

Documentazione Basi Di Dati

Compagnie di navigazione

Anno accademico 2023-2024



UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Professori:

Silvio barra

Porfiro Tramontana

Alunni:

Francesco Sorrentino N86004817

Antonio Paudice N86004566

Deyvid Dimitrov Manolov N86004796

Indice

1	Introduzione	2
1.1	Analisi dei requisiti	2
2	Progettazione Concettuale	3
2.1	Class diagram UML	3
2.2	Analisi delle ridondanze	3
2.3	Rimozione delle Aggregazioni e delle Composizioni	3
2.4	Rimozione degli attributi multipli	3
2.5	Rimozione degli Attributi Strutturati	3
2.6	Accorpamento/Partizionamento di Entità/Relazioni	3
2.7	Identificazione Chiavi Primarie	4
2.8	Class diagram UML ristrutturato	4
2.9	Class diagram ER ristrutturato	5
2.10	Dizionario delle Classi	5
2.10.1	Descrizione delle Classi	5
2.10.2	Dizionario degli Attributi	6
2.10.3	Dizionario delle Associazioni	7
2.10.4	Dizionario dei Vincoli	8
3	Progettazione Logica	9
3.1	Schema Logico	9
4	Progettazione Fisica	10
4.1	Svuotare il database	10
4.2	Creazione tabelle	10
4.3	Creazione dei Trigger	12
4.3.1	codice	12
4.4	Popolazione del Database	13

1 Introduzione

La base di dati deve occuparsi della gestione di un sistema di corse offerte dalle compagnie di navigazione, con cadenza giornaliera, orari di partenza e arrivo, e operate solo alcuni giorni dell'anno distinguendo inoltre vari tipi di natanti, prezzi diversi o se presenti scali.

Il sistema deve gestire il tabellone riguardante queste corse, rendendole visibili ai clienti oltre che modificabili dalle compagnie.

1.1 Analisi dei requisiti

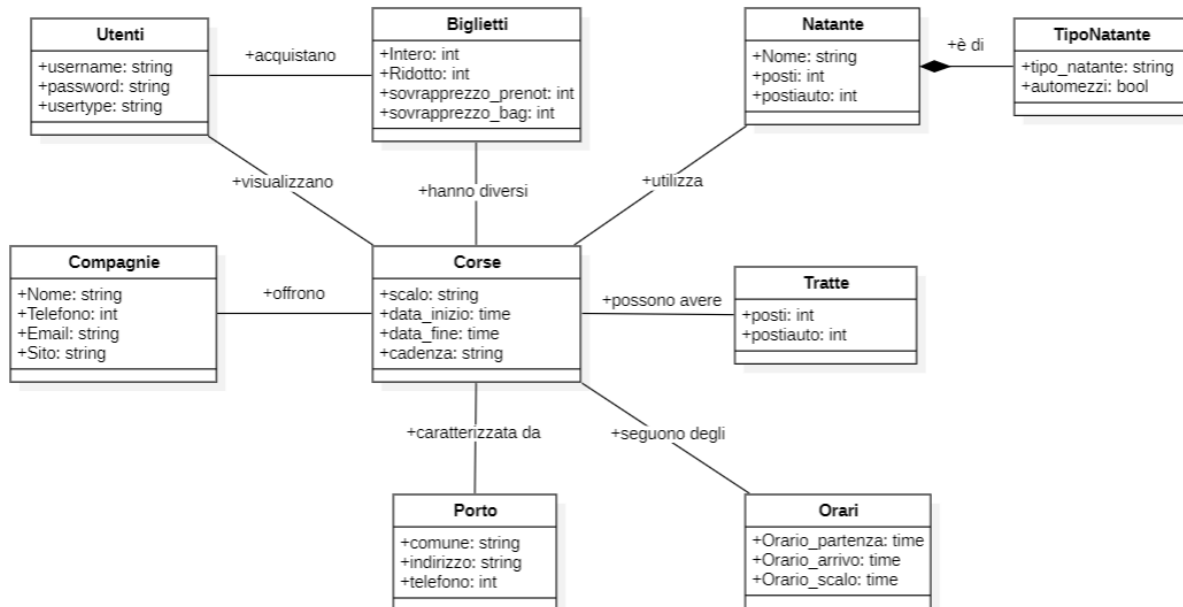
Per l'analisi dei requisiti procediamo identificando le entità e le associazioni presenti nel database, qui elencate:

