Università degli Studi di Trieste
Artificial Intelligence & Data Analytics

# Azienda di trasporti di Trieste

Facchin Alberto
SM3201282

# Presentazione progetto

---

Un’azienda di trasporti del comune di Trieste necessita di un database per poter salvare e gestire i dati relativi a tutti i mezzi, le linee, i dipendenti, gli uffici, gli autisti ed i passeggeri.
Si vuole poter fornire ai passeggeri, previa registrazione, la possibilità di prenotare il proprio posto a sedere all’interno del mezzo, in modo da evitare di non poter usufruire del servizio a causa del sovraffollamento di quel determinato autobus.
Ogni passeggero deve avere la possibilità di poter controllare quanti posti a sedere rimangono per una determinata tratta, così da poter, nell’eventualità, organizzarsi per prendere l’autobus precedente o successivo, inoltre deve poter prenotare anche più di un posto nella stessa prenotazione.
La tratta è il percorso compreso tra due fermate, anche non successive, ed ogni tratta può essere percorsa da mezzi di linee diverse.
Il costo totale della prenotazione è determinato dalla quantità di posti prenotati moltiplicata al costo di una determinata tratta (la tratta ha sempre lo stesso costo).
Tutti i dipendenti devono essere assegnati ad uno degli uffici presenti all’interno del comune, e nessun dipendente può lavorare in più di un ufficio.
Ogni mezzo deve essere guidato da un solo autista, ma quest’ultimo può, durante l’arco della giornata, guidare mezzi diversi, a seconda delle linee che deve percorrere.
I mezzi vengono assegnati ad una sola linea e non possono effettuare tratte che non siano incluse in essa.

# Analisi del problema

---

Nella seguente sezione analizzo, periodo per periodo, il testo soprastante per individuare le entità, le relazioni e gli attributi.
Non è esclusa l'ipotesi di dover aggiungere ulteriori entità, relazioni ed attributi in seguito.
Ho riportato in grassetto delle frasi o parole chiave da cui ho tratto alcune mie conclusioni.

"Una società di trasporti del **comune di Trieste** necessita di un **database** per poter **salvare e gestire i dati relativi a tutti i mezzi, le linee, i dipendenti, gli uffici, gli autisti ed i passeggeri.**"

- ➔ Entità Mezzo con attributo targa
- ➔ Entità Linea contrassegnata da una denominazione
- ➔ Entità Dipendente con attributi IBAN, livello
- ➔ Entità Ufficio
- ➔ Entità Autista con attributo IBAN
- ➔ Entità Passeggero

"Si vuole poter fornire ai passeggeri, **previa registrazione**, la possibilità di **prenotare il proprio posto a sedere all'interno del mezzo**, in modo da evitare di non poter usufruire del servizio a causa del sovraffollamento di quel determinato autobus."

- ➔ Entità Account con attributi email e password
- ➔ Entità Prenotazione con attributi data e ora della prenotazione.
- ➔ Passeggero avrà come attributi quelli relativi al metodo di pagamento, da utilizzare per la prenotazione dei posti a sedere
- ➔ Relazione tra Passeggero e Prenotazione
- ➔ Mezzo avrà come attributo il numero di posti disponibili

“Ogni passeggero deve avere la possibilità di poter controllare quanti posti a sedere rimangono per una determinata tratta, così da poter, nell’eventualità, organizzarsi per prendere l’autobus precedente o successivo, inoltre **deve poter prenotare anche più di un posto nella stessa prenotazione**.”

➔ Prenotazione avrà come attributo anche quantità

“La tratta è il percorso compreso tra due fermate, anche non successive, ed ogni tratta può essere percorsa da mezzi di linee diverse.”

➔ Entità Fermata con attributi indirizzo e riferimento. Un esempio di riferimento può essere “di fronte al supermercato”

➔ Entità Tratta

➔ Relazione tra Tratta e Fermata

“Il costo totale della prenotazione è determinato dalla quantità di posti prenotati moltiplicata al **costo di una determinata tratta** (la tratta ha sempre lo stesso costo).”

➔ Tratta avrà come attributo il costo della stessa

➔ Relazione tra Tratta e Prenotazione

“Tutti i dipendenti devono essere assegnati ad uno degli uffici presenti all’interno del comune, e nessun dipendente può lavorare in più di un ufficio.”

