Relazione per Progetto di Basi di Dati e Laboratorio

Baldo Massimiliano
Zuccato Francesco

2020-2021

Introduzione

L'obiettivo di questo elaborato è di descrivere tutto il processo di creazione di un database, partendo da un testo scritto in linguaggio naturale fino a raggiungere una analisi dei dati residenti nella base di dati.

Tutto il codice realizzato è presente su GitHub:
https://github.com/massimilianobaldo/ProjectDB

1 Testo originale

Si voglia modellare il seguente insieme di informazioni riguardanti un sistema di prenotazione dei voli aerei.

- Ogni aeroporto sia identificato univocamente da un codice e sia caratterizzato da un nome, da una città e da una nazione. Si assuma che in una data città non vi possano essere due aeroporti con lo stesso nome (e che ogni città sia identificata univocamente dal nome).
- Ogni tipo di aeroplano sia identificato dal nome e sia caratterizzato dal nome dell'azienda costruttrice, dal numero massimo di posti a sedere e dall'autonomia di volo.
- Ogni aeroplano sia identificato da un codice e sia caratterizzato dal tipo (di aeroplano) e dal numero di posti a sedere messi effettivamente a disposizione per i passeggeri. Si assuma che uno stesso aeroplano possa essere utilizzato da compagnie diverse.
- Per ogni aeroporto, sia definito l'insieme dei tipi di aeroplano che possono decollare/atterrare.
- Ogni volo sia caratterizzato da un codice, che lo identifica univocamente, dalla compagnia aerea che offre il volo, dai giorni della settimana in cui viene offerto (si assuma che ogni volo sia effettuato al più una volta in un dato giorno, che gli orari di ogni volo siano sempre gli stessi e che ogni volo, anche quando costituito da più tratte, inizi e termini nella stessa giornata) e, per ogni classe (business, economica, ..), dal prezzo del biglietto (si assuma che biglietti di un dato volo relativi alla stessa classe abbiano tutti lo stesso prezzo).
- Ogni volo sia costituito da un insieme di tratte intermedie (una tratta sia una porzione di volo priva di scali intermedi), ciascuna identificata da un numero progressivo (prima tratta del volo, seconda tratta del volo, ..). Ogni tratta sia caratterizzata da un aeroporto di partenza, un aeroporto di arrivo, un orario previsto di partenza e un orario previsto di arrivo.
- Per ogni specifica istanza di tratta, la data in cui avrà luogo (ad esempio, la tratta Trieste-Monaco del 25 febbraio 2012), l'aeroplano utilizzato (si assuma che su una stessa tratta possano essere utilizzati in date diverse aeroplani diversi) e il numero di posti ancora disponibili.
- Ogni prenotazione relativa ad una certa istanza di tratta sia identificata dal posto prenotato (numero più lettera; ad esempio, posto 16D) e sia caratterizzata dal nome, dal cognome e dal recapito telefonico della persona che ha prenotato il posto.

2 Raccolta e Analisi dei Requisiti

2.1 Glossario dei Termini

Termine	Descrizione	Relazioni
Aeroporto	Luogo dove atterrano e decollano gli aerei. Il suo nome è unico dentro la città	Tipo di Aeroplano Tratta
Tipo di Aeroplano	La tipologia dell'aeroplano.	Aeroporto Aeroplano
Aeroplano	Mezzo con cui avvengono i voli. Può essere usato da più compagnie	Compagnia Aerea Istanza di Tratta Tipo di Aeroplano
Volo	Insieme di una o più tratte offerte da una compagnia aerea. Si ripete in determinati giorni della settimana.	Compagnia Aerea Tratta
Classe	Insieme di tutte le classi che possono essere presenti sugli aerei	Volo
Compagnia Aerea	Azienda che utilizza gli aeroplani (non li possiede in modo esclusivo) e offre i voli	Aeroplano Volo
Tratta	Porzione di volo priva di scali intermedi. Può essere porzione di più voli. Rispetto ad ogni volo ha associato un numero progressivo.	Aeroporto Istanza di Tratta Volo
Istanza di Tratta	Istanza effettiva di una certa tratta che avviene in una determinata data e con un determinato aeroplano.	Prenotazione Tratta Aeroplano
Prenotazione	Prenotazione per un cliente di un volo (che può essere composto da più tratte). Per ogni istanza di tratta è associato il relativo posto nell'aeroplano.	Istanza di Tratta Cliente
Cliente	Acquirente di una o più prenotazioni.	Prenotazione

2.2 Riscrittura e Strutturazione dei Requisiti

Entità	Frasi Inerenti
Aeroporto	Ogni aeroporto sia identificato univocamente da un codice e sia caratterizzato da un nome, da una città e da una nazione. Si assuma che in una data città non vi possano essere due aeroporti con lo stesso nome. Per ogni aeroporto, sia definito l'insieme dei tipi di aeroplano che possono decollare/atterrare.
Tipo di Aeroplano	Ogni tipo di aeroplano sia identificato dal nome e sia caratterizzato dal nome dell'azienda costruttrice, dal numero max di posti a sedere e dall'autonomia di volo.
Aeroplano	Ogni aeroplano sia identificato da un codice e sia caratterizzato dal tipo (di aeroplano) e dal numero di posti a sedere messi effettivamente a disposizione per i passeggeri. Si assuma che uno stesso aeroplano possa essere utilizzato da compagnie diverse.
Volo	Ogni volo sia caratterizzato da un codice, che lo identifica univocamente, dalla compagnia aerea che offre il volo, dai giorni della settimana in cui viene offerto (si assuma che ogni volo sia effettuato al pi`u una volta in un dato giorno, che gli orari di ogni volo siano sempre gli stessi e che ogni volo, anche quando costituito da più tratte, inizi e termini nella stessa giornata) e, per ogni classe (business, economica, ..), dal prezzo del biglietto (si assuma che biglietti di un dato volo relativi alla stessa classe abbiano tutti lo stesso prezzo).
Tratta	Ogni volo sia costituito da un insieme di tratte intermedie (una tratta sia una porzione di volo priva di scali intermedi), ciascuna identificata da un numero progressivo (prima tratta del volo, seconda tratta del volo, ..). Ogni tratta sia caratterizzata da un aeroporto di partenza, un aeroporto di arrivo, un orario previsto di partenza e un orario previsto di arrivo.
Istanza di Tratta	Per ogni specifica istanza di tratta si specifichi la data in cui avrà luogo (ad esempio, la tratta Trieste-Monaco del 25 febbraio 2012), l'aeroplano utilizzato (si assuma che su una stessa tratta possano essere utilizzati in date diverse aeroplani diversi) e il numero di posti ancora disponibili.
Prenotazione	Ogni prenotazione relativa ad una certa istanza di tratta sia identificata dal posto prenotato (numero più lettera; ad esempio, posto 16D) e sia caratterizzata dal nome, dal cognome e dal recapito telefonico della persona che ha prenotato il posto.

2.3 Definizione dei Requisiti

Legenda: entità **(nuova)**, attributo di entità, relazione, attributo di relazione
Si voglia modellare il seguente insieme di informazioni riguardanti un sistema di prenotazione dei voli aerei.