- **Entità "Corse"**: Rappresenta il tabellone delle corse, che possono essere modificate, aggiunte o rimosse se si è una compagnia o visualizzate se si è un cliente. qui sono contenute le informazioni riguardo i prezzi dei biglietti, o l'eventuale ritardo/cancellazione di una corsa
- **Entità "Compagnia"**: Descrive le compagnie che possono modificare, aggiungere o rimuovere le Corse, ed è compreso di informazioni come "nome", "telefono" e "social"
- **Entità "Utenti"**: Rappresenta la tipologia di utenti che hanno accesso al sistema, distinguendosi in "Clienti" e "Compagnie"
- **Entità "Tratte"**: Descrive le varie tratte, indicando un orario di partenza, uno di arrivo, oltre a posti ed eventuali posti auto fra le varie tratte
- **Entità "Natanti"**: Descrive i singoli natanti, includendo informazioni quali "nome", numero di "posti per passeggeri" e numero di "posti auto"
- **Entità "TipoNatante"**: indica il tipo di natante utilizzato, facendo distinzione fra "traghetti", "aliscafi" e "motonavi"
- **Entità "Porto"**: Indica il porto, d'arrivo o di partenza, oltre a altri dati come "comune", "indirizzo", e "telefono"
- **Associazione "Utenti-Corse"**: Permette agli utenti di accedere al tabellone delle corse e, in base al tipo di utente, effettuare diverse azioni
- **Associazione "Corse-Porto"**: Permette alle Corse di avere un porto da cui partire e uno a cui arrivare
- **Associazione "Corse-Tratte"**: Permette alle Corse di specificare diverse possibili tratte intermedie o finali
- **Associazione "Compagnia-Corse"**: Permette a una Compagnia di gestire una o più corse
- **Associazione "Compagnia-Natante"**: Indica quali natanti sono posseduti da una Compagnia
- **Associazione "Corse-Natante"**: Indica il natante utilizzato per una corsa specifica, fornendone anche il tipo
- **Associazione "Natante-TipoNatante" (Composizione)**: Specifica il tipo di un determinato natante utilizzato

2 Progettazione Concettuale

Sulla base dei requisiti analizzati in precedenza, costruiamo un primo modello concettuale Class Diagram

2.1 Class diagram UML



2.2 Analisi delle ridondanze

Nel diagramma non sono presenti ridondanze

2.3 Rimozione delle Aggregazioni e delle Composizioni

Nel diagramma non vengono rimosse Aggregazioni e Composizioni

2.4 Rimozione degli attributi multipli

Nel diagramma non sono presenti attributi multipli

2.5 Rimozione degli Attributi Strutturati

Nel diagramma non sono presenti Attributi Strutturati

2.6 Accorpamento/Partizionamento di Entità/Relazioni

La classe "Biglietti" è stata accorpata alla classe "Corse"