➔ Relazione tra Dipendente ed Ufficio

“Ogni mezzo deve essere guidato da un solo autista, ma quest’ultimo può, durante l’arco della giornata, guidare mezzi diversi, a seconda delle linee che deve percorrere.”

➔ Relazione tra Mezzo ed Autista

“I mezzi vengono assegnati ad una sola linea e non possono effettuare tratte che non siano incluse in essa.”

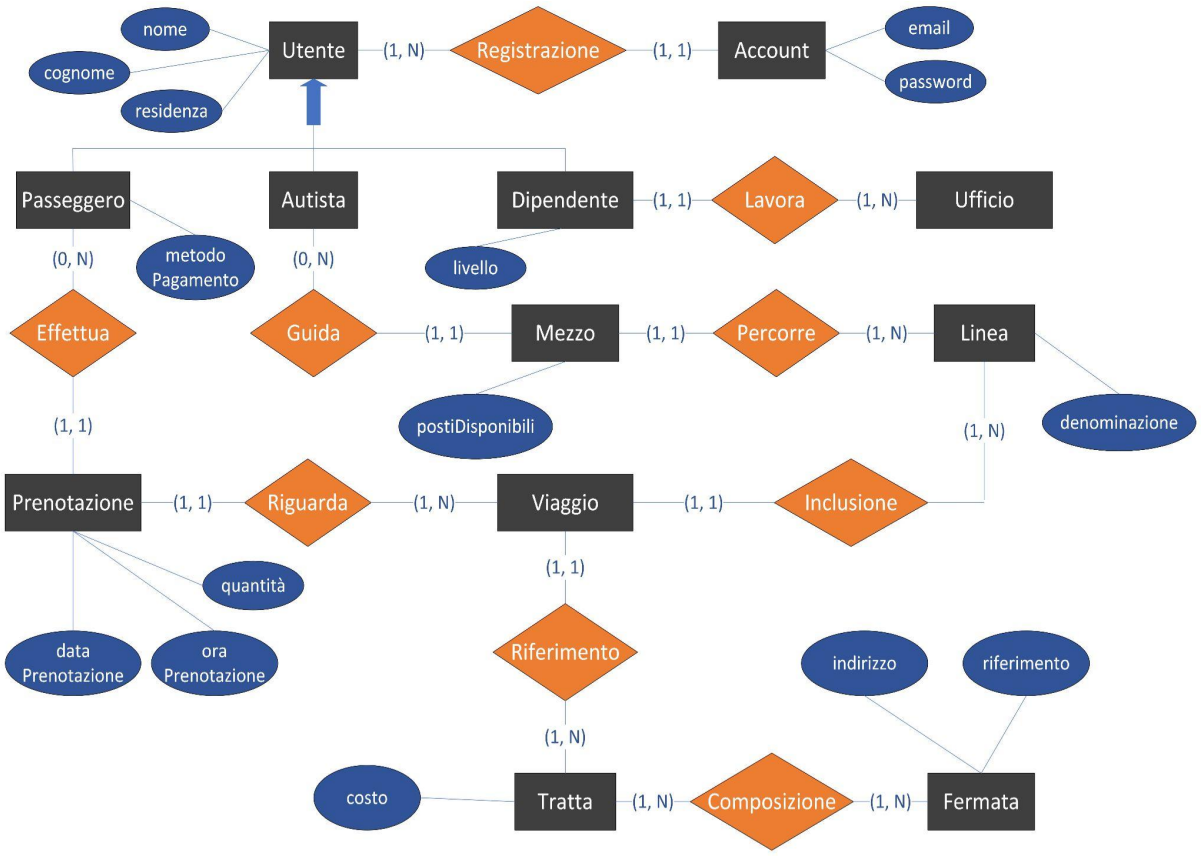
➔ Relazione tra Mezzo e Linea

# Considerazioni generali

---

Tratta è in relazione con Prenotazione e Fermata, e per quanto riguarda la prima relazione sembra che non sussistano problemi.
Analizzando, però, la relazione tra Tratta e Fermata, essendo una “molti a molti”, in quanto una tratta include n fermate e ogni fermata può essere inclusa in n tratte, si potrebbe rendere la loro relazione un’entità, ma verrebbe meno la possibilità di gestire in  modo efficiente il fatto che la stessa tratta può essere percorsa in orari differenti, quindi necessita di attributi per indicare l’orario di partenza e l’orario di arrivo, anche perché è fondamentale per il passeggero sapere queste due informazioni.
La strategia che utilizzo è, quindi, quella di introdurre una nuova entità Viaggio, la quale sarà in relazione con Prenotazione, Tratta e Linea, in modo da poterci inserire all’interno gli attributi per l’orario di partenza e arrivo.

