre porti, e viceversa, un porto può essere percorso da nessuna o più corse. Questo perché una corsa ha bisogno di almeno 2 porti per essere completo, ossia quello di partenza e destinazione, al massimo 3 aggiungendo quello di scalo. Invece un porto può non essere indicato in nessun porto per un determinato periodo.
Utilizza	Associazione *molti-a-uno* tra ***Corsa*** e ***Imbarcazione***. Questa associazione dice che una corsa deve utilizzare una imbarcazione, e viceversa, un'imbarcazione può essere utilizzata da nessuna o più corse. Questo perchè una corsa ha bisogno obbligatoriamente un'imbarcazione, mentre può capitare che un'imbarcazione può non essere utilizzata da nessuna corsa.
Possiede	Associazione *uno-a-molti* tra ***Compagnia*** e ***Imbarcazione***. Questa associazione descrive che una Compagnia deve possedere una o più imbarcazioni, e viceversa, un'imbarcazione dev'essere posseduta da una sola compagnia. Questo perchè una compagnia per esistere, e offrire corse, deve almeno avere un'imbarcazione
Offre	Associazione *uno-a-molti* tra ***Compagnia*** e ***Corsa***. Questa associazione descrive che una Compagnia può offrire una o più corse, e viceversa, una corsa dev'essere offerta da una sola Compagnia.
Iscritto	Associazione *uno-a-molti* tra ***Compagnia*** e ***Social***. Questa associazione descrive che una Compagnia può essere iscritto in nessuna o più social, e viceversa, un social dev'essere iscritto da una sola Compagnia.

3 Traduzione al Modello Logico

3.1 Mapping Associazioni

3.1.1 Associazioni 1:N

- *Compagnia* - ***Iscritto*** - *Social*: inserimento della chiave di **Compagnia** in **Social** come chiave esterna.
- *Compagnia* - ***Possiede*** - *Imbarcazione*: inserimento della chiave di **Compagnia** in **Imbarcazione** come chiave esterna.
- *Compagnia* - ***Offre*** - *Corsa*: inserimento della chiave di **Compagnia** in **Corsa** come chiave esterna.
- *Biglietto* - ***Assegna*** - *Corsa*: inserimento della chiave di **Corsa** in **Biglietto** come chiave esterna.
- *Passeggero* - ***Acquisto*** - *Biglietto*: inserimento della chiave di **Passeggero** in **Biglietto** come chiave esterna.
- *Corsa* - ***Utilizza*** - *Imbarcazione*: inserimento della chiave di **Corsa** in **Imbarcazione** come chiave esterna.

3.1.2 Associazioni N:N

Per ogni Associazione N:N, si inserisco le chiavi delle due entità, associate ad esso, come chiavi esterne.

Entità	Relazione	Entità
Corsa	Percorso	Porto

3.2 Modello Logico

Gli attributi sottolineati sono le chiavi primarie per ogni relazione. Le $\uparrow$ indicano le chiavi esterne e $*$ gli attributi null per ogni relazione.

1. **Compagnia**(NomeCompagnia, PasswCompagnia, Telefono, EmailCompagnia, SitoWeb)
2. **Social** (URL, NomeSocial, CompIsc$\uparrow$)
    - Social.CompIsc $\Rightarrow$ Compagnia.NomeCompagnia
3. **Imbarcazione**(CodiceImbarcazione, NomeImbarcazione, TipoImbarcazione, NumMaxPersone, NumMaxVeicoli, CompPoss$\uparrow$)
    - Imbarcazione.CompProp $\Rightarrow$ Compagnia.NomeCompagnia

4. **Corsa**(CodiceCorsa, CostoCorsa, Stato, Avviso$*$, FKImb$\uparrow$, FKComp$\uparrow$)
    - Corsa.FKImb $\Rightarrow$ Imbarcazione.CodiceImbarcazione
    - Corsa.FKComp $\Rightarrow$ Compagnia.NomeCompagnia
5. **Passeggero**(CF, Nome, Compagnia, DataNascita, Email, Passw)
6. **Porto**(IdPorto, NomePorto, Citta, Nazione)
7. **Biglietto**(CodiceBiglietto, TipoBiglietto, Prezzo, NumeroBagagli, Prenotazione, DataAcquisto, Veicolo$*$, CFPass$\uparrow$, CorAs$\uparrow$, NumeroBagagli, Prenotazione)
    - Biglietto.CFPass $\Rightarrow$ Passeggero.CF
    - Biglietto.CorAs $\Rightarrow$ Corsa.CodiceCorsa
8. **Percorso** (CorPer$\uparrow$, IdPer$\uparrow$, TipoPercorso, OrarioPartenza, OrarioArrivo, DataAttivazione, DataScadenza)
    - Percorso.CorPer $\Rightarrow$ Corsa.CodiceCorsa
    - Percorso.IdPer $\Rightarrow$ Porto.IdPorto

4 Progettazione Fisica

Questa quarta fase ha l'obiettivo di implementare lo schema logico descrivendo e dettagliando gli aspetti fisici di memorizzazione e rappresentazione in memoria gestito poi da un DMBS, in questo caso da PostgreSQL

4.1 Domini

I domini in SQL sono dei vincoli che indicano un tipo di dati e le sue regole da rispettare per l'insieme dei valori assegnatosi alla colonna

- Domino per indicare il tipo di biglietto (intero o ridotto):

```
1    CREATE DOMAIN TipoBigl AS VARCHAR(8)
2    CONSTRAINT verificabiglietto CHECK(VALUE IN('intero', '
       ridotto'));
```

- Dominio per controllare che l'email venga inserita nel formato giusto:

```
1   CREATE DOMAIN Email AS Varchar(30)
2    CONSTRAINT FormatoEmail CHECK(VALUE LIKE '%_@_%._%');
```

- Dominio per controllare che venga inserito uno tra i tre tipi di stato per le corse (annullato, ritardo, regolare):

```
1    CREATE DOMAIN statocorsa AS Varchar(9)
2    CONSTRAINT checkcorsa CHECK(VALUE IN('annullato', '
       ritardo','regolare'));
```

- Dominio per controllare che venga inserito uno tra i tre tipi di imbarcazione (traghetto, motonave, aliscafo):

```
1    CREATE DOMAIN tipoimbar AS Varchar(9)
2    CONSTRAINT checkimbar CHECK(VALUE IN('traghetto', '
       motonave','aliscafo'));
```