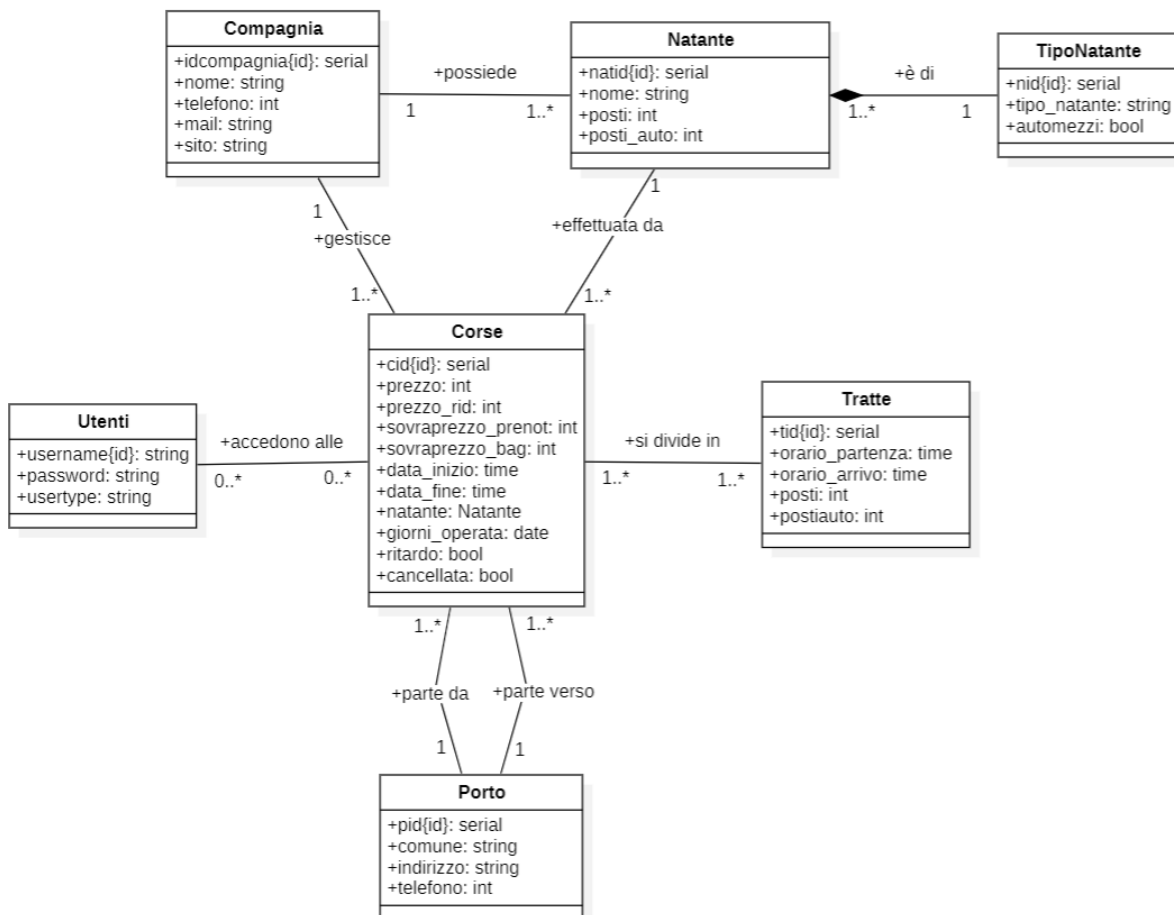
La classe "Orari" è stata accorpata alla classe "Tratte"