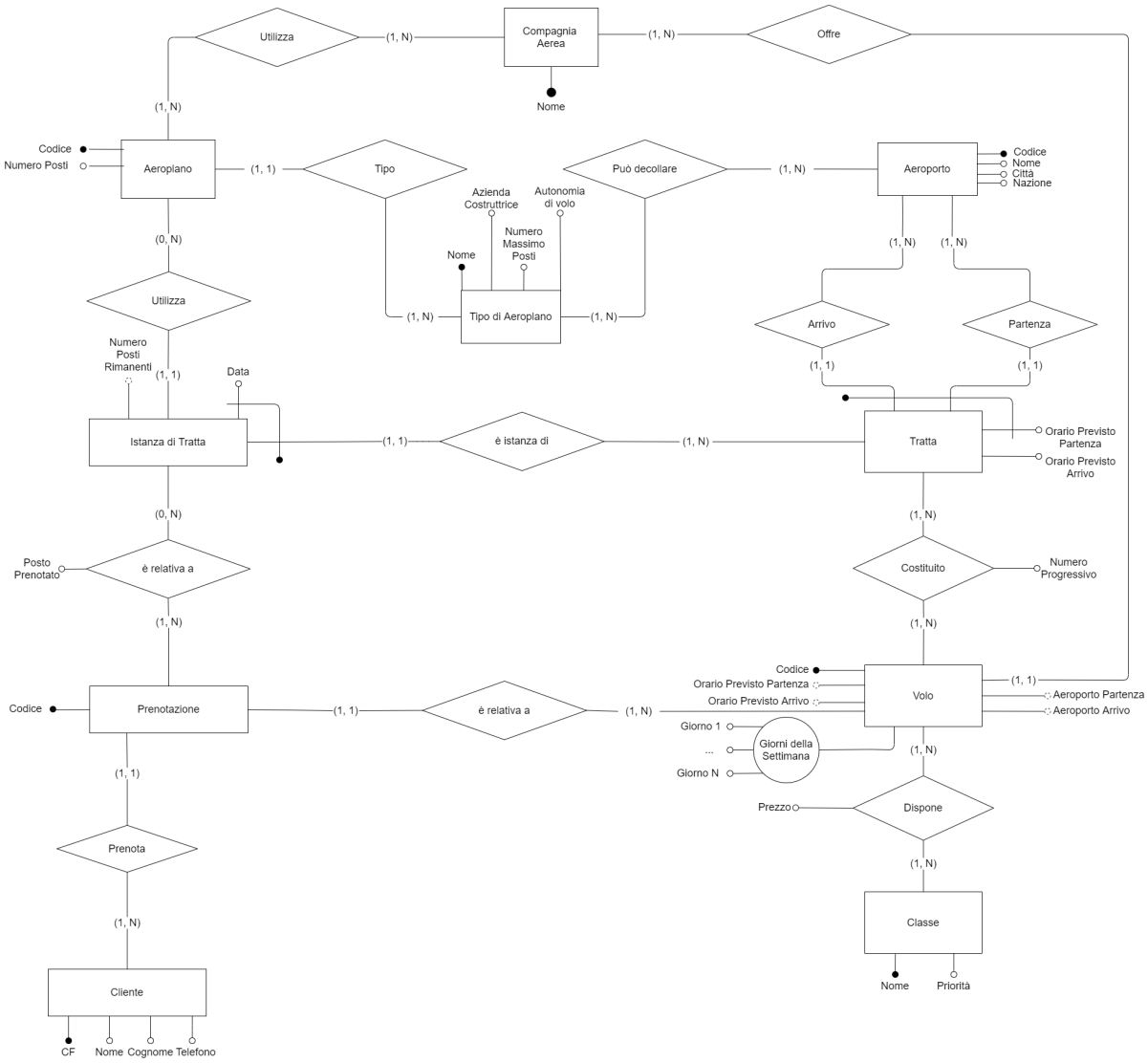
- Ogni **aeroporto** sia identificato univocamente da un codice e sia caratterizzato da un nome, da una città e da una nazione. Si assuma che in una data città non vi possano essere due aeroporti con lo stesso nome
- Ogni **tipo di aeroplano** sia identificato dal nome e sia caratterizzato dal nome dell'azienda costruttrice, dal numero massimo di posti a sedere e dall'autonomia di volo.
- Ogni **aeroplano** sia identificato da un codice e sia caratterizzato dal tipo (di aeroplano) e dal numero di posti a sedere messi effettivamente a disposizione per i passeggeri. Si assuma che uno stesso aeroplano possa essere utilizzato da **compagnie** diverse.
- Per ogni aeroporto, sia definito l'insieme dei tipi di aeroplano che possono decollare/atterrare.
- Ogni **volo** sia caratterizzato da un codice, che lo identifica univocamente, dalla compagnia aerea che offre il volo, dai giorni della settimana (attributo multivalore) in cui viene offerto (si assuma che ogni volo sia effettuato al più una volta in un dato giorno, che gli orari di ogni volo siano sempre gli stessi e che ogni volo, anche quando costituito da più tratte, inizi e termini nella stessa giornata (una sola data inizio e fine) e, per ogni **classe (di volo)** (business, economica, ..), dal prezzo del biglietto (si assuma che biglietti di un dato volo relativi alla stessa classe abbiano tutti lo stesso prezzo).
- Ogni volo sia costituito da un insieme di tratte intermedie (una **tratta** sia una porzione di volo priva di scali intermedi), ciascuna identificata da un numero progressivo (prima tratta del volo, seconda tratta del volo, ..). Ogni tratta sia caratterizzata da un aeroporto di partenza, un aeroporto di arrivo, un orario previsto di partenza e un orario previsto di arrivo.
- Per ogni specifica **istanza di tratta** memorizzare la data in cui avrà luogo (ad esempio, la tratta Trieste-Monaco del 25 febbraio 2012), l'aeroplano utilizzato (si assuma che su una stessa tratta possano essere utilizzati in date diverse aeroplani diversi) e il numero di posti ancora disponibili.
- Ogni **prenotazione** relativa ad una certa istanza di tratta sia identificata dal posto prenotato (numero più lettera; ad esempio, posto 16D) e sia caratterizzata dal nome, dal cognome e dal recapito telefonico della **persona** (entità cliente) che ha prenotato il posto.

2.4 Operazioni Frequenti

1. Aggiungere un volo a quelli offerti da una certa compagnia aerea (5 volte all'anno)
2. Aggiungere una classe di volo ad un volo (5 volte l'anno)
3. Aggiungere un nuovo aeroplano (10 volte all'anno)
4. Modificare gli orari di una tratta (1 volta a settimana)
5. Ottenere tutti aeroplani atterrati e decollati in un giorno in un aeroporto (1 volta al giorno)
6. Aggiungere un'istanza di tratta (200 volte al giorno)
7. Fare una prenotazione (5000 volte al giorno)
8. Conoscere i posti rimanenti di un volo di un giorno (10000 volte al giorno)
9. Conoscere date, ore e aeroporti di p/a di un volo (15000 volte al giorno)
10. Ottenere tutti i voli diretti verso una città in un intervallo di tempo (20000 volte al giorno)

3 Progettazione Concettuale

3.1 Modello Entità-Relazione



3.2 Considerazioni

- Tra l'entità **TRATTA** e l'entità **ISTANZA DI TRATTA** si è riconosciuto il pattern "istanza di", nel quale **ISTANZA DI TRATTA** rappresenta concretamente l'entità astratta **TRATTA**.
- Anche se nel testo viene usato il termine "caratterizzato" ("*Ogni aeroplano sia identificato da un codice e sia caratterizzato dal tipo (di aeroplano)...*"), si è deciso di non usare una caratterizzazione poichè effettivamente il domino delle tipologie di aeroplani non era definito. Pertanto la soluzione ideale è creare una entità **TIPO DI AEROPLANO** e metterla in relazione con **AEROPLANO**.
- Si è deciso di riportare "Aeroporto di Partenza", "Aeroporto di Arrivo", "Orario Previsto di Partenza" ed "Orario Previsto di Arrivo" come attributi derivati nell'entità **VOLO** per una possibile comodità nel reperimenti delle informazioni di un volo.
- Con lo schema attuale non è possibile prenotare un volo scegliendo anche la classe. Pertanto, nella fase di progettazione logica, si dovrà provvedere a una reificazione della relazione "Dispone" in una entità.

3.3 Vincoli Aziendali

1. La città di partenza e arrivo di un volo devono essere diverse
2. Un aeroplano può essere utilizzato nella istanza di tratta se l'aeroplano può atterrare negli aeroporti di partenza e di arrivo della tratta.
3. Una compagnia aerea può offrire un volo solo se dispone di tutti gli aerei che sono utilizzati dalle tratte del volo.
4. Un volo è offerto da una compagnia, la quale prevede tutte le tratte del volo.
5. Per ogni tratta è presente anche il relativo volo diretto con solo quella tratta.
6. Ogni prenotazione riguarda un unico volo.
7. L'insieme delle tratte derivate dalle istanze di tratta di una prenotazione è uguale all'insieme delle tratte che costituiscono il volo prenotato.
8. se un volo è composto da più tratte allora l'orario di partenza della tratta i-esima deve essere maggiore rispetto all'orario di arrivo della tratta (i-1)-esima e devono essere sullo stesso aeroporto.
9. il numero di posti di un aeroplano deve essere minore o uguale del numero massimo di posti del tipo di aeroplano a cui aeroplano appartiene

4 Progettazione Logica

4.1 Analisi delle Ridondanze

In questa sezione analizziamo se conviene mantenere gli attributi ridondanti oppure se è meglio rimuoverli. Tale analisi si basa sul numero medio di accessi in lettura e in scrittura. Nel calcolo bisogna considerare che, come da convenzione, il costo di una lettura è 1 mentre di una scrittura è 2.