4.2 Creazione delle Tabelle

```
CREATE TABLE Compagnia(
    NomeCompagnia    Varchar(50)          PRIMARY KEY,
    PasswCompagnia VARCHAR(30)            NOT NULL,
    Telefono         Varchar(11)      NOT NULL,
    EmailCompagnia   email            NOT NULL,
    SitoWeb          Varchar(50)      NOT NULL
);


CREATE TABLE Social(
    URL          Varchar(100)         PRIMARY KEY,
    NomeSocial       Varchar(30)      NOT NULL,
    CompIsc          Varchar(50)      NOT NULL,

    CONSTRAINT CompagniaIscritta FOREIGN KEY(CompIsc)
       REFERENCES Compagnia(NomeCompagnia)
        ON DELETE cascade
        ON UPDATE cascade
);


CREATE TABLE Imbarcazione(
    CodiceImbarcazione  Varchar(5)           PRIMARY KEY,
    NomeImbarcazione    Varchar(30)          NOT NULL,
    TipoImbarcazione    tipoimbar            NOT NULL,
    CompPoss            Varchar(50)          NOT NULL,
    NumMaxPersone       int                  NOT NULL,
    NumMaxVeicoli       int                  NOT NULL,

    CONSTRAINT Compagnia_che_possiede FOREIGN KEY(CompPoss)
       REFERENCES Compagnia(NomeCompagnia)
    ON DELETE cascade
    ON UPDATE cascade
);


CREATE TABLE Corsa(
    CodiceCorsa      Varchar(10)      PRIMARY KEY,
    CostoCorsa       Numeric(10,2)    NOT NULL,
    Avviso           Varchar(40),
    Stato            statocorsa       NOT NULL,
    FKImb            Varchar(5)       NOT NULL,
    FKComp           Varchar(50)      NOT NULL,

    CONSTRAINT Imbarcazione_Utilizzata FOREIGN KEY(FKImb)
       REFERENCES Imbarcazione(CodiceImbarcazione)
    ON DELETE cascade
```

```
45      ON UPDATE cascade,
46
47      CONSTRAINT FormatoCodCorsa CHECK (CodiceCorsa LIKE '
          C_______S0'OR CodiceCorsa LIKE 'C_______S1' OR
          CodiceCorsa LIKE 'C_______S2'),
48
49      CONSTRAINT Compagnia_Offerta FOREIGN KEY(FKComp) REFERENCES
           Compagnia(NomeCompagnia)
50      ON DELETE cascade
51      ON UPDATE cascade
52
53  );
54
55  CREATE TABLE Passeggero(
56      CF              char(16)            PRIMARY KEY,
57      Nome                Varchar(20)         NOT NULL,
58      Cognome             Varchar(30)         NOT NULL,
59      DataNascita         Date                NOT NULL,
60      Email               Email       NOT NULL,
61      Passw               Varchar(30) NOT NULL
62  );
63
64  CREATE TABLE Porto(
65      IdPorto         SERIAL              PRIMARY KEY,
66      NomePorto       Varchar(30)         NOT NULL,
67      Citta           Varchar(30)         NOT NULL,
68      Nazione         Varchar(30)         NOT NULL
69  );
70
71
72
73
74  CREATE TABLE Biglietto(
75      CodiceBiglietto         Varchar(5)          PRIMARY KEY,
76      TipoBiglietto           tipobigl            NOT NULL,
77      Prezzo                  Numeric(10,2)       ,
78      DataAcquisto            Date                NOT NULL,
79      Veicolo                 boolean,
80      NumBagagli              Int NOT NULL,
81      Prenotazione            boolean             ,
82      CFPass                  Char(16)            NOT NULL,
83      CorAs                   Varchar(10)         NOT NULL,
84
85      CONSTRAINT ProprietarioBiglietto FOREIGN KEY(CFPass)
          REFERENCES Passeggero(CF)
86      ON DELETE set NULL
87      ON UPDATE cascade,
88
```

```
89      CONSTRAINT CorsaAssociata FOREIGN KEY(CorAs) REFERENCES
            Corsa(CodiceCorsa)
90      ON DELETE cascade
91      ON UPDATE cascade
92
93  );
94
95
96
97  CREATE TABLE Percorso(
98      CorPer          Varchar(10)     NOT NULL,
99      IdPer               int         NOT NULL,
100         TipoPercorso            VarChar(14)             NOT
                NULL,
101     OrarioPartenza      Time            NOT NULL,
102     OrarioArrivo        Time            NOT NULL,
103     DataAttivazione     Date            NOT NULL,
104     DataScadenza        Date            NOT NULL,
105
106     CONSTRAINT PKPercorso PRIMARY KEY(CorPer, IdPer),
107     CONSTRAINT VerificaPercorso CHECK (TipoPercorso IN ('
            partenza', 'scalo', 'destinazione')),
108
109     CONSTRAINT CorsaPercorsa FOREIGN KEY(CorPer) REFERENCES
            Corsa(CodiceCorsa)
110     ON DELETE cascade
111     ON UPDATE cascade,
112
113     CONSTRAINT IdPercorso FOREIGN KEY(IdPer) REFERENCES Porto(
            IdPorto)
114     ON DELETE cascade
115     ON UPDATE cascade
116 );
```

4.3 Implementazione Vincoli intra-relazionali

I vincoli intra-relazionali sono vincoli che coinvolgono solo una tabella

- Vincolo per controllare che l'orario di partenza sia prima di quella di arrivo (anche in base al giorno):

```
ALTER TABLE Percorso ADD CONSTRAINT checkorario CHECK(
((DataAttivazione <= Datascadenza) AND (OrarioPartenza<=
   OrarioArrivo)) OR ((DataAttivazione < Datascadenza)
AND (OrarioPartenza>OrarioArrivo)));
```

- Vincolo per controllare che due o più persone diverse non possono avere la stessa email (su passeggeri):

```
ALTER TABLE Passeggero ADD CONSTRAINT emailUnicaP UNIQUE(
   Email);
```

- Vincolo per controllare che due o più persone diverse non possono avere la stessa email (su Compagnia):

```
ALTER TABLE Compagnia ADD CONSTRAINT emailUnicaC UNIQUE(
   EmailCompagnia);
```

- Vincolo per controllare che non ci siano imbarcazioni con lo stesso nome e codice:

```
ALTER TABLE Imbarcazione ADD CONSTRAINT nomeImbUnica UNIQUE
   (codiceimbarcazione, nomeimbarcazione);
```

- Vincolo per controllare che il costo di una corsa non sia un valore negativo:

```
ALTER TABLE Corsa ADD CONSTRAINT CostoPositivoCorsa CHECK(
   CostoCorsa>=0.00);
```

- Vincolo per controllare che il prezzo di un biglietto non sia un valore negativo:

```
ALTER TABLE Biglietto ADD CONSTRAINT
   PrezzoPositivoBiglietto CHECK(Prezzo>=0.00);
```

4.4 Vincoli inter-relazionali

I vincoli inter-relazionali sono vincoli che coinvolgono due o più tabelle. Possono essere gestiti attraverso trigger, che si innescano attraverso una particolare istruzione di DML (Data Manipulation Language), quali l'insert, delete e update

4.4.1 calcoloPrezzo (Procedura calcoloPrezzo)

```
CREATE OR REPLACE FUNCTION calcoloprezzo()
RETURNS TRIGGER AS $$

DECLARE
    costo Numeric(10,2);
    data_partenza Date;
    preno Boolean;
BEGIN


    SELECT CostoCorsa INTO costo
    FROM Corsa
    WHERE CodiceCorsa=new.coras;

    SELECT datapartenza INTO data_partenza
    FROM Tabellone
    WHERE new.coras = codicecorsa;

    -- Calcola la prenotazione
    IF(CURRENT_DATE = data_partenza) THEN
        preno = false;
    ELSE
        preno = true;
    END IF;

    UPDATE biglietto
    SET prenotazione = preno
    where NEW.codiceBiglietto = codiceBiglietto;


    -- il prezzo di una singola valigia    di 5 euro, mentre
       quello della prenotazione e di 2 euro
    IF (New.tipobiglietto = 'ridotto') THEN
            IF (preno = true) THEN

                UPDATE Biglietto
                SET Prezzo = (costo/2) + (New.numbagagli*5)+2
                WHERE NEW.codiceBiglietto = codiceBiglietto;

            ELSE