2.7 Identificazione Chiavi Primarie

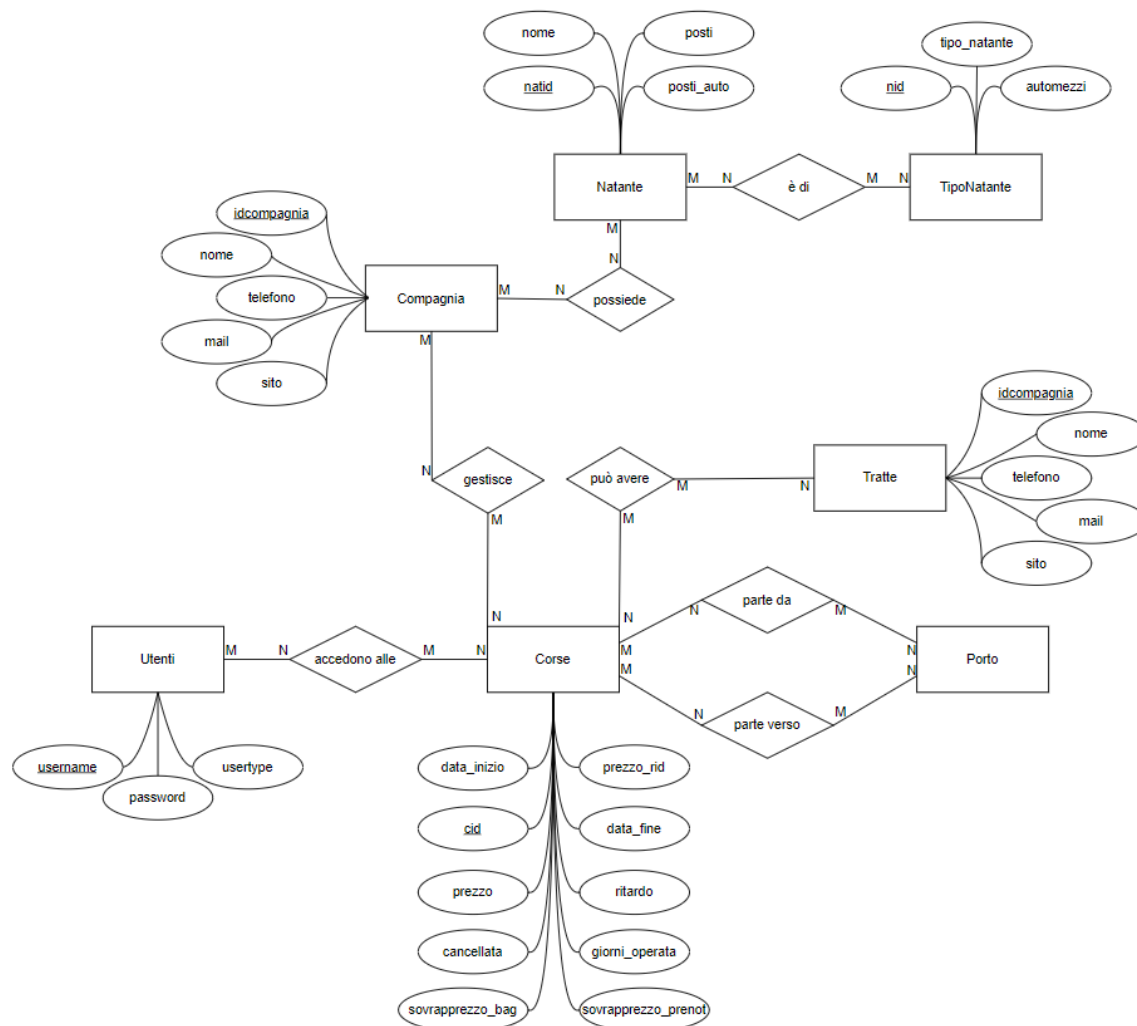
Sono state assegnate chiavi primarie a ciascuna delle entità presenti, in modo tale da riusci identificare in maniera univoca ciascuna delle istanze presenti nel database. Ecco le modifiche apportate:

- **idcompagnia:** Aggiunto come chiave primaria nella tabella "compagnia"
- **username:** Utilizzato come chiave primaria nella tabella "Utenti", poiché campo univoco e non nullo, rendendolo un identificatore affidabile per gli Utenti
- **cid:** Aggiunto come chiave primaria nella tabella "Corse"
- **natid:** Aggiunto come chiave primaria nella tabella "Natante"
- **nid:** Aggiunto come chiave primaria nella tabella "TipoNatante"
- **tid:** Aggiunto come chiave primaria nella tabella "Tratte"
- **pid:** Aggiunto come chiave primaria nella tabella "Porto"

2.8 Class diagram UML ristrutturato



2.9 Class diagram ER ristrutturato



2.10 Dizionario delle Classi

2.10.1 Descrizione delle Classi

Utenti: Descrive il profilo degli Utenti, incluso il tipo di utente (Compagnia o Cliente) e l'username

Compagnia: Descrive tutte le informazioni complete della Compagnia che gestisce le corse

Corse: Descrive tutte le informazioni riguardanti le Corse, quali il prezzo del loro biglietto (compreso di sovrapprezzo, se c'è) date e orari di partenza e stato della corsa

Porto: Descrive tutte le informazioni riguardanti il porto, come comune, indirizzo e numero di telefono

Natante: Descrive le informazioni riguardo il natante utilizzato, come nome, posti e eventuali posti auto

TipoNatante: Descrive informazioni aggiuntive riguardo il natante, come il tipo e la presenza di automezzi

Tratte: Descrive tutte le informazioni riguardanti le tratte, e comprende gli orari di partenza e arrivo, i posti e i posti auto del natante calcolati a ogni tratta