4.1.1 Attributi ridondanti

- Numero di posti rimanenti di un volo
- Aeroporto di partenza e arrivo, orario previsto di partenza e arrivo

4.1.2 Ipotesi iniziali

Prima di eseguire il calcolo, facciamo delle ipotesi iniziali:

- i calcoli sono effettuati basando il popolamento del database
- 20 anni di uso del sistema e una crescita costante dei volumi
    - da cui: la media è 20 anni / 2 = 10 anni di popolamento

Si stimano **in media**:

- 5000 prenotazioni al giorno
- 200 posti per ogni aereo
    - da cui: ipotizzando di vendere il 100% dei biglietti: 25 voli al giorno
- ricordando che: ogni prenotazione riguarda un volo (può essere composto da più tratte)
- ogni volo sia composto da 1 o 2 tratte (ovvero in media o 0 o 1 scalo)
    - da cui: numero di tratte diverse che compongono un volo = 1,5
- ricordando che: per ogni tratta ‘singola’ è presente anche il relativo volo diretto ed esistono altrettanti voli con 1 scalo (dall’ipotesi sopra che afferma che il numero di tratte per volo è 1,5)
    - da cui: numeri di voli diversi che usano una stessa tratta = 2
- ogni volo sia ripeta in media 2 volte a settimana
- aggiungere 10 voli/anno
    - da cui: 10 voli annui * (20 anni / 2) = 100 voli in più dopo 10 anni
- quando si cerca il numero di posti rimanenti la metà dei posti è stato già prenotato
    - da cui: 200 posti aereo / 2 = 100 posti aereo

NB: totale accessi di ogni singola sotto-operazione:
(frequenza operazione) * (costo sotto-operazione) * (numero di accessi)

4.1.3 Numero di posti rimanenti in un volo

Calcolo dell'attributo derivato, dato un volo specifico (un volo e una data):

1. identificare tutte le tratte che lo compongono (tramite relazione 'Costituito')
2. identificare tutte le istanze di tratte cercate: tramite le tratte trovate e la data
3. per ogni istanza di tratta trovata:
    a. identificare l'aeroplano utilizzato (tramite relazione Utilizza)
    b. leggere il numero di posti disponibili totali (attributo di Aeroplano)
    c. contare il numero di prenotazioni dell'istanza (relazione 'è relativa a')
    d. sottrarre il valore ricavato al punto b) con il valore ricavato al punto c)
4. ritornare il minimo tra i posti disponibili di ogni istanza di tratta

4.1.3.1 Tabella delle operazioni

indice operazione	operazione	tipo operazione	frequenza giornaliera
1	Fare una prenotazione	scrittura	5.000
2	Conoscere i posti rimanenti	lettura	10.000

4.1.3.2 Tabella dei volumi

concetto	tipo	numero di tuple
Aeroplano	E	1.000
Aeroplano 'Utilizza' Istanza di Tratta	R	136.875
Istanza di Tratta	E	136.875
Istanza di Tratta 'è relativa a' Prenotazione	R	27.375.000
Prenotazione	E	18.250.000

4.1.3.3 Tabella degli accessi (in presenza di attributo ridondante)

operaz.	entità o relazione	tipo operaz.	costo operaz.	dettaglio sotto-operazione	numero accessi	totale accessi
1	Prenotazione	scrittura	2	inserire la nuova tupla	1	10.000
1	è relativa a	scrittura	2	inserire la nuova tupla in ogni istanza di tratta prenotata	1,5	15.000
1	Istanza di Tratta	lettura	1	leggere il dato precedente (verificare sia maggiore di 0)	1,5	7.500
1	Istanza di Tratta	scrittura	2	aggiornare l'attributo ridondante togliendo un posto	1,5	15.000
2	Costituito	lettura	1	cercare tutte le tratte che compongono il volo	1,5	15.000
2	Tratta	lettura	1	lettura della tupla richiesta	1,5	15.000
2	è istanza di	lettura	1	lettura della tupla richiesta	1,5	15.000
2	Istanza di Tratta	lettura	1	leggere l'attributo ridondante	1,5	15.000

Costo totale in presenza di attributo ridondante: **107.500 accessi**

4.1.3.4 Tabella degli accessi (in assenza di attributo ridondante)

operaz.	entità o relazione	tipo operaz.	costo operaz.	dettaglio sotto-operazione	numero accessi	totale accessi
1	Prenotazione	scrittura	2	Inserisco la nuova tupla	1	10.000
1	è relativa a	scrittura	2	inserisco la nuova tupla in ogni istanza di tratta prenotata	1,5	15.000
2	Costituito	lettura	1	cercare tutte le tratte che compongono il volo	1,5	15.000
2	Tratta	lettura	1	lettura della tupla richiesta	1,5	15.000
2	è istanza di	lettura	1	lettura della tupla richiesta	1,5	15.000
2	Istanza di Tratta	lettura	1	Lettura della tupla richiesta	1,5	15.000
2	Utilizza	lettura	1	Lettura della tupla richiesta	1,5	15.000
2	Aeroplano	lettura	1	Lettura attributo "Numero Posti"	1,5	15.000
2	è relativa a	lettura	1	Contare il numero di prenotazioni associate a quella determinata istanza di tratta	150	1.500.000

Costo totale in assenza di attributo ridondante: **1.615.000 accessi**

Dato che 107.500 è di gran lunga minore di 1.615.000:
il primo attributo ridondante è sicuramente conveniente mantenerlo.

4.1.4 Aeroporti di partenza e arrivo, orari previsti di partenza e arrivo

In questo caso abbiamo quattro attributi ridondanti, ma li consideriamo in una sola analisi, in quanto riguardano le stesse operazioni previste e sono tutti contenuti nella stessa entità.

Calcolo degli attributi ridondanti, dato un volo:

1. Aeroporto di partenza e orario previsto di partenza:
    a. identificare la prima tra le tratte che lo compongono (tramite relazione ‘Costituito’), ovvero quella con attributo ‘numero_progressivo’ = 1
    b. leggere l’orario previsto di partenza nella tratta identificata
    c. leggere l’aeroporto di partenza (tramite la relazione ‘Partenza’)
2. Aeroporto di arrivo e orario previsto di arrivo:
    a. speculare a sopra, ma bisogna identificare l’ultima tratta, ovvero quella con attributo ‘numero_progressivo’ = max

4.1.4.1 Tabella delle operazioni

indice operazione	operazione	tipo operazione	frequenza settimanale
1	Modificare orari di una tratta	scrittura	1
2	Conoscere date, orari e aeroporti di un volo	lettura	105.000

4.1.4.2 Tabella degli accessi (in presenza di attributo ridondante)

operaz.	entità o relazione	tipo operaz.	costo operaz.	dettaglio sotto-operazione	numero accessi	totale accessi
1	Tratta	scrittura	2	Inserisco la nuova tupla (orari aggiornati)	1	2
1	Costituito	lettura	1	trovo tutti i voli che usano quella tratta come prima o ultima	2	2
1	Volo	scrittura	2	inserisco la nuova tupla nei voli che usano tratta come prima o ultima	2	4
1	Arrivo	scrittura	2	modifico aeroporto arrivo	1	2
1	Partenza	scrittura	2	modifico aeroporto partenza	1	2
2	volo	lettura	1	accedo alla tupla	1	105.000