```

```
41                  UPDATE Biglietto
42                  SET Prezzo = (costo/2) + (New.numbagagli*5)
43                  WHERE NEW.codiceBiglietto = codiceBiglietto;
44
45              END IF;
46      ELSE
47
48          IF(NEW.numbagagli>0 AND preno = true) THEN
49
50              UPDATE Biglietto
51              SET Prezzo = costo + (New.numbagagli*5)+2
52              WHERE NEW.codiceBiglietto = codiceBiglietto;
53
54          -- saranno 2 euro in pi
55          ELSEIF (preno = true) THEN
56
57              UPDATE Biglietto
58              SET Prezzo = costo + 2
59              WHERE NEW.codiceBiglietto = codiceBiglietto;
60
61          ELSEIF(NEW.numbagagli>0) THEN
62
63              UPDATE Biglietto
64              SET Prezzo = costo + (New.numbagagli*5)
65              WHERE NEW.codiceBiglietto = codiceBiglietto;
66          ELSE
67              UPDATE Biglietto
68              SET Prezzo = costo
69              WHERE NEW.codiceBiglietto = codiceBiglietto;
70
71          END IF;
72
73      END IF;
74
75      RETURN NEW;
76  END;
77
78  $$ LANGUAGE plpgsql;
79
80  CREATE TRIGGER calcoloprezzo
81  AFTER INSERT ON biglietto
82  FOR EACH ROW
83  EXECUTE PROCEDURE calcoloprezzo();
```

4.4.2 calcolomaxpersone (Procedura controllomax)

```
CREATE OR REPLACE FUNCTION controllomax()
RETURNS TRIGGER AS $$

DECLARE
    PostiPersoneDisp INTEGER;
    PostiVeicoliDisp INTEGER;

BEGIN
    --vado a prelevare la capienza sia per i passeggeri che
       veicoli disponibili chiamando la funzione
       getPostiDisponibili
    select persone, veicoli into PostiPersoneDisp,
       PostiVeicoliDisp
    FROM getPostiDisponibili(new.coras);


    if (PostiPersoneDisp = 0) then
        raise exception 'attenzione: capienza massima dei
           passeggeri raggiunta';
    end if;
    if(PostiVeicoliDisp = 0 AND new.veicolo=true) then
        raise exception 'attenzione: capienza massima dei
           veicoli raggiunta';
    end if;

    RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER calcolomaxpersone
BEFORE INSERT ON biglietto
FOR EACH ROW
EXECUTE PROCEDURE controllomax();
```

4.4.3 numpercorsi (Procedura controllopercorsi)

```
CREATE OR REPLACE FUNCTION controllopercorsi()
RETURNS TRIGGER AS $$

DECLARE

    numPercorsi Integer;
BEGIN
    if(new.tipoPercorso='partenza') then
        --se sto inserando un percorso di partenza, vado a
           vedere se per quella corsa c'    gi
        SELECT count(*) into numPercorsi
```

```
11          FROM percorso
12          where corper=new.corper and tipoPercorso='partenza';
13
14          if(numPercorsi<>0) then
15              raise exception 'attenzione: la corsa (%) ha gi
                    un porto di partenza', new.corper;
16          end if;
17      else
18          if(new.tipoPercorso='destinazione') then
19              --se sto inserando un percorso di destinazione,
                    vado a vedere se per quella corsa c'   gi
20              SELECT count(*) into numPercorsi
21              FROM percorso
22              where corper=new.corper and tipoPercorso='
                    destinazione';
23
24              if(numPercorsi<>0) then
25                  raise exception 'attenzione: la corsa (%) ha
                        gi   un porto di destinazione', new.corper;
26              end if;
27          else
28              --se sto inserando un percorso di scalo, vado a
                    vedere se per quella corsa c'   gi
29              SELECT count(*) into numPercorsi
30              FROM percorso
31              where corper=new.corper and tipoPercorso='scalo';
32
33              if(numPercorsi<>0) then
34                  raise exception 'attenzione: la corsa (%) ha
                        gi   un porto di scalo', new.corper;
35              end if;
36          end if;
37      end if;
38
39
40      RETURN NEW;
41  END; $$ LANGUAGE plpgsql;
42
43  CREATE TRIGGER numpercorsi
44  BEFORE INSERT ON percorso
45  FOR EACH ROW
46  EXECUTE PROCEDURE controllopercorsi();
```

4.4.4 ControlloCorsa (Procedura corsaValida)

```
CREATE OR REPLACE FUNCTION corsaValida()
RETURNS TRIGGER AS $$

DECLARE

    numPercorsi Integer;
BEGIN
    SELECT count(*) into numPercorsi
        FROM percorso
        where corper=new.coras and (tipoPercorso='partenza' or
           tipoPercorso='destinazione');

    --se la corsa ha un percorso partenza e destinazione,
       allora va bene

        if(numPercorsi<>2) then
            raise exception 'attenzione: la corsa (%) non
               valida. Non ha ancora una destinazione/partenza
               decisa', new.coras;
        end if;


    RETURN NEW;
END; $$ LANGUAGE plpgsql;


CREATE TRIGGER ControlloCorsa
BEFORE INSERT ON biglietto
FOR EACH ROW
EXECUTE PROCEDURE corsaValida();
```

4.4.5 noModificaPercorso (Procedura noModificaPercorso)

```
CREATE OR REPLACE FUNCTION noModificaPercorso()
RETURNS TRIGGER AS $$

BEGIN

    raise exception 'attenzione:    vietato modificare i
       percorsi. Cancella prima la corsa e inserisci di nuovo i
       percorsi';


    RETURN NEW;
END; $$ LANGUAGE plpgsql;


CREATE or REPLACE TRIGGER noModificaPercorso
BEFORE UPDATE ON percorso
FOR EACH ROW
EXECUTE PROCEDURE noModificaPercorso();
```

4.4.6 noModificaImb (Procedura noModificaImb)

```
CREATE OR REPLACE FUNCTION noModificaCorsaImb()
RETURNS TRIGGER AS $$

BEGIN

    raise exception 'attenzione:    vietato cambiare
       imbarcazione';


    RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE or REPLACE TRIGGER noModificaCorsaImb
BEFORE UPDATE OF fkimb ON Corsa
FOR EACH ROW
EXECUTE PROCEDURE noModificaCorsaImb();
```