# Schema concettuale (Entity-Relationship)




Non sono stati tutti gli attributi delle entità per evitare di riempire troppo lo schema.
In particolare non è presente l'attributo "Codice" o "ID" relativo ad ogni entità, ma ogni entità ha un proprio ID o codice, utilizzato come identificatore interno.

Nella relazione "registrazione" la cardinalità dalla parte di Utente è (1, N) in quanto ogni utente può avere molteplici account, anche considerando il fatto che un dipendente o un autista potrebbe, mentre non lavora o proprio per andare al lavoro, utilizzare un mezzo per i propri spostamenti, di conseguenza dovrà avere anche un account passeggero.

# Vincoli non esprimibili graficamente

---

Considerando il livello di ogni dipendente c'è il vincolo relativo allo stipendio, in particolare un dipendente di livello 2 non può percepire uno stipendio inferiore ad un dipendente di livello 1.
I livelli sono suddivisi come segue:

- Livello 1 → dipendente base
- Livello 2 → dipendente esperto
- Livello 3 → capo ufficio

# Dizionario dei dati

---

Entità

| **Entità** | **Descrizione** | **Attributi** | **Identificatore** |
|---|---|---|---|
| Account | Credenziali di ogni utente | IdAccount, email, password | IdAccount |
| Utente | Persona interna o esterna all'azienda che interagisce con essa | IdUtente, nome, cognome, dataNascita, residenza, città, telefono | IdUtente |
| Passeggero | Passeggeri che usufruiscono dei mezzi | IdPasseggero, metodoPagamento, numeroCarta, scadenzaCarta, CVV, CF | IdPasseggero |
| Autista | Autisti dei mezzi | IdAutista, dataAssunzione, | IdAutista |

| | | | |
|---|---|---|---|
| | | dataScadenzaContratto, IBAN, stipendio | |
| Dipendente | Dipendenti dei vari uffici | IdDipendente, dataAssunzione, dataScadenzaContratto, IBAN, livello, stipendio | IdDipendente |
| Ufficio | Uffici dell'azienda di trasporti | IdUfficio, indirizzo, città, provincia, CAP | IdUfficio |
| Mezzo | Mezzi utilizzati per gli spostamenti | IdMezzo, targa, postiDisponibili | IdMezzo |
| Linea | Linee su cui viaggiano i mezzi | IdLinea, denominazione | IdLinea |
| Prenotazione | Prenotazioni effettuate dai passeggeri per un determinato viaggio | IdPrenotazione, dataPrenotazione, quantità | IdPrenotazione |
| Viaggio | Spostamento che il passeggero prenota per un determinato orario | IdViaggio, data, oraPartenza, oraArrivo | IdViaggio |
| Tratta | Percorso compreso tra due diverse fermate | IdTratta, costo | IdTratta |
| Fermata | Luogo in cui i mezzi si fermano per far salire/scendere i passeggeri | IdFermata, indirizzo, riferimento | IdFermata |

Relazioni

| **Relazione** | **Descrizione** | **Componenti** |
|---|---|---|
| Registrazione | Ogni utente deve registrarsi e creare uno | Utente, Account |

| | | |
|---|---|---|
| | o più account | |
| Lavora | Ciascun dipendente lavora presso un ufficio | Dipendente, Ufficio |
| Effettua | Ogni passeggero effettua prenotazioni | Passeggero, Prenotazione |
| Guida | Ciascun autista può guidare mezzi diversi nell'arco della giornata | Autista, Mezzo |
| Percorre | Ogni mezzo percorre una linea | Mezzo, Linea |
| Riguarda | Ciascuna prenotazione riguarda un solo determinato viaggio | Prenotazione, Viaggio |
| Inclusione | Ogni viaggio include una sola linea | Viaggio, Linea |
| Riferimento | Ciascun viaggio fa riferimento ad una sola tratta | Viaggio, Tratta |
| Composizione | Ogni tratta è composta da molteplici fermate | Tratta, Fermata |

Non ho inserito la colonna "attributi" delle relazioni perché non ho relazioni con attributi.