Costo totale in presenza di attributo ridondante: **105.012 accessi**

4.1.4.3 Tabella degli accessi (in assenza di attributo ridondante)

operaz.	entità o relazione	tipo operaz.	costo operaz.	dettaglio sotto-operazione	numero accessi	totale accessi
1	Tratta	scrittura	2	Inserisco la nuova tupla	1	2
1	Arrivo	scrittura	2	modifico aeroporto arrivo	1	2
1	Partenza	scrittura	2	modifico aeroporto partenza	1	2
2	Volo	lettura	1	accedo alla tupla	1	105.000
2	Costituito	lettura	1	trovo la prima e ultima tratta del volo	1,5	157.500
2	Tratta	lettura	1	leggo orari della tratta	1,5	157.500
2	Arrivo	lettura	1	leggo aeroporto arrivo	1,5	157.500
2	Partenza	lettura	1	leggo aeroporto partenza	1,5	157.500

Costo totale in assenza di attributo ridondante: **735.006 accessi**

Dato che 105.012 è di gran lunga minore di 735.006:
il secondo attributo ridondante è sicuramente conveniente mantenerlo

4.2 Ristrutturazione del Modello Entità-Relazione

Dall'analisi delle ridondanze ne segue che gli attributi derivati sono vantaggiosi e quindi verranno mantenuti.
Si è deciso per motivi di efficienza e di comodità di usare come primary key di Tratta non la tupla (Aereporto_Arrivo, Aeroporto_Partenza, Orario_Previsto_Partenza) ma di aggiungere un campo ID numerico.
Motivi della reificazione della relazione DISPONE nell'entità CLASSE DI VOLO
Inizialmente:

- DISPONE : relazione (M-M) tra VOLO e CLASSE,
    - dotata di un attributo prezzo
- "è relativa a": relazione (1-M) tra PRENOTAZIONE e VOLO.

Problematiche di tale soluzione:

- La prenotazione di un volo (ovvero la relazione) riguarda il volo generico e non una specifica classe del volo. Ne segue che non è possibile prenotare una specifica classe e quindi nemmeno conoscere il prezzo dato che dipende dalla classe di volo.
- Si noti che una relazione ternaria CLASSE DI VOLO composta da PRENOTAZIONE, VOLO e CLASSE non sarebbe stata una buona soluzione in quanto avrebbe generato una dipendenza funzionale della prenotazione verso il prezzo di una classe ed anche verso l'insieme delle classi disponibili per un certo volo:
    - il prezzo sarebbe stato un dato ridondante per ogni prenotazione
    - in assenza di prenotazione per una certa classe:
        - non si può sapere se quella classe non è disponibile per quello o se non sono ancora state fatte prenotazioni
        - non si può nemmeno sapere il prezzo di quella classe di volo

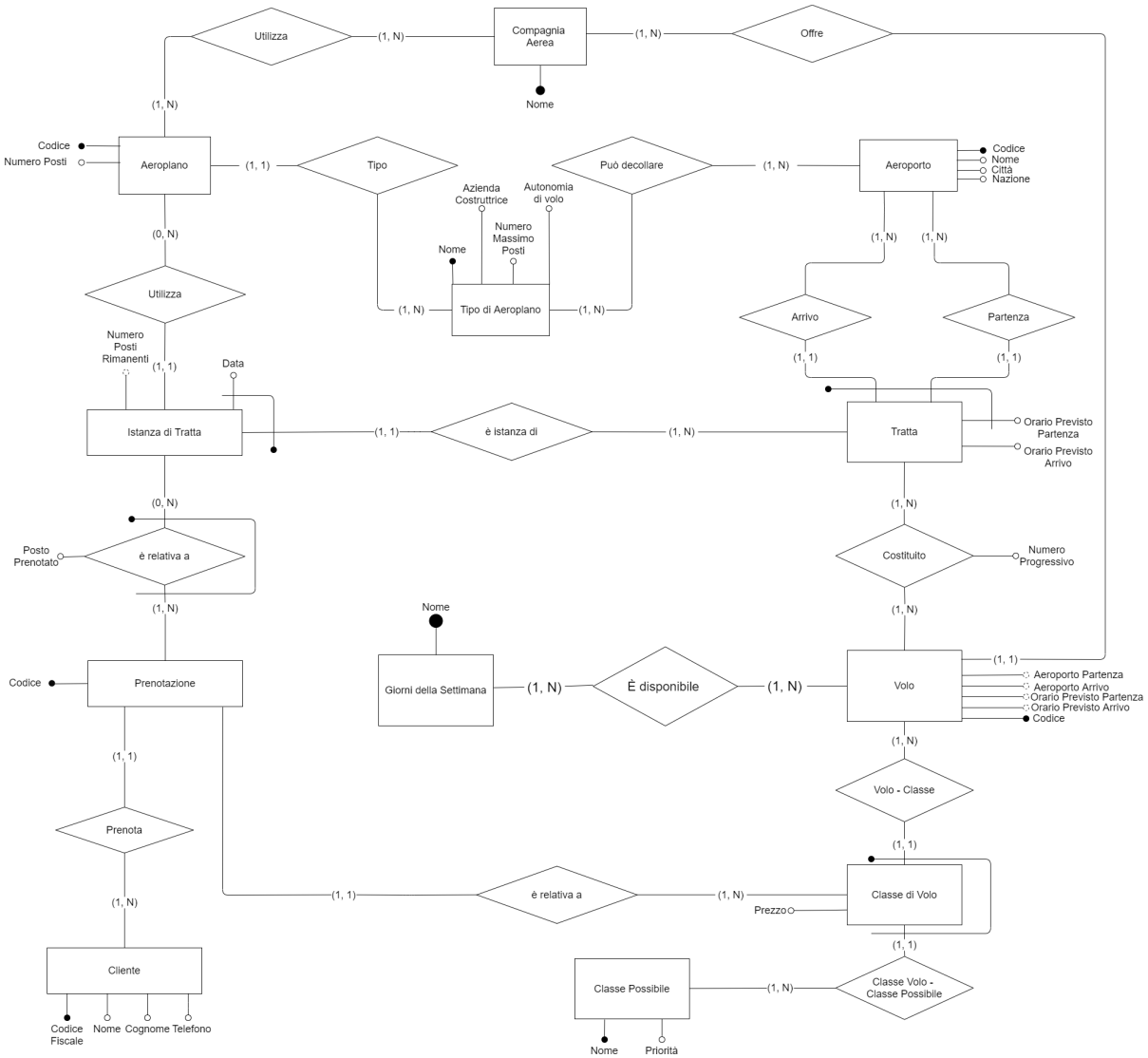
Soluzione:

- REIFICAZIONE della relazione DISPONE nell'entità CLASSE DI VOLO.

NB: Entità CLASSE è stata rinominata CLASSE POSSIBILE per aiutare la comprensione dello schema
Vantaggi:

- possibilità di scegliere la classe di volo e conoscerne il prezzo
- non generazione di dipendenze funzionali errate
    - (causate da un'eventuale relazione ternaria)