2.10.2 Dizionario degli Attributi

Classi	Attributi
Utenti	Username(string) : identificatore univoco dell'utente Password(string) : password associata all'utente Usertype(string) : tipologia di utente
Compagnia	Idcompagnia(serial) : identificatore seriale della compagnia Nome(string) : nome della compagnia Telefono(int) : numero di telefono della compagnia Mail(string) : e-mail della compagnia Sito(string) : sito web della compagnia
Corse	Cid(serial) : identificatore seriale delle corse Prezzo(int) : prezzo del biglietto della corsa Prezzo_rid(int) : prezzo ridotto del biglietto Sovrapprezzo_prenot(int) : sovrapprezzo biglietto in caso di prenotazione Sovrapprezzo_bag(int) : sovrapprezzo biglietto in caso di bagaglio Data_inizio(time) : data inizio corsa Data_fine(time) : data fine corsa Natante(Natante) : natante usato per le singole corse Giorni_operata(date) : Ritardo(bool) : mostra se la corsa ha fatto ritardo Cancellata(bool) : mostra se la corsa è stata cancellata
Porto	Pid(serial) : identificatore seriale del porto Comune(string) : comune di appartenenza del porto Indirizzo(string) : indirizzo del porto Telefono(int) : numero di telefono del porto
Natante	Natid(serial) : identificatore seriale del natante Nome(string) : nome del natante Posti(int) : posti disponibili nel natante Posti_auto(int) : posti auto disponibili nel natante
TipoNatante	Nid(serial) : identificatore seriale del tipo del natante Tipo_natante(string) : tipo di natante, scegliendo fra Automezzi(bool) : indica la possibilità di trasportare automezzi in quel tipo di natante
Tratte	Tid(serial) : identificatore seriale della tratta Orario_partenza(time) : l'orario di partenza da una determinata tratta Orario_arrivo(time) : l'orario di arrivo a una determinata tratta Posti(int) : numero posti disponibili a una determinata tratta Postiauto(int) : numero posti auto disponibili a una determinata tratta

2.10.3 Dizionario delle Associazioni

Associazione	Descrizione	Classi coinvolte
Accedono alle	Indica le corse a cui gli utenti accedono	Utenti[0...*]: da 0 a N utenti possono accedere alle corse Corse[0...*]: da 0 a N corse possono essere visualizzate dagli utenti
Gestisce	Indica le corse gestite da una compagnia	Compagnia[1...1]: Una corsa è gestita da una sola compagnia Corse[1...*]: Una compagnia gestisce una o più corse
Possiede	Indica i natanti posseduti da una compagnia	Compagnia[1...1]: Un natante è posseduto da una sola compagnia Natante[1...*]: Una compagnia possiede da 1 o più natanti
Effettuata da	Indica il natante utilizzato per una corsa	Corse[1...*]: 1 o più corse possono essere effettuate da un natante Natante[1...1]: Una corsa è effettuata da un solo natante
Parte da	Indica il porto di partenza di una corsa	Corse[1...*]: Da 1 o più corse partono da un porto Porto[1...1]: Una corsa parte da un solo porto
Parte verso	Indica il porto di arrivo di una corsa	Corse[1...*]: 1 o più corse partono verso un porto Porto[1...1]: Una corsa parte verso un solo porto
Si divide in	Indica le tratte che una corsa può avere	Corse[1...*]: Una corsa si divide in in 1 o più tratte Tratte[0...*]: Una tratta è attraversata da 0 o più corse
è di	Indica il tipo di natante di un natante	Natanti[1...*]: Uno o più natanti sono di un tipo TipoNatante[1...1]: Ogni natante è di un solo tipo

2.10.4 Dizionario dei Vincoli

Classe	Descrizione vincoli.
Utenti	username: Chiave Primaria usertype: Dev'essere necessariamente "User" o "Compagnia"
Compagnia	idcompagnia: Chiave Primaria mail: Non nullo, unico, scritto in formato: _%@_%._%
Corse	cid: Chiave Primaria natante: Chiave composta, riferimento esterno alla tabella "Natante"
Porto	pid: Chiave Primaria
Natante	natid: Chiave Primaria
TipoNatante	nid: Chiave Primaria tipo_natante: Dev'essere necessariamente "Traghetto", "Aliscafo" o "Motonave"
Tratte	tid: Chiave Primaria

3 Progettazione Logica

3.1 Schema Logico

Nel seguente schema logico, le chiavi primarie sono indicate in **grassetto** e le chiavi esterne con una sottolineatura

Utenti: **username**, password, usertype

Compagnia: **idcompagnia**, nome, telefono, mail, sito

Corse: **cid**, prezzo, prezzo_rid, sovrapprezzo_prenot, sovrapprezzo_bag, data_inizio, data_fine, natante, giorni_operata, ritardo, cancellata

Porto: **pid**, comune, indirizzo, telefono

Natante: **natid**, nome, posti, posti_auto

Tratte: **tid**, cid, orario_partenza, orario_arrivo, posti, postiauto

TipoNatante: **nid**, tipo_natante, automezzi

4 Progettazione Fisica

In questa sezione descriveremo la trasformazione del modello concettuale del sistema di navigazione nella sua implementazione fisica usando il DBMS PostgreSQL.