4.4.7 imbarcazioniGiorniDiversi (Procedura imbarcazioneGiorni)

```
1  CREATE OR REPLACE FUNCTION imbarcazioniGiorni()
2  RETURNS TRIGGER AS $$
3
4  DECLARE
5
6      risultato Integer;
7      compagnia_nuovo VARCHAR(30);
8      imbarcazione_nuovo VARCHAR(5);
9
10 BEGIN
11     SELECT DISTINCT fkcomp into compagnia_nuovo
12         FROM corsa join percorso on codicecorsa=new.corper
13         where corper=new.corper;
14
15     SELECT DISTINCT fkimb into imbarcazione_nuovo
16         FROM corsa join percorso on codicecorsa=new.corper
17         where corper=new.corper;
18
19     --dopo aver prelevato la compagnia e l'imbarcazione,
       controllo se l'imbarcazione    utilizzata o meno nello
       stesso intervallo di partenza e scadenza
20
21     SELECT count(*) into risultato
22     FROM (Corsa join tappainiziale as ti on codicecorsa=ti.
       corper) join tappafinale as tf on ti.corper=tf.corper
23     where (substring(codicecorsa,2,7)<>substring(new.corper
       ,2,7)) and corsa.fkimb=imbarcazione_nuovo and corsa.
       fkcomp=compagnia_nuovo
24     and ((new.dataattivazione<dataarrivo) and(new.datascadenza>
       datapartenza));
25
26
27
28         if(risultato<>0) then
29             raise exception 'ATTENZIONE: quella imbarcazione
                   gi  usata in quella data, usa una imbarcazione
                 diversa da (%)', imbarcazione_nuovo;
30         end if;
31
32
33     RETURN NEW;
34 END; $$ LANGUAGE plpgsql;
35
36 CREATE TRIGGER imbarcazioniGiorniDiversi
37 BEFORE INSERT ON percorso
38 FOR EACH ROW
39 EXECUTE PROCEDURE imbarcazioniGiorni();
```

4.4.8 CheckImbarcazione (Procedura imbarcazioneGiusta)

```
1     CREATE OR REPLACE FUNCTION imbarcazioneGiusta()
2  RETURNS TRIGGER AS $$
3
4  DECLARE
5
6      risultato Integer;
7  BEGIN
8      SELECT count(*) into risultato
9          FROM imbarcazione
10         where codiceimbarcazione=new.fkimb and compposs=new.
               fkcomp;
11
12
13
14         if(risultato=0) then
15             raise exception 'ATTENZIONE: quella imbarcazione
                  non    proprieta di (%)', new.fkcomp;
16         end if;
17
18
19     RETURN NEW;
20 END; $$ LANGUAGE plpgsql;
21
22 CCREATE TRIGGER CheckImbarcazione
23 BEFORE INSERT ON Corsa
24 FOR EACH ROW
25 EXECUTE PROCEDURE imbarcazioneGiusta();
```

4.5 Dizionario dei Vincoli

Nome Vincolo	Descrizione
verificabiglietto	Vincolo di **dominio** sulle tuple utilizzato per il tipo di dato TipoBiglietto in Biglietto. Essa vincola che il valore, ossia un varchar(8), del tipo di biglietto, può essere solamente intero o ridotto
FormatoEmail	Vincolo di **dominio** sulle tuple utilizzato per la corretta formattazione dell'email
checkcorsa	Vincolo di **dominio** sulle tuple utilizzato per il tipo di dato TipoImbarcazione in Imbarcazione. Essa vincola che il valore, ossia un varchar(9), del tipo d'imbarcazione, può essere solamente traghetto, motonave o aliscafo
CompagniaIscritta	Vincolo di **integrità referenziale** per la chiave esterna **CompIsc** in Social, che si riferisce alla chiave primaria di Compagnia. Ha un effetto cascade sia su ON DELETE che ON UPDATE
Compagnia che possiede	Vincolo di **integrità referenziale** per la chiave esterna **CompPoss** in Imbarcazione, che si riferisce alla chiave primaria di Compagnia. Ha un effetto cascade sia su ON DELETE che ON UPDATE
Imbarcazione Utilizzata	Vincolo di **integrità referenziale** per la chiave esterna **FKImb** in Corsa, che si riferisce alla chiave primaria di Imbarcazione. Ha un effetto cascade sia su ON DELETE che ON UPDATE
Compagnia Offerta	Vincolo di **integrità referenziale** per la chiave esterna **FKComp** in Corsa, che si riferisce alla chiave primaria di Compagnia. Ha un effetto cascade sia su ON DELETE che ON UPDATE
FormatoCodCorsa	Vincolo sulle **tuple (check)** utilizzato su CodiceCorsa in Corsa, dove limita il corretto formattamento del codice corsa. Un carattere C all'inizio, un codice numero di 7 cifre fisso, e alla fine una stringa che può essere S0,S1,S2, dove S0 rappresenta la corsa con scalo dal porto partenza-porto scalo-porto destinazione. S1 rappresenta la "sotto" corsa di S0, da porto partenza-porto scalo. S2 rappresenta la sottocorsa sempre di S0, da porto scalo a porto destinazione.
ProprietarioBiglietto	Vincolo di **integrità referenziale** per la chiave esterna **CFPass** in Biglietto, che si riferisce alla chiave primaria di Biglietto. Ha un effetto cascade su ON UPDATE e un effetto SET NULL su ON UPDATE, utile per memorizzare gli incassi della compagnia
CorsaAssociata	Vincolo di **integrità referenziale** per la chiave esterna **CorAs** in Biglietto, che si riferisce alla chiave primaria di Corsa. Ha un effetto cascade sia su ON DELETE che ON UPDATE
PKPercorso	Vincolo di **chiave** di ***CorPer e IdPer***, su Percorso.

VerificaPercorso	Vincolo sulle **tuple (check)**, utilizzato per il tipo di dato TipoPercorso in Percorso. Essa vincola che il valore, ossia un varchar(14), del tipo di Percorso, può essere solamente partenza, scalo o destinazione
CorsaPercorsa	Vincolo di **integrità referenziale** per la chiave esterna **CorPer** in Percorso, che si riferisce alla chiave primaria di Corsa. Ha un effetto cascade sia su ON DELETE che ON UPDATE
IdPercorso	Vincolo di **integrità referenziale** per la chiave esterna **IdPer** in Percorso, che si riferisce alla chiave primaria di Porto. Ha un effetto cascade sia su ON DELETE che ON UPDATE
checkorario	Vincolo sulle **tuple (check)** utilizzato in percorso il quale vincola che l'orario di attivazione e data di attivazione venga prima (o uguale) all'orario e data di scadenza. Oppure che la data di attivazione venga prima di quella di scadenza, e l'orario di partenza supera quella di arrivo. Esempio partenza alle 23 del 14 marzo e arrivo alle 15 del 16 marzo
emailUnicaP	Vincolo di **unicità** dell'email su Passeggero
emailUnicaC	Vincolo di **unicità** dell'email su Compagnia
CostoPositivoCorsa	Vincolo sulle **tuple (check)** utilizzato nel costo della Corsa, il quale dev'essere positivo, ossia maggiore o uguale a 0
PrezzoPositivoBiglietto	Vincolo sulle **tuple (check)** utilizzato nel prezzo del Biglietto, il quale dev'essere positivo, ossia maggiore o uguale a 0
calcoloprezzo (Procedura calcoloprezzo)	Vincolo **inter-relazionale (trigger)**, il quale prima di inserire una riga su biglietto, calcola il prezzo del biglietto in base alla corsa, al tipo biglietto, alla prenotazione, bagagli e veicolo
calcolomaxpersone (Procedura calcolomax)	Vincolo **inter-relazionale (trigger)**, il quale prima di inserire una riga su biglietto, controlla se ci sono ancora posti disponibili (anche per i veicoli) sull'imbarcazione della corsa. Se è tutto pieno, solleva un'eccezione
numpercorsi (Procedura controllopercorsi)	Vincolo **inter-relazionale (trigger)**, il quale prima di inserire una riga su percorsi, controlla se la corsa, presente nel percorso che stiamo per inserire, ha già una partenza e destinazione (eventualemente anche scalo). Se ci sono, solleva un'eccezione
ControlloCorsa (Procedura corsaValida)	Vincolo **inter-relazionale (trigger)**, il quale prima di inserire una riga su biglietto, controlla se la corsa è valida, ossia se ha almeno partenza o destinazione. Se incompleta, solleva un'eccezione
noModificaPercorso (Procedura noModificaPercorso)	Vincolo **inter-relazionale (trigger)**, il quale vieta di modificare una riga su percorso.
noModificaImb (Procedura noModificaImb)	Vincolo **inter-relazionale (trigger)**, il quale vieta di modificare imbarcazione su una riga di corsa.