4.3 Modello Entità-Relazione Ristrutturato



4.4 Traduzione in Modello Relazionale

1. CLIENTE(*codice_fiscale, nome, cognome, telefono*)
    - PK: codice_fiscale
    - NOT NULL: nome, cognome
2. PRENOTAZIONE(*codice, cliente, volo, classe*)
    - PK: codice
    - FK: cliente → CLIENTE.codice_fiscale
    - FK: volo → CLASSE_DI_VOLO.volo
    - FK: classe → CLASSE_DI_VOLO.classe
    - NOT NULL: cliente, volo, classe
3. PRENOTAZIONE_ISTANZA_DI_TRATTA(*codice_prenotazione,tratta, data_istanza_tratta, posto_prenotato*)
    - PK: codice_prenotazione, tratta, data_istanza_tratta
    - FK: codice_prenotazione → PRENOTAZIONE.codice
    - FK: data_istanza_tratta → ISTANZA_DI_TRATTA.data
    - FK: tratta → ISTANZA_DI_TRATTA.tratta
    - NOT NULL: posto_prenotato
4. ISTANZA_DI_TRATTA(*tratta, data, aeroplano, numero_posti_rimanenti*)
    - PK: tratta, data
    - FK: tratta → TRATTA.id
    - FK: aeroplano → AEROPLANO.codice
    - NOT NULL: numero_posti_rimanenti DEFAULT (AEROPLANO.numero_posti where AEROPLANO.codice = aeroplano)
    - NOT NULL: aeroplano
5. CLASSE_POSSIBILE(*nome, priorità*)
    - PK: priorità
    - NOT NULL, UNIQUE: nome
6. CLASSE_DI_VOLO(*classe, volo, prezzo*)
    - PK: classe, volo
    - FK: classe → CLASSE_POSSIBILE.priorità
    - FK: volo → VOLO.codice
    - NOT NULL: prezzo
7. VOLO(*codice, orario_previsto_arrivo, orario_previsto_partenza, aeroporto_arrivo, aeroporto_partenza, compagnia_aerea*)
    - PK: codice
    - FK: compagnia_aerea → COMPAGNIA_AEREA.nome
    - NOT NULL: orario_previsto_arrivo, orario_previsto_partenza, aeroporto_arrivo, aeroporto_partenza, compagnia_aerea
8. GIORNI_DELLA_SETTIMANA_VOLO(giorno, volo)
    - PK: giorno, volo
    - FK: giorno → GIORNI_DELLA_SETTIMANA.nome
    - FK: volo → VOLO.codice
9. VOLO_TRATTA(tratta, volo, numero_progressivo)

    - PK: tratta, volo
    - FK: tratta → TRATTA.id
    - FK: volo → VOLO.codice
    - NOT NULL: numero_progressivo
10. GIORNI_DELLA_SETTIMANA(*nome*)
    - PK: nome
11. COMPAGNIA_AEREA(*nome*)
    - PK: nome
12. COMPAGNIA_AEREA_AEROPLANO(*compagnia_aerea, aeroplano*)
    - PK: compagnia_aerea, aeroplano
    - FK: compagnia_aerea → COMPAGNIA_AEREA.nome
    - FK: aereoplano → AEROPLANO.codice
13. AEROPLANO(*codice, numero_posti, tipo_aeroplano*)
    - PK: codice
    - NOT NULL: numero_posti (DEFAULT 200), tipo_aeroplano
    - FK: tipo_aereoplano → TIPO_DI_AEROPLANO.nome
14. TIPO_DI_AEROPLANO(*nome, azienda_costruttrice, numero_massimo_posti, autonomia_di_volo*)
    - PK: nome
    - NOT NULL: azienda_costruttrice, numero_massimo_posti, autonomia_di_volo
15. PUÒ_DECOLLARE(tipo_*aeroplano, aeroporto*)
    - PK: tipo_*aeroplano*, aeroporto
    - FK: tipo_*aeroplano* → TIPO_DI_AEROPLANO.nome
    - FK: *aeroporto* → AEROPORTO.codice
16. AEROPORTO(*codice, nome, città, nazione*)
    - PK: codice
    - NOT NULL: nome, città, nazione
17. TRATTA(*id, orario_previsto_arrivo, orario_previsto_partenza, aeroporto_arrivo, aeroporto_partenza*)
    - PK: id
    - FK: aeroporto_arrivo → AEROPORTO.codice
    - FK: aeroporto_partenza → AEROPORTO.codice
    - NOT NULL: orario_previsto_arrivo, orario_previsto_partenza, aeroporto_arrivo, aeroporto_partenza

5 Progettazione Fisica

5.1 Indici proposti

Per garantire una migliore efficienza nelle operazioni più frequenti, andiamo a creare degli indici appositi. Essi garantiscono dei tempi di accesso molto più veloci in quanto eseguono un ordinamento dei valori. Tale vantaggi sono compensati da un maggiore costo in fase di inserimento o modifica, in quanto bisogna modificare anche l'indice per garantirne la correttezza (solitamente un B+Tree).
La documentazione di PostgreSQL afferma che "PostgreSQL genera in automatico un indice unique (ovvero che non ammette valori non unique) per tutti i campi con vincolo di chiave primaria o vincolo unique. Di conseguenza non c'è alcun bisogno di creare tali indici perchè semplicemente duplicherebbero quelli già esistenti."
Di conseguenza ci concentreremo sugli indici secondari che non richiedono vincoli di unicità nei valori.
Osservando le operazioni più frequenti (8-9-10):

- 8 → è sufficiente una ricerca sulla PK di IstanzaDiTratta
    - **non sono necessari indici espliciti**
- 9 → è sufficiente una ricerca sulla PK di Volo e GiorniDellaSettimana_Volo
    - non sono necessari indici espliciti MA
    - è importante **dichiarare la PK** composta **<volo, giorno>** e non <giorno, volo> in modo da rendere efficiente la ricerca sul volo nell'indice generato in automatico
- 10 → la **citta in Aeroporto** non è un campo UNIQUE, e, considerata la frequenza delle operazioni di ricerca un indice, è un'ottima soluzione per ridurre drasticamente i tempi di ricerca. Inoltre la città in cui si trova un aeroporto non è un valore che verrà mai aggiornato, l'unica possibilità è che vengano aggiunti al database nuovi aeroporti, operazione decisamente infrequente (potrebbe anche non accadere mai).

Codice PostgreSQL per la creazione dell'indice:
CREATE INDEX citta_idx ON Aeroporto (citta);

5.2 Implementazione

Il codice per l'implementazione del database in Postgresql è allegato alla relazione.
In particolare, si hanno i seguenti file:

- *0_deleteDB.sql*: eliminare il database in caso di errori,
- *1_createDB.sql:* creare il database e le tabelle con rispettivi campi e domini
- *2_constraints.sql:* per aggiungere vincoli di chiave esterna e definizione indici
- *3_triggers.sql:* per creazione dei trigger (per i vincoli o gli attributi ridondanti)
- *4_functions.sql:* per la creazione di alcune view, utili per la parte di analisi
- *5_populateDB_obsolete_for_test_only.sql*: per il testing del database iniziale
- *5_generatePseudoRandomData.R*: per generazione dei dati pseudo casuali
- *6_populateDatabase.R*: per il popolamento del db (caricamento dei dati in pg)

5.2.1. Popolamento del database

Il popolamento del database è stato realizzato prima eseguendo prima i file sql direttamente dalla console di postgres: *1_createDB.sql*, *2_constraints.sql, 3_triggers.sql, 4_functions.sql.*
Infine si è lanciato da R *5_generatePseudoRandomData.R* per la generazione pseudo casuali dei dati, *6_populateDatabase.R* per il popolamento vero e proprio del database.

Purtroppo, dato che la generazione di tutti di dati pseudo casuale è risultata molto complicata nel nostro dominio applicativo sia per la quantità delle tabelle presenti sia per la mole di dati da generare ma soprattutto per cercare di rispettare tutti i vincoli imposti, si è dovuti scendere a dei compromessi, violando qualche condizione.

Nello specifico gli aeroplani che possono essere utilizzati da un'istanza di tratta, teoricamente sono solo quelli che possono decollare ed atterrare negli aeroporti di quella tratta, ma nella pratica l'estrazione non è avvenuta solo da tale sottoinsieme ma da tutti i possibili aeroplani. Ne consegue che tale vincolo non è detto non sia stato violato (ed effettivamente è stato così in qualche caso). Dato che il trigger era però già stato implementato si è dovuti rimuoverlo prima del popolamento che altrimenti lanciava eccezione.

Un altro esempio, ma non è detto sia un problema, è che è consentito l' *"overbooking"* ovvero le prenotazioni di istanza di tratta possono essere di più rispetto al numero di posti dell'aereo disponibili.

Un'ultima semplificazione apportata: dall'analisi delle ridondanze si hanno due tabelle piuttosto grandi: Prenotazione_IstanzaDiTratta con 27 milioni di elementi e IstanzaDiTratta con 18 milioni. Ma gestire una mole così grande di dati per una macchina di uso comune con 8-16 GB di RAM è praticamente impossibile. Il problema non è tanto nelle dimensioni del file o nelle prestazioni di postgres, quanto nella generazione pseudo casuale da R.
Quindi siamo stati costretti ad eseguire un forte ridimensionamento utilizzando solo un milione di records per Prenotazione.