4.1 Svuotare il database

Svuotiamo il database per fare i test.

```
DROP TRIGGER IF EXISTS tratte ON Corse;
DROP FUNCTION IF EXISTS divide_tratte();
DROP TABLE IF EXISTS Compagnia CASCADE;
DROP TABLE IF EXISTS Utenti CASCADE;
DROP TABLE IF EXISTS Tratte CASCADE;
DROP TABLE IF EXISTS Natanti CASCADE;
DROP TABLE IF EXISTS TipoNatante CASCADE;
DROP TABLE IF EXISTS Corse CASCADE;
DROP TABLE IF EXISTS Porto CASCADE;
```

4.2 Creazione tabelle

Passiamo adesso alla creazione di tutte le tabelle, ognuna con le proprie chiavi primarie e secondarie.

```
CREATE TABLE IF NOT EXISTS TipoNatante (
    nid SERIAL PRIMARY KEY NOT NULL,
    tipo_natante VARCHAR(20) NOT NULL,
    automezzi BOOLEAN NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS Utenti (
    username VARCHAR(16) PRIMARY KEY NOT NULL,
    password TEXT NOT NULL,
    usertype VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS Compagnia (
    id SERIAL PRIMARY KEY NOT NULL,
    nome TEXT,
    telefono TEXT,
    email TEXT,
    sito TEXT
);
```

```
CREATE TABLE IF NOT EXISTS Porto (
    pid SERIAL PRIMARY KEY NOT NULL,
```

```
    comune TEXT NOT NULL,
    indirizzo TEXT NOT NULL,
    tel TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS Natanti (
    id SERIAL PRIMARY KEY NOT NULL,
    nome TEXT NOT NULL,
    typeid INTEGER NOT NULL REFERENCES TipoNatante(nid),
    posti INTEGER NOT NULL,
    postiauto INTEGER DEFAULT 0,
    idcompagnia INTEGER NOT NULL REFERENCES Compagnia(id)
);

CREATE TABLE IF NOT EXISTS Corse (
    cid SERIAL PRIMARY KEY NOT NULL,
    nid INTEGER NOT NULL REFERENCES Natanti(id),
    idCompagnia INTEGER NOT NULL REFERENCES Compagnia(id),
    porto_partenza INTEGER NOT NULL REFERENCES Porto(pid),
    porto_arrivo INTEGER NOT NULL REFERENCES Porto(pid),
    scalo INTEGER DEFAULT NULL, --ARRAY[20],
    orario_partenza TIMESTAMP NOT NULL,
    orario_arrivo TIMESTAMP NOT NULL,
    orario_scalo TIMESTAMP DEFAULT NULL,
    data_inizio TIMESTAMP NOT NULL,
    data_fine TIMESTAMP NOT NULL,
    cadenza VARCHAR(10) [],
    prezzo REAL,
    prezzo_rid REAL,
    sovrapprezzo_prenot REAL,
    sovrapprezzo_bag REAL
);

CREATE TABLE IF NOT EXISTS Tratte(
    tid SERIAL PRIMARY KEY NOT NULL,
    cid INTEGER NOT NULL REFERENCES Corse(cid),
    porto_partenza INTEGER NOT NULL REFERENCES Porto(pid),
    orario_partenza TIMESTAMP NOT NULL,
    porto_arrivo INTEGER NOT NULL REFERENCES Porto(pid),
    orario_arrivo TIMESTAMP NOT NULL,
    posti INTEGER,
    postiauto INTEGER
);
```

4.3 Creazione dei Trigger

In questa sezione analizziamo i trigger e le funzioni ad essi annesse.