imbarcazioneGiorniDiversi (*Procedura imbarcazioniGiorni*)	Vincolo **inter-relazionale (trigger)**, il quale prima dell'inserimento di una riga su percorso, controlla se l'imbarcazione su percorso associata alla corsa che stiamo inserendo, non sia già utilizzata in un'altra sua corsa negli due intervalli di date, o meglio, l'intervallo di tempo che stiamo controllando, e l'intervallo di tempo della riga che stiamo per inserire, non devono sovrapporsi.
CheckImbarcazione (*Procedura imbarcazioneGiusta*)	Vincolo **inter-relazionale (trigger)**, il quale prima di inserire una riga su corsa, controlla se l'imbarcazione associata alla corsa che stiamo inserendo, sia sua. Se non lo è, solleva un'eccezione.

4.6 Procedure e Funzioni

4.6.1 Procedura AggiungiImbarcazione

```
CREATE OR REPLACE PROCEDURE AggiungiImbarcazione(
    CodImbar varchar(5),
    NomeImbar varchar(30),
    TipoImbar tipoImbar,
    NomeCompPoss varchar(50),  -- Nome della compagnia che la
       possiede
    MaxPersone int,
    MaxVeicoli int
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO Imbarcazione
    VALUES (CodImbar,NomeImbar,TipoImbar,NomeCompPoss,
       MaxPersone, MaxVeicoli);
END; $$;
```

4.6.2 Procedura AcquistaBiglietto

```
    CREATE OR REPLACE PROCEDURE AcquistaBiglietto(
    codicebiglietto varchar(5),
    vtipobiglietto tipobigl,
    dataacquisto date,
    veicolo boolean,
    numerobagagli int,
    cfpass varchar(16),
    varcoras varchar(5)
)
LANGUAGE plpgsql
```

```
AS $$
BEGIN
    INSERT INTO Biglietto(codicebiglietto, tipobiglietto ,
       dataacquisto, veicolo, numbagagli, cfpass, coras)
    VALUES (codicebiglietto, vtipobiglietto, dataacquisto,
       veicolo, numerobagagli, cfpass, varcoras);
END; $$;
```

4.6.3 Procedura AggiungiPass

```
CREATE OR REPLACE PROCEDURE AggiungiPass(CF CHAR(16),Nome
   VARCHAR(20),Cognome VARCHAR(30), dataNascita date,Email
   VARCHAR(30), Passw VARCHAR(30))
as $$
begin
    INSERT INTO Passeggero
    VALUES(CF, Nome, Cognome, dataNascita, Email, Passw);
end;
$$  LANGUAGE plpgsql;
```

4.6.4 Procedura cambiastato

```
CREATE OR replace PROCEDURE cambiastato(Codice VARCHAR(10),
   messaggio VARCHAR(40), statop statocorsa, NomeCompagnia
   VARCHAR(50))
as $$
begin
    update Corsa
    set stato=statop, Avviso=messaggio
    where CodiceCorsa=Codice AND FKComp = NomeCompagnia;
EXCEPTION
    WHEN others THEN
        RAISE EXCEPTION 'Errore: inserisci uno stato tra
           regolare, annullato o ritardo ';

end;
$$  LANGUAGE plpgsql;
```

4.6.5 Funzione incassicorsa

```
CREATE FUNCTION incassicorsa(Codice VARCHAR(10))
RETURNS NUMERIC AS $$
DECLARE somma numeric(10,2) := 0.0;
var numeric(10,2) :=0.0;
cursor1 cursor for select Prezzo from biglietto join Corsa on
   Coras=CodiceCorsa WHERE CodiceCorsa=Codice;
```

```
BEGIN
        OPEN cursor1;
    LOOP
        FETCH cursor1 INTO var;
        EXIT WHEN NOT FOUND;
        somma := somma+var;
    END LOOP;
    CLOSE cursor1;


        RETURN somma;
END;
$$  LANGUAGE plpgsql;
```