6 Generazione dei dati

Si è deciso di generare dati tramite un generatore pseudo casuale poiché non c'erano dataset che modellassero completamente il database idealizzato. In ogni caso, la creazione dei dati si è basata su alcuni dataset (in formato csv) reperiti dal web, in particolare da un sito open source denominato OpenFlights (https://openflights.org/data.html).
I file in questione sono :

- Airport
- Airline
- Routes

I file originali si possono trovare all'interno della directory *"./old_csv/"*.

6.1 Pulizia dei dati

I file descritti nell'introduzione del capitolo contenevano informazioni non necessarie alla creazione del database e in aggiunta, alcuni valori che rappresentavano chiavi esterne non erano presenti nella tabella dove erano chiavi primari. Si è deciso quindi di eseguire un minimo di pulizia e di consistenza tra le tabelle.

```
aeroporto <- read.csv("./old_csv/Aeroporti.csv", header = FALSE)
%>%
  select(2:5) %>%
  rename(nome = V2, citta = V3, nazione = V4, codice = V5) %>%
  filter(codice != "\\N") %>%
  unique()

compagniaAerea <- read.csv("./old_csv/CompagniaAerea.csv",
                           header = FALSE) %>%
  select(2) %>%
  rename(nome = V2) %>%
  unique()

tratta <- read.csv("./old_csv/Tratta.csv", header = FALSE) %>%
  select(3, 5) %>%
  rename(aeroporto_partenza = V3, aeroporto_arrivo = V5) %>%
  subset(aeroporto_partenza %in% aeroporto$codice) %>%
  subset(aeroporto_arrivo %in% aeroporto$codice) %>%
  unique() %>%
  slice_sample(n = 50)
```

6.2 Creazione dei dati pseudo casuali

Successivamente, si sono predisposti diversi dataframe in modo che avessero una struttura simile alle "future" tabelle presenti nel database. Si riportano le principali generazione di dati sfruttando una libreria esterna denominata **charlatan** (https://docs.ropensci.org/charlatan/) e funzioni create ad-hoc:

- **Aeroplano**

```
aeroplano <- data.frame(matrix(ncol = 0, nrow = 1000))
aeroplano$codice <- generateUniqueId(n = nrow(aeroplano), prefix =
"AP")
aeroplano$tipo_aeroplano <-
  sample(tipoDiAeroplano$nome, nrow(aeroplano), replace = TRUE)
aeroplano$numero_posti <-
  map(aeroplano$tipo_aeroplano, function(v) {
    numberRow <- which(tipoDiAeroplano$nome == v)
    maxValue <- tipoDiAeroplano[numberRow, 2]
    t <- ch_integer(n = 1, min = 20, max = maxValue)
    return(t)
  }) %>% unlist()
```

Per generare la colonna *"numero_posti"* si è usato il costrutto map (integrato in dplyr) per determinare in quali righe il nome dell'aeroplano fosse uguale a quello del tipo di aeroplano, e successivamente estrapolare il numero massimo per poi usarlo come estremo maggiore nell'intervallo di generazione dei posti.

- **Cliente**

```
cliente <- ch_generate('name', 'phone_number', n = 30000, locale =
"it_IT")
colnames(cliente) <- c("nome","telefono")
cliente$nome <- gsub(pattern = "Sig. |Sig.ra |Dott. ", replacement
= "", x = cliente$nome)
cliente$codice_fiscale <- generateId(n=nrow(cliente), format =
"??????##?##???#?")
cliente <- cliente %>% separate("nome", c("nome","cognome"), extra
= "merge")
cliente <- cliente[,
c("codice_fiscale","nome","cognome","telefono")]
```

In questo caso, sfruttando la funzione "*ch_generate"* contenuta nel package charlatan, si è potuto generare nomi e numeri di telefoni per avere dai dati semi realistici. La funzione *"generateId"* è una funzione custom che in base a un formato generare stringhe alfanumeriche (in questo caso per rappresentare un codice fiscale).

- **Volo_Tratta**

```
voloTratta1 <- data.frame(matrix(ncol = 0, nrow = 50))
voloTratta1$volo <- sample(volo$codice, size = nrow(voloTratta1))
voloTratta1$tratta <- sample(tratta$id, size = nrow(voloTratta1))
voloTratta1$numero_progressivo <- rep(1)

voloTratta2 <- anti_join(volo, voloTratta1, by = c("codice" =
"volo")) %>% subset(select = -c(compagnia_aerea))
voloTratta2$tratta <- sample(tratta$id, size = nrow(voloTratta2))
voloTratta2$numero_progressivo <- rep(1)

## Trovo delle tratte per le quali l'aeroporto di arrivo è anche
aereoporto di partenza
secondaTratta <- inner_join(voloTratta2, tratta, by = c("tratta" =
"id")) %>%
  inner_join(tratta, by = c("aeroporto_arrivo" =
"aeroporto_partenza")) %>%
  select(codice, id, numero_progressivo) %>%
  unique()

colnames(voloTratta2) <- c("volo", "tratta", "numero_progressivo")
colnames(secondaTratta) <- c("volo", "tratta",
"numero_progressivo")

secondaTratta$numero_progressivo <-
secondaTratta$numero_progressivo + 1

voloTratta <- rbind(voloTratta1, voloTratta2, secondaTratta)

remove(voloTratta1, voloTratta2, secondaTratta)
```

In questa parte di codice, si è deciso di aggregare a partizioni i voli con le rispettive tratte. In particolare, una parte dei voli doveva prevedere una sola tratta mentre una seconda parte doveva prevedere una seconda tratta. Nel dataframe temporaneo secondaTratta avviene la seconda aggregazione, nella quale si esegue un join tra *secondaTratta* e *voloTratta2* (un dataframe con i voli che non avevano ricevuto ancora una tratta), usando come chiavi comuni *"aeroporto_arrivo"* di *secondaTratta* e *"aeroporto_partenza"* di *voloTratta*. In questo modo si ottengono tratte per le quali sappiamo che l'aeroporto di partenza è uguale all'aeroporto di arrivo di almeno una tratta esistente.

- **Prenotazione_IstanzaDiTratta**

```
prenotazione_IstanzaDiTratta <-
inner_join(prenotazione[c("codice", "volo")], voloTratta, by =
"volo")

primeTratte <- prenotazione_IstanzaDiTratta %>%
  filter(numero_progressivo == 1)

trattePossibili <- tratta$id %>% as.list()

# Lista di vettori che ha come nomi i valori delle tratte
# Permette un accesso più rapido alle possibili date ti una tratta
specifica
dateTrattePossibili <- trattePossibili %>%
  setNames(as.character(trattePossibili)) %>%
  lapply(., function(x) {
    istanzaDiTratta %>%
      filter(tratta == x) %>%
      pull(data_istanza)
  })

primeTratte$data_istanza <- lapply(primeTratte$tratta, function(x)
{
  sample(dateTrattePossibili[[x]], size = 1)
}) %>% unlist()


secondeTratte <- prenotazione_IstanzaDiTratta %>%
filter(numero_progressivo >= 2) %>%
  inner_join(select(primeTratte, data_istanza, codice), by =
"codice")

prenotazione_IstanzaDiTratta <- rbind(primeTratte, secondeTratte)
%>%
  select(codice, tratta, data_istanza) %>%
  mutate(data_istanza =  as_date(data_istanza))
```

```
prenotazione_IstanzaDiTratta$posto_prenotato <-
  inner_join(
    prenotazione_IstanzaDiTratta,
    istanzaDiTratta,
    by = c("tratta" = "tratta", "data_istanza" = "data_istanza")
  ) %>%
  select(aeroplano) %>%
  inner_join(aeroplano, by = c("aeroplano" = "codice")) %>%
  select(numero_posti) %>%
  unlist() %>%
  lapply(., function(v) {
    v <- sample((1:v), size = 1)
  })
```

Per creare il dataframe *prenotazione_IstanzaDiTratta* si è voluti rispettare il vincolo che impone che se una Prenotazione P riguarda un Volo V allora in Prenotazione_IstanzaDiTratta la prenotazione P sarà in relazione con delle istanze di tratte IdT1, … , IdT che sono istanze delle tratte T1, … , Tn che sono le tratte che compongono il volo V.
Inoltre (ovviamente) le date delle istanze IdT1, … , IdTn devono essere tutte uguali.