Il trigger `dividi_tratte()` è eseguito all'inserimento di una nuova corsa nella tabella `Corse` e ha come scopo la generazione di ogni tratta (da porto a porto) e l'inserimento di quest'ultima nella tabella `Tratte` con i relativi posti a sedere liberi e posti auto liberi

Il trigger `addCompagnia()` viene eseguito all'inserimento di un nuovo utente nella tabella `Utenti`. Se l'utente è di tipo 'Compagnia', viene automaticamente aggiunta una nuova riga nella tabella `Compagnia`.

4.3.1 codice

```
CREATE OR REPLACE FUNCTION divide_tratte()
  RETURNS TRIGGER AS
$$
DECLARE
  n_id INTEGER;
  posti_totali INTEGER;
  posti_auto INTEGER;
BEGIN
  SELECT nid INTO n_id FROM Corse WHERE cid = NEW.cid;
  SELECT posti INTO posti_totali FROM Natanti WHERE id = n_id;
  SELECT postiauto INTO posti_auto FROM Natanti WHERE id = n_id;
  INSERT INTO Tratte(cid, porto_partenza, orario_partenza, porto_arrivo, orario_arrivo, posti,postiauto)
  VALUES (NEW.cid, NEW.porto_partenza, NEW.orario_partenza, NEW.porto_arrivo, NEW.orario_arrivo,
  (posti_totali - posti_auto), NEW.postiauto);
  IF NEW.scalo IS NOT NULL THEN
    INSERT INTO Tratte(cid, porto_partenza,orario_partenza, porto_arrivo, orario_arrivo, posti,postiauto)
    VALUES (NEW.cid, NEW.porto_partenza,NEW.orario_partenza, NEW.porto_arrivo, NEW.orario_arrivo,
    (posti_totali - posti_auto), NEW.postiauto);
  END IF;
  RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION addCompagnia()
  RETURNS TRIGGER AS
$$
BEGIN
  IF NEW.usertype = 'Compagnia' THEN
    INSERT INTO Compagnia(nome) VALUES (NEW.username);
  END IF;
  RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER tratte
  AFTER INSERT ON Corse
  FOR EACH ROW
EXECUTE FUNCTION divide_tratte();
```

```
CREATE TRIGGER addCompagnia
    AFTER INSERT ON Utenti
    FOR EACH ROW
    EXECUTE FUNCTION addCompagnia();
```

4.4 Popolazione del Database

In questa ultima sezione, ci occupiamo di riempire i Database con i dati che ci servono, in questo caso con gli account delle compagnie, i natanti presenti e alcune tratte già esistenti.

```
INSERT INTO Utenti (username, password, usertype) VALUES ('hemonios', 'hemo1234', 'Utente');
INSERT INTO Utenti (username, password, usertype) VALUES ('medmar', '12345678', 'Compagnia');
INSERT INTO Utenti (username, password, usertype) VALUES ('Compagnia1', '123456', 'Compagnia');
INSERT INTO TipoNatante (tipo_natante, automezzi) VALUES ('Traghetto', true), ('Aliscafo', false),
('Motonave', false);
INSERT INTO Porto(comune, indirizzo,tel) VALUES('Molo Beverello ', 'Molo Beverello, 1, 80133 Napoli
NA', '0812346789');
INSERT INTO Porto(comune, indirizzo,tel) VALUES('Porto dí Ischia', 'Via Porto, 25, 80077 Ischia NA',
'0811111111');
INSERT INTO Porto(comune, indirizzo,tel) VALUES('Porto Turistico di Capri', 'Marina di Caterola,
80073 Capri NA', '0816381459');
INSERT INTO Porto(comune, indirizzo,tel) VALUES('Porto di Pozzuoli', 'Lungomare C. Colombo, 17,
80078 Pozzuoli NA', '0812556678');
INSERT INTO Natanti(nome,typeid,posti,postiauto, idcompagnia) VALUES ('Meraviglia', 1, 300,50,1);
INSERT INTO Natanti(nome,typeid,posti,idcompagnia) VALUES ('Amilcare', 2, 320,1);
INSERT INTO Natanti(nome,typeid,posti,idcompagnia) VALUES ('Astra', 3, 150,1);
INSERT INTO Natanti(nome,typeid,posti,idcompagnia) VALUES ('Miryam', 2, 300,2);
```