4.6.6 Funzione getPostiDisponibili

```
CREATE or replace function getPostiDisponibili(CodCorsa varchar
   (10), OUT persone Integer, Out veicoli integer) AS $$
DECLARE
    v1 int;
    v2 int;
    p1 int;
    p2 int;
BEGIN

    IF(CodCorsa like 'C_______S0') THEN
       select (SELECT nummaxveicoli
        FROM imbarcazione JOIN corsa ON codiceimbarcazione =
           FKImb
        WHERE codiceCorsa = CodCorsa)
        -
        (SELECT count(*)
        FROM biglietto
        WHERE coras = CodCorsa AND veicolo = true)
        into veicoli;

        select (SELECT nummaxpersone
         FROM imbarcazione JOIN corsa ON codiceimbarcazione =
            FKImb
         WHERE codiceCorsa = CodCorsa)
         -
         (SELECT count(*)
         FROM biglietto
         WHERE coras = CodCorsa) into persone;

         --calcola numero di posti dei veicoli occupati nella
            prima corsa
        SELECT count(*)into v1
```

```
29          FROM biglietto
30          WHERE coras = regexp_replace(CodCorsa, '.$', '1') AND
               veicolo = true;
31
32          --calcola numero di posti dei veicoli occupati nella
               seconda corsa
33          SELECT count(*)into v2
34          FROM biglietto
35          WHERE coras = regexp_replace(CodCorsa, '.$', '2') AND
               veicolo = true;
36
37          if(v1 >v2) then
38              veicoli:= veicoli-v1;
39          else
40              veicoli:=veicoli-v2;
41          end if;
42
43          --calcola numero di posti di persone occupate nella
               prima corsa
44          SELECT count(*)into p1
45           FROM biglietto
46          WHERE coras = regexp_replace(CodCorsa, '.$', '1');
47
48          --calcola numero di posti di persone occupate nella
               seconda corsa
49          SELECT count(*)into p2
50           FROM biglietto
51           WHERE coras = regexp_replace(CodCorsa, '.$', '2');
52
53          --sottrae ai posti delle corsa intera solo il valore
               pi   grande tra i due
54          if(p1>p2)then
55              persone:= persone - p1;
56          else
57              persone:= persone - p2;
58          end if;
59
60      ELSEIF(CodCorsa like 'C_______S1') then
61
62          select (SELECT nummaxveicoli
63          FROM imbarcazione JOIN corsa ON codiceimbarcazione =
               FKImb
64          WHERE codiceCorsa = CodCorsa)
65          -
66          (SELECT count(*)
67          FROM biglietto
68          WHERE coras = CodCorsa AND veicolo = true)
69          into veicoli;
70
```

```
71          select (SELECT nummaxpersone
72          FROM imbarcazione JOIN corsa ON codiceimbarcazione =
                FKImb
73          WHERE codiceCorsa = CodCorsa)
74          -
75          (SELECT count(*)
76          FROM biglietto
77          WHERE coras = CodCorsa) into persone;
78
79          --Calcolo posti occupati dalla corsa principale
80          CodCorsa := regexp_replace(CodCorsa, '.$', '0');
81
82       veicoli := veicoli -(SELECT count(*)
83      FROM biglietto
84      WHERE coras = CodCorsa AND veicolo = true);
85
86      persone := persone - (SELECT count(*)
87                            FROM biglietto
88                  WHERE coras = CodCorsa);
89
90          ELSEIF(CodCorsa like 'C_______S2') then
91                  select (SELECT nummaxveicoli
92                  FROM imbarcazione JOIN corsa ON
                        codiceimbarcazione = FKImb
93                  WHERE codiceCorsa = CodCorsa)
94                  -
95                  (SELECT count(*)
96                  FROM biglietto
97                  WHERE coras = CodCorsa AND veicolo = true)
98                  into veicoli;
99
100                 select (SELECT nummaxpersone
101                  FROM imbarcazione JOIN corsa ON
                        codiceimbarcazione = FKImb
102                  WHERE codiceCorsa = CodCorsa)
103                  -
104                  (SELECT count(*)
105                  FROM biglietto
106                  WHERE coras = CodCorsa) into persone;
107
108                 --Calcolo posti occupati dalla corsa principale
109                 CodCOrsa := regexp_replace(CodCorsa, '.$', '0')
                        ;
110
111                 veicoli := veicoli -(SELECT count(*)
112                                      FROM biglietto
113                                      WHERE coras = CodCorsa AND
                                            veicolo = true);
```

```
114
115                 persone := persone - (SELECT count(*)
116                                       FROM biglietto
117                                       WHERE coras = CodCorsa);
118
119             END IF;
120         END;
121 $$ LANGUAGE plpgsql;
```

4.7 Viste

Le viste sono delle tabelle virtuali basate da una o più tabelle esistenti nel db, che viene utilizzata come se fosse una tabella fisica. Vengono utilizzate per semplificare delle interrogazioni, per delle interfacce utente, o anche per ragioni di sicurezza

- Le seguente viste servono per far visualizzare il tabellone contente dati delle corse e i porti

```
--qui prelevo la destinazione di ogni corsa

create view tappafinale as(select percorso.corper, percorso.idper, percorso.TipoPercorso, percorso.datascadenza AS dataarrivo, percorso.orarioarrivo,
  porto.nomeporto AS destinazione, porto.citta AS cittadestinazione, porto.nazione AS nazionedestinazione
from percorso JOIN porto ON percorso.idper = porto.idporto
where tipopercorso = 'destinazione');

--qui prelevo la partenza di ogni corsa

create view tappainiziale as(select percorso.corper, percorso.idper, percorso.TipoPercorso, percorso.dataattivazione AS datapartenza, percorso.orariopartenza,
porto.nomeporto AS partenza, porto.citta AS cittapartenza, porto.nazione AS nazionepartenza
from percorso JOIN porto ON percorso.idper = porto.idporto
where tipopercorso = 'partenza');

-- qui controllo se sono presenti degli scali

create view scali as(
select corper, count(*)-2 as scali
from percorso
group by corper);

-- collegando il tutto:
CREATE VIEW Tabellone as(
SELECT CodiceCorsa,costocorsa,scali,corsa.fkcomp AS nomecompagnia,partenza,cittapartenza,nazionepartenza, destinazione,cittadestinazione,nazionedestinazione, datapartenza,dataarrivo,orariopartenza,orarioarrivo,stato,avviso
FROM ((Corsa join tappainiziale as ti on codicecorsa=ti.corper) join tappafinale as tf on ti.corper=tf.corper) join scali on codicecorsa=scali.corper);
```

- La seguenta vista serve per far visualizzare una tabella contente le informazione dei biglietti, col relativo passeggero che li ha acquistato, e di quale corsa

```
CREATE VIEW biglettiAcquistati AS(
select codicecorsa,cf,nome,cognome,datanascita,
   codicebiglietto,tipobiglietto,prezzo,dataacquisto,veicolo
   ,numbagagli,prenotazione
from (corsa join biglietto on codicecorsa=coras) join
   passeggero on cfpass=cf
order by dataacquisto desc
);
```

- La seguente vista serve per far visualizzare tutte quelle corse che non hanno un collegamento completo, ossia non hanno una partenza e/o una destinazione

```
CREATE VIEW CorseIncomplete AS (
select codicecorsa from corsa
except
SELECT codicecorsa FROM corsa left join percorso on
   codicecorsa=corper WHERE tipoPercorso='partenza' or
   tipoPercorso='destinazione'
group by codicecorsa
having count(*)=2);
```

in questo modo è più facile l'eliminazione delle corse incomplete, in questo modo:

```
        delete from corsa where codicecorsa in (select *
           from corseincomplete);
```