Quindi si è dovuti cercare di scegliere come data una a caso ma solo tra quelle presenti per la relativa istanza di tratta relativa. In aggiunta, si è dovuto trovare un metodo per verificare se una tratta fosse successiva ad un'altra all'intero di un volo, nel caso anche tale tratta avrebbe avuto la stessa data.
Nella prima parte della generazione di *prenotazione_IstanzaDiTratta* si crea una matrice sparsa sotto forma di liste di liste, le liste sono nominate con le possibili tratte percorribili e contengono rispettivamente le date possibili per quella tratta. Questo ha permesso di ottimizzare il codice, che per ogni riga di *"primeTratte"* (circa 1 milione), si estrae una data casuale tra quelle possibili.
Nella seconda parte si selezionano tutte le tratte che han *"numero_progressivo"* uguale a 2 (ovvero che son tutte *secondeTratte*, che non è uguale a *secondaTratta* usata nella generazione di *volo_Tratta*) e si replica la data appartenente alla rispettiva "tratta di scalo" (ovvero quella che ha *"numero_progressivo"* uguale a 1 e che ha lo stesso codice del volo). Il tutto viene inserito all'interno del dataframe *prenotazione_IstanzaDiTratta* e successivamente si genera un posto_prenotato che appartenga effettivamente al volo (ovvero che sia un valore compreso tra 1 e il numero di posti disponibili dentro a quel specifico aereo)

Tutto il restante codice per la generare dei dati pseudo randomici è contenuta nel file *"./generateFakeData.R"*.

7 Analisi dei dati

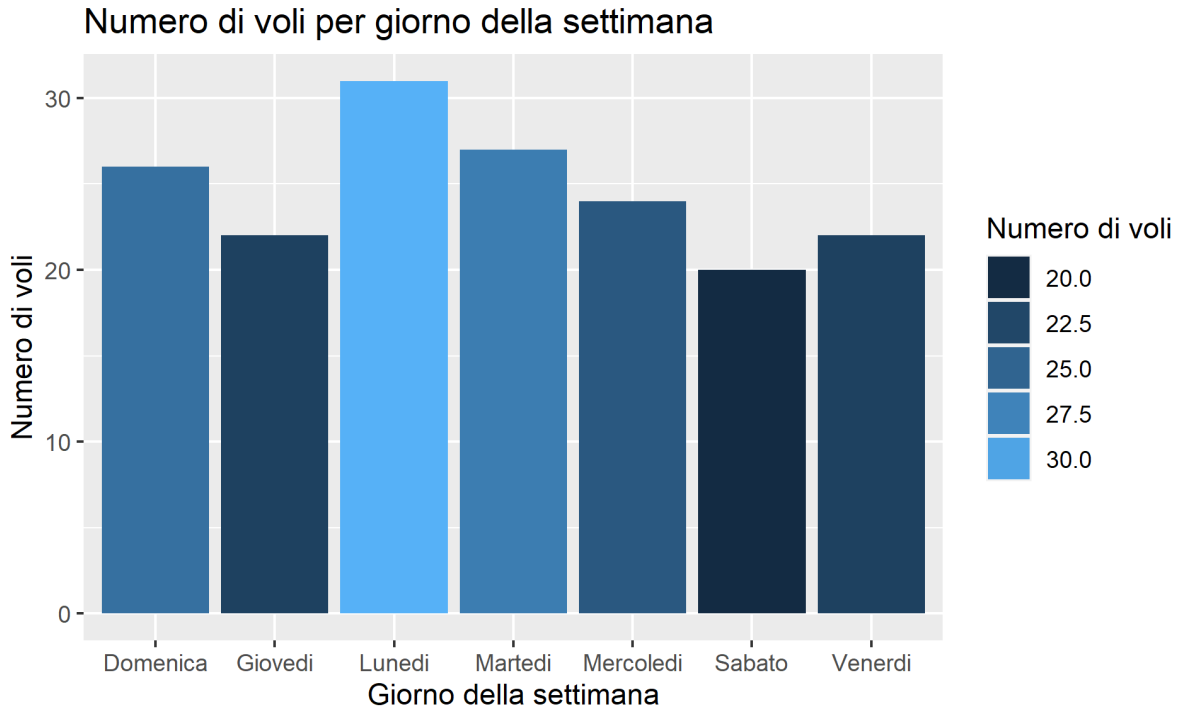
Siccome i dati sono stati generati in modo pseudo casuale ci si attende una leggera varianza nei dati ottenuti ed una distribuzione normale.

7.1 Giorni della settimana con più voli

In questa query, si vuole determinare il numero di voli durante i giorni della settimana. Si osservi che il giorno **Lunedì** è quello con più voli.

```
select giorno_settimana, count(*) as numero_voli
from giornidellasettimana_volo
group by giorno_settimana;
```

```
ggplot(
    data = giornisettimana_volo_count_df,
    aes(x = giorno_settimana, y = numero_voli, fill = numero_voli)
  ) +
  geom_bar(stat = "identity") +
  labs(
    title = "Numero di voli per giorno della settimana",
    x = "Giorno della settimana",
    y = "Numero di voli"
  )
```

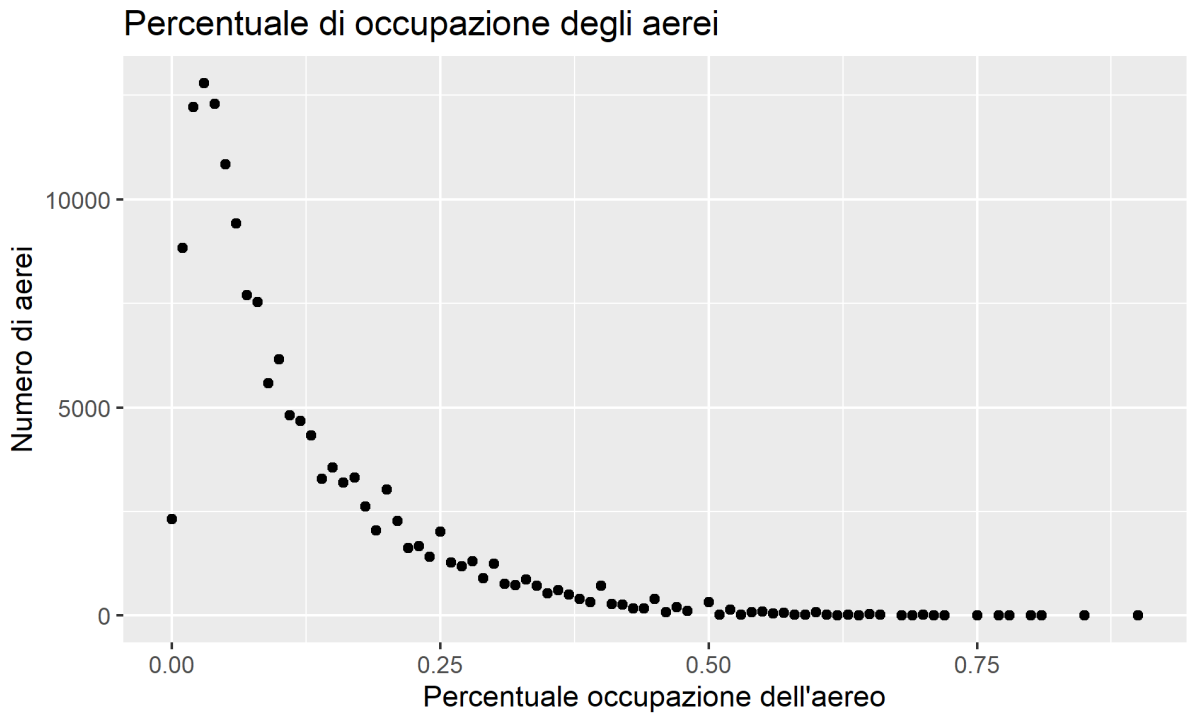


7.2 Percentuale di occupazione degli aerei

L'obiettivo di questa query è di conoscere la percentuale di occupazione degli aerei, ovvero il rapporto tra il numero totale di prenotazioni relative ad una certa istanza di tratta e il numero di posti disponibili totali sull'aereo in quella istanza di tratta, per ogni istanza di tratta.

```
select trunc(perc_occup_aereo, 2) as perc_occupazione_aereo,
count(trunc(perc_occup_aereo, 2)) as numero_aerei
from posti_rimanenti_info
group by perc_occupazione_aereo
order by perc_occupazione_aereo;
```

```
ggplot(
    data = percentuale_occupazione_aerei_df,
    aes(x = perc_occupazione_aereo, y = numero_aerei)
  ) +
  geom_point() +
  labs(
    title = "Percentuale di occupazione degli aerei",
    x = "Percentuale occupazione aereo",
    y = "Numero di aerei"
  )
```