5 Esempio Popolazione DB

```
1     INSERT INTO Compagnia (NomeCompagnia, PasswCompagnia,
        Telefono, EmailCompagnia, SitoWeb)
2  VALUES ('MSC', 'msc_passw','3341111111', 'msc@email.it', 'www.
     msc.it'),
3     ('Costa', 'costa_passw', '6249909909', 'costa@email.it', '
        www.costa.it'),
4     ('Royal Caribbean', ' RC_passw', '7924975238', '
        RoyalCaribbean@email.it', 'www. RoyalCaribbean.it'),
5     ('Celebrity Cruises', 'CC_passw', '3682985185', '
        CelebrityCruises@email.it', 'www.CelebrityCruises.it'),
6     ('Holland America Line', 'HAL_passw', '2499099096', '
        HollandAmericaLine@email.it', 'www.HollandAmericaLine.it'
        );
7
8  -- Inserimento dati nella tabella Social
9  INSERT INTO Social (URL, NomeSocial, CompIsc)
10 VALUES ('www.twitter.com/msc', 'Twitter', 'MSC'),
11    ('www.facebook.com/msc','Facebook','MSC');
12
13 -- Inserimento dati nella tabella Imbarcazione
14 INSERT INTO Imbarcazione (CodiceImbarcazione, NomeImbarcazione,
     TipoImbarcazione, CompPoss, NumMaxPersone, NumMaxVeicoli)
15 VALUES ('IT001', 'Poseidone', 'traghetto', 'MSC', 200, 50),
16    ('CFHX', 'Fenice', 'motonave', 'Costa', 24, 0),
17    ('CMT', 'Minotauro', 'aliscafo', 'Royal Caribbean', 300,0),
18    ('POIU', 'Pegaso', 'aliscafo', 'Celebrity Cruises', 100, 0)
        ,
19    ('BVCX', 'Tritone', 'traghetto', 'Royal Caribbean', 180,
        40),
20    ('IT02', 'Zeus', 'motonave', 'Holland America Line', 128,
        0);
21
22 -- Inserimento dati nella tabella Porto
23 INSERT INTO Porto
24 VALUES (1,'Porto di Pozzuoli', 'Pozzuoli', 'Italia'),
25    (2, 'Porto di Genova', 'Genova', 'Italia'),
26    (3, 'Porto di Napoli', 'Napoli', 'Italia'),
27    (4, 'Porto di Trieste', 'Trieste', 'Italia'),
28    (5, 'Porto di Palermo', 'Palermo', 'Italia'),
29    (6,'Porto di Malaga', 'Malaga', 'Spagna'),
30    (7,'Porto La Rochelle', 'La Rochelle', 'Francia'),
31    (8,'Porto di Lagos', 'Lagos', 'Portogallo'),
32    (9, 'Porto di Lisbona', 'Lisbona', 'Portogallo'),
33    (10, 'Porto di Alc ntara', 'Lisbona', 'Portogallo'),
34    (11, 'Porto di Helsinki', 'Helsinki', 'Finlandia'),
35    (12, 'Porto di Turku', 'Turku', 'Finlandia'),
36    (13, 'Porto di Hanko', 'Hanko', 'Finlandia'),
```

```
37      (14, 'Porto di Gdansk', 'Gdansk', 'Polonia'),
38      (15, 'Porto di Vancouver', 'Vancouver', 'Canada'),
39      (16, 'Porto di Sydney', 'Sydney', 'Australia'),
40      (17, 'Porto di Haifa', 'Haifa', 'Israele'),
41      (18, 'Porto di Jeddah', 'Jeddah', 'Arabia Saudita'),
42      (19, 'Porto di Beirut', 'Beirut', 'Libano');
43
44  -- Inserimento dati nella tabella Passeggero
45  INSERT INTO Passeggero (CF, Nome, Cognome, DataNascita, Email,
        Passw)
46  VALUES ('6BZ6AHENEWXB47DV', 'Mario', 'Rossi', '2010-07-02', '
        mario@email.it', 'mario'),
47         ('TFE5764PZ2PDLWV5', 'Paolo', 'Moccia', '1990-01-01', '
            paolo@email.it', 'paolo'),
48         ('VNSU7JQSGL6Y6PKK', 'Martina', 'Gialla', '1995-04-04',
            'martina@email.it', 'martina'),
49         ('UPS5K7YBWY93WUX4', 'Federico', 'Chiesa', '2012-11-04',
             'federico@email.it', 'federico'),
50         ('LB8BGHWEKG5G24UE', 'Zhou', 'Fei Wa', '2001-01-02', '
            zhou@email.it', 'zhou');
51
52
53
54  -- Inserimento dati nella tabella Corsa
55  INSERT INTO Corsa (CodiceCorsa, CostoCorsa, Avviso, Stato,
        FKImb, FKComp)
56  VALUES  ('C0000006S0', 220.00, '', 'regolare', 'IT001', 'MSC'),
57      ('C0000006S1', 110.00, '', 'regolare', 'IT001', 'MSC'),
58      ('C0000006S2', 110.00, '', 'regolare', 'IT001', 'MSC'),
59      ('C0000007S0', 220.00, '', 'regolare', 'IT001', 'MSC'),
60      ('C0000007S1', 110.00, '', 'regolare', 'IT001', 'MSC'),
61      ('C0000007S2', 110.00, '', 'regolare', 'IT001', 'MSC'),
62      ('C0000010S0', 60.00, 'maltempo, posticipato di un giorno',
            'ritardo', 'IT02', 'Holland America Line'),
63      ('C0000020S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
64      ('C0000021S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
65      ('C0000022S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
66      ('C0000023S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
67      ('C0000024S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
68      ('C0000025S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
69      ('C0000026S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
70      ('C0000027S0', 180.00, '', 'regolare', 'CFHX', 'Costa'),
71      ('C0000018S0', 90.75, 'sciopero', 'annullato', 'CMT', '
            Royal Caribbean'),
72      ('C0000018S1', 45.38, 'sciopero', 'annullato', 'CMT', '
            Royal Caribbean'),
73      ('C0000018S2', 45.38, 'sciopero', 'annullato', 'CMT', '
            Royal Caribbean'),
```

```
74      ('C0000001S0', 200.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
75      ('C0000001S1', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
76      ('C0000001S2', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
77      ('C0000002S0', 200.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
78      ('C0000002S1', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
79      ('C0000002S2', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
80      ('C0000003S0', 200.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
81      ('C0000003S1', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
82      ('C0000003S2', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
83      ('C0000004S0', 200.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
84      ('C0000004S1', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line'),
85      ('C0000004S2', 100.00, '', 'regolare', 'IT02', 'Holland
           America Line');
86
87
88          -- Inserimento dati nella tabella Percorso
89  INSERT INTO Percorso (CorPer, IdPer,TipoPercorso,
       OrarioPartenza, OrarioArrivo, DataAttivazione, DataScadenza)
90  VALUES ('C0000001S0', 1, 'partenza', '08:10:00', '21:25:00', '
       2024-02-05', '2024-02-05'),
91  ('C0000001S0', 10, 'scalo', '18:00:00', '05:07:00', '2024-02-06
       ', '2024-02-08'),
92  ('C0000001S0', 15, 'destinazione', '05:07:00', '05:07:00', '
       2024-02-08', '2024-02-08'),
93  ('C0000001S1', 1, 'partenza', '08:10:00', '21:25:00', '
       2024-02-05', '2024-02-05'),
94  ('C0000001S1', 10, 'destinazione', '18:00:00', '18:00:00', '
       2024-02-06', '2024-02-06'),
95  ('C0000001S2', 10, 'partenza', '18:00:00', '05:07:00', '
       2024-02-06', '2024-02-08'),
96  ('C0000001S2', 15, 'destinazione', '05:07:00', '05:07:00', '
       2024-02-08', '2024-02-08'),
97  ('C0000002S0', 1, 'partenza', '08:10:00', '21:25:00', '
       2024-02-12', '2024-02-12'),
98  ('C0000002S0', 10, 'scalo', '18:00:00', '05:07:00', '2024-02-13
       ', '2024-02-15'),
99  ('C0000002S0', 15, 'destinazione', '05:07:00', '05:07:00', '
       2024-02-15', '2024-02-15'),
```

```
100 ('C0000002S1', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-12', '2024-02-12'),
101 ('C0000002S1', 10, 'destinazione', '18:00:00', '18:00:00', '
    2024-02-13', '2024-02-13'),
102 ('C0000002S2', 10, 'partenza', '18:00:00', '05:07:00', '
    2024-02-13', '2024-02-15'),
103 ('C0000002S2', 15, 'destinazione', '05:07:00', '05:07:00', '
    2024-02-15', '2024-02-15'),
104 ('C0000003S0', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-02', '2024-02-02'),
105 ('C0000003S0', 10, 'scalo', '18:00:00', '05:07:00', '2024-02-03
    ', '2024-02-05'),
106 ('C0000003S0', 15, 'destinazione', '05:07:00', '05:07:00', '
    2024-02-05', '2024-02-05'),
107 ('C0000003S1', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-02', '2024-02-02'),
108 ('C0000003S1', 10, 'destinazione', '18:00:00', '18:00:00', '
    2024-02-03', '2024-02-03'),
109 ('C0000003S2', 10, 'partenza', '18:00:00', '05:07:00', '
    2024-02-03', '2024-02-05'),
110 ('C0000003S2', 15, 'destinazione', '05:07:00', '05:07:00', '
    2024-02-05', '2024-02-05'),
111 ('C0000004S0', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-09', '2024-02-09'),
112 ('C0000004S0', 10, 'scalo', '18:00:00', '05:07:00', '2024-02-10
    ', '2024-02-12'),
113 ('C0000004S0', 15, 'destinazione', '05:07:00', '05:07:00', '
    2024-02-12', '2024-02-12'),
114 ('C0000004S1', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-09', '2024-02-09'),
115 ('C0000004S1', 10, 'destinazione', '18:00:00', '18:00:00', '
    2024-02-10', '2024-02-10'),
116 ('C0000004S2', 10, 'partenza', '18:00:00', '05:07:00', '
    2024-02-10', '2024-02-12'),
117 ('C0000004S2', 15, 'destinazione', '05:07:00', '05:07:00', '
    2024-02-12', '2024-02-12'),
118 ('C0000006S0', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-16', '2024-02-16'),
119 ('C0000006S0', 10, 'scalo', '18:00:00', '06:13:00', '2024-02-17
    ', '2024-02-19'),
120 ('C0000006S0', 15, 'destinazione', '06:13:00', '06:13:00', '
    2024-02-19', '2024-02-19'),
121 ('C0000006S1', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-16', '2024-02-16'),
122 ('C0000006S1', 10, 'destinazione', '18:00:00', '18:00:00', '
    2024-02-17', '2024-02-17'),
123 ('C0000006S2', 10, 'partenza', '18:00:00', '06:13:00', '
    2024-02-17', '2024-02-19'),
```

```
124 ('C0000006S2', 15, 'destinazione', '06:13:00', '06:13:00', '
    2024-02-19', '2024-02-19'),
125 ('C0000007S0', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-23', '2024-02-23'),
126 ('C0000007S0', 10, 'scalo', '18:00:00', '06:13:00', '2024-02-24
    ', '2024-02-26'),
127 ('C0000007S0', 15, 'destinazione', '06:13:00', '06:13:00', '
    2024-02-26', '2024-02-26'),
128 ('C0000007S1', 1, 'partenza', '08:10:00', '21:25:00', '
    2024-02-23', '2024-02-23'),
129 ('C0000007S1', 10, 'destinazione', '18:00:00', '18:00:00', '
    2024-02-24', '2024-02-24'),
130 ('C0000007S2', 10, 'partenza', '18:00:00', '06:13:00', '
    2024-02-24', '2024-02-26'),
131 ('C0000007S2', 15, 'destinazione', '06:13:00', '06:13:00', '
    2024-02-26', '2024-02-26'),
132 ('C0000010S0', 3, 'partenza', '09:00:00', '15:00:00', '
    2024-02-29', '2024-02-29'),
133 ('C0000010S0', 5, 'destinazione', '15:00:00', '15:00:00', '
    2024-02-29', '2024-02-29'),
134 ('C0000020S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-03', '2024-02-03'),
135 ('C0000020S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-03', '2024-02-03'),
136 ('C0000021S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-10', '2024-02-10'),
137 ('C0000021S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-10', '2024-02-10'),
138 ('C0000022S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-17', '2024-02-17'),
139 ('C0000022S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-17', '2024-02-17'),
140 ('C0000023S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-24', '2024-02-24'),
141 ('C0000023S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-24', '2024-02-24'),
142 ('C0000024S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-04', '2024-02-04'),
143 ('C0000024S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-04', '2024-02-04'),
144 ('C0000025S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-11', '2024-02-11'),
145 ('C0000025S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-11', '2024-02-11'),
146 ('C0000026S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-18', '2024-02-18'),
147 ('C0000026S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-18', '2024-02-18'),
```

```
148 ('C0000027S0', 18, 'partenza', '07:43:00', '23:46:00', '
    2024-02-25', '2024-02-25'),
149 ('C0000027S0', 7, 'destinazione', '23:46:00', '23:46:00', '
    2024-02-25', '2024-02-25'),
150 ('C0000018S0', 6, 'partenza', '15:00:00', '23:19:00', '
    2024-02-28', '2024-02-28'),
151 ('C0000018S0', 11, 'scalo', '13:00:00', '20:25:00', '2024-02-29
    ', '2024-02-29'),
152 ('C0000018S0', 14, 'destinazione', '20:25:00', '20:25:00', '
    2024-02-29', '2024-02-29'),
153 ('C0000018S1', 6, 'partenza', '15:00:00', '23:19:00', '
    2024-02-28', '2024-02-28'),
154 ('C0000018S1', 11, 'destinazione', '13:00:00', '13:00:00', '
    2024-02-29', '2024-02-29'),
155 ('C0000018S2', 11, 'partenza', '13:00:00', '20:25:00', '
    2024-02-29', '2024-02-29'),
156 ('C0000018S2', 14, 'destinazione', '20:25:00', '20:25:00', '
    2024-02-29', '2024-02-29');
157
158
159
160 -- Inserimento dati nella tabella Biglietto
161 INSERT INTO biglietto VALUES
162 ('A3O42', 'intero',null , '2024-02-04', false, 2, true, '
    VNSU7JQSGL6Y6PKK', 'C0000001S0'),
163 ('QC4M3', 'intero',null , '2024-02-04', false, 0, true, '
    VNSU7JQSGL6Y6PKK', 'C0000001S0'),
164 ('I8CEK', 'intero',null, '2024-02-04', false, 0, true, '
    VNSU7JQSGL6Y6PKK', 'C0000001S0'),
165 ('SOGSB', 'ridotto', null, '2024-02-04', false, 0, true, '
    VNSU7JQSGL6Y6PKK', 'C0000001S0'),
166 ('LGSK3', 'ridotto', null, '2024-02-04', false, 0, true, '
    VNSU7JQSGL6Y6PKK', 'C0000001S0'),
167 ('BU81Y', 'intero',null , '2024-02-04', true, 2, false, '
    LB8BGHWEKG5G24UE', 'C0000006S2'),
168 ('Y9Z2Y', 'ridotto', null, '2024-02-04', true, 0, false, '
    LB8BGHWEKG5G24UE', 'C0000006S2');
```