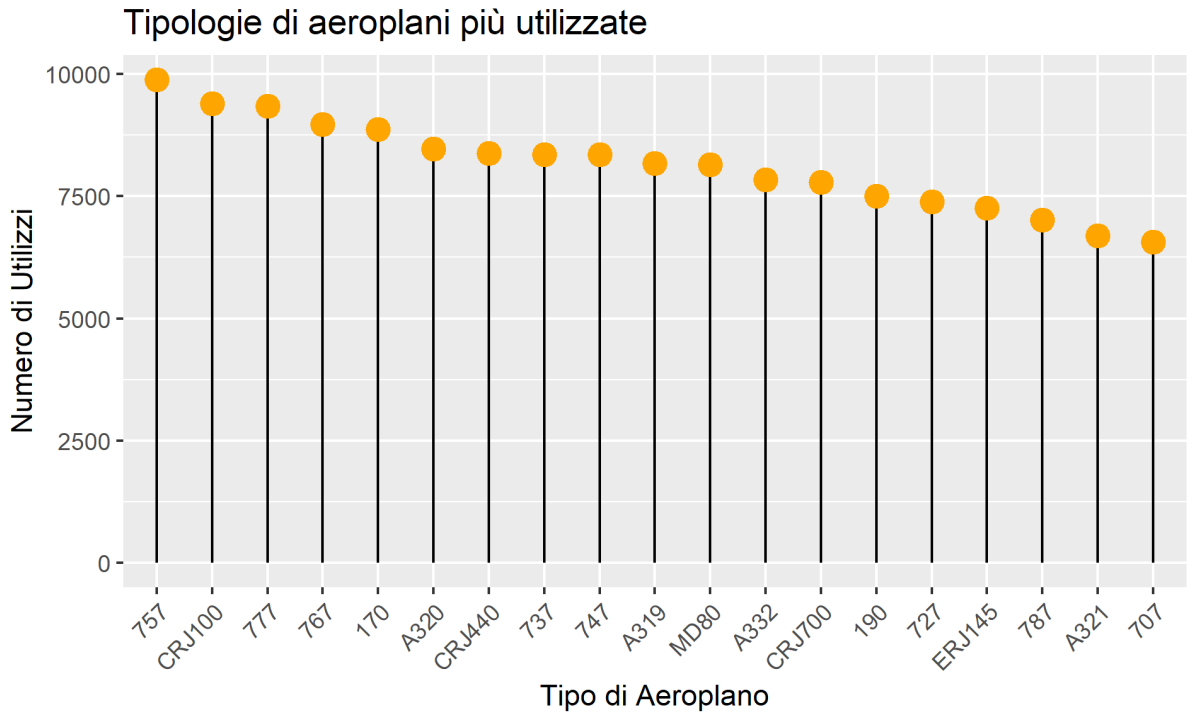


7.3 Tipologie di aeroplano più utilizzate

Qui si è interessati a conoscere quali sono i tipi di aeroplani più utilizzati.

```
select a.tipo_aeroplano, count(*) as numero_tipi
from istanzaditratta idt
join aeroplano a on a.codice = idt.aeroplano
group by tipo_aeroplano
order by numero_tipi;
```

```
ggplot(
    data = compagnia_aeree_economiche_df,
    aes(x = reorder(compagnia_aerea, +costo_medio), y =
costo_medio, fill = compagnia_aerea)
  ) +
  geom_bar(stat = "identity") +
  etichette_asse_x_diagonale +
  labs(
    title = "Compagnie aeree più economiche",
    x = "Compagnia Aerea",
    y = "Costo Medio"
  ) +
  guides(fill="none")
```

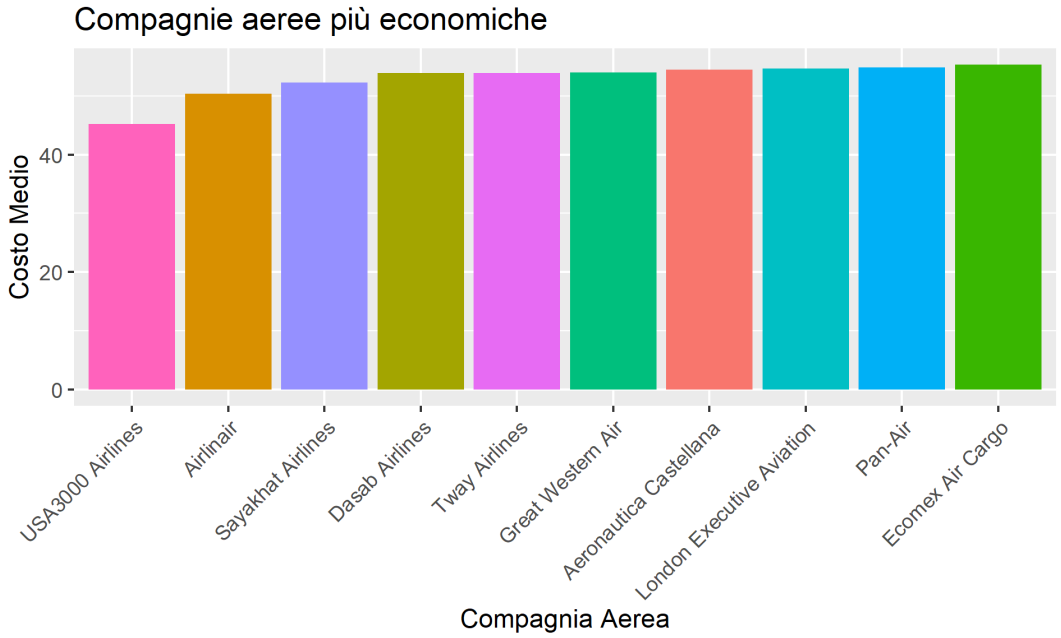


7.4 Le compagnie aeree più economiche

In questa interrogazione, si vuole scoprire quali sono le principali compagnie aeree che offrono voli economici. In particolare, si osservi che la compagnia aerea **USA 3000 Airlines** è quella che offre voli in media a 45$.

```
select trunc(((sum(prezzo))::decimal / count(*)), 2)
as costo_medio, compagnia_aerea
from prezzi_voli_compagnia_aerea
group by compagnia_aerea
order by costo_medio asc
limit 10;
```

```
ggplot(
    data = compagnia_aeree_economiche_df,
    aes(x = reorder(compagnia_aerea, +costo_medio), y =
costo_medio, fill = compagnia_aerea)
  ) +
  geom_bar(stat = "identity") +
  etichette_asse_x_diagonale +
  labs(
    title = "Compagnie aeree più economiche",
    x = "Compagnia Aerea",
    y = "Costo Medio"
  ) +
  guides(fill="none")
```



Conclusioni

Il progetto ha permesso di entrare nel dettaglio di tutte le fasi della realizzazione di un database: dall'analisi dei requisiti iniziale, alle progettazione -concettuale prima e logica poi-, alla realizzazione fisica. Inoltre sono state approfondite le conoscenze di R sia per la generazione di dati e tabelle pseudo casuali sia per l'analisi dei dati e la creazione di grafici rappresentativi.

In R si sono è appreso l'uso di alcune delle funzionalità di *tidyverse*, una delle librerie più utilizzate, tra cui sono presenti *dplyr*, fondamentale per la pulizia e l'elaborazione dei dati, e *ggplot* per la creazione di grafici.
Inoltre per la popolazione del database e l'esecuzione delle query direttamente da R si è utilizzato *RPostgreSQL.*