# Eliminazione generalizzazioni

---

L'unica generalizzazione presente nello schema E-R è la generalizzazione parziale che coinvolge le entità:

- Utente (genitore)
- Passeggero (figlio)
- Autista (figlio)

- Dipendente (figlio)

Considerando le varie possibilità, per eliminare questa generalizzazione, ho deciso di accorpare l'entità genitore alle entità figlio, in quanto gli accessi ai figli sono distinti.
Non sarebbe stato molto utile fare il contrario, quindi accorpare le entità figlio nell'entità genitore, perché gli accessi non sono contestuali, infatti, molto probabilmente, sarebbe complicato gestire la relazione tra dipendente ed ufficio.
In particolar modo avrei dovuto inserire un attributo in Utente per specificare il tipo di utente e poi avere un campo opzionale per la relazione con ufficio.
Altrimenti avrei potuto inserire una relazione tra Utente ed i figli, ma conviene solo se gli accessi ai figli ed al genitore sono separati ed, in questo caso, non è fondamentale distinguere l'accesso ad Utente dagli accessi ai figli, perciò, in conclusione, ho preferito accorpare il genitore ai figli.

# Operazioni di interesse

---

| | **Operazione** | **Frequenza** | **Tipo** |
|---|---|---|---|
| 1. | Selezionare i posti rimanenti per i mezzi di una determinata linea in una certa fascia oraria | 100.000/giorno | Interattiva |
| 2. | Selezionare nome, cognome e telefono di tutti i passeggeri che hanno effettuato una prenotazione | 1/anno | Interattiva |

| | | | |
|---|---|---|---|
| | in una particolare data e che hanno pagato con carta di credito | | |
| 3. | Selezionare tutti i mezzi che percorrono una determinata tratta | 1/mese | Interattiva |
| 4. | Selezionare il costo totale per un certo numero di biglietti per una tratta | 100·000/giorno | Interattiva |
| 5. | Inserire un nuovo viaggio | 1/anno | Interattiva |
| 6. | Selezionare tutti i dipendenti di un ufficio | 2/mese | Batch |
| 7. | Creazione nuovo account | 50/giorno | Interattiva |
| 8. | Diminuire il numero di posti disponibili in un mezzo dopo la prenotazione | 100·000/giorno | Batch |

# Tavola dei volumi

---

Si stimano:

- 50·000 passeggeri
- 300 autisti
- 400 dipendenti (addetti alla manutenzione, addetti alla pulizia dei veicoli, personale agli sportelli, personale amministrativo)
- 50·700 utenti (50·000 + 300 + 400)
- 51·050 account (1 account per ogni passeggero, 1.5 account per ogni dipendente ed 1.5 account per ogni autista)
- 10 uffici
- 200 mezzi
- 20 linee

- 25·000·000 prenotazioni (2 al giorno per ogni passeggero, per 250 giorni lavorativi in un anno)
- 36·000·000 viaggi (2 al giorno per ogni passeggero, per 360 giorni all'anno)
- 4·200 tratte (15 x 14 = 210, per ogni linea)
- 300 fermate (15 fermate per ogni linea)

| **Concetto** | **Tipo** | **Volume** |
|---|---|---|
| Account | E | 51·050 |
| Utente | E | 50·700 |
| Passeggero | E | 50·000 |
| Autista | E | 300 |
| Dipendente | E | 400 |
| Ufficio | E | 10 |
| Mezzo | E | 200 |
| Linea | E | 20 |
| Prenotazione | E | 25·000·000 |
| Viaggio | E | 36·000·000 |
| Tratta | E | 4·200 |
| Fermata | E | 300 |
| Registrazione | R | 51·050 |
| Lavora | R | 400 |
| Effettua | R | 25·000·000 |
| Guida | R | 300 |

| | | |
|---|---|---|
| Percorre | R | 200 |
| Riguarda | R | 36.000.000 |
| Inclusione | R | 36.000.000 |
| Riferimento | R | 36.000.000 |
| Composizione | R | 4.200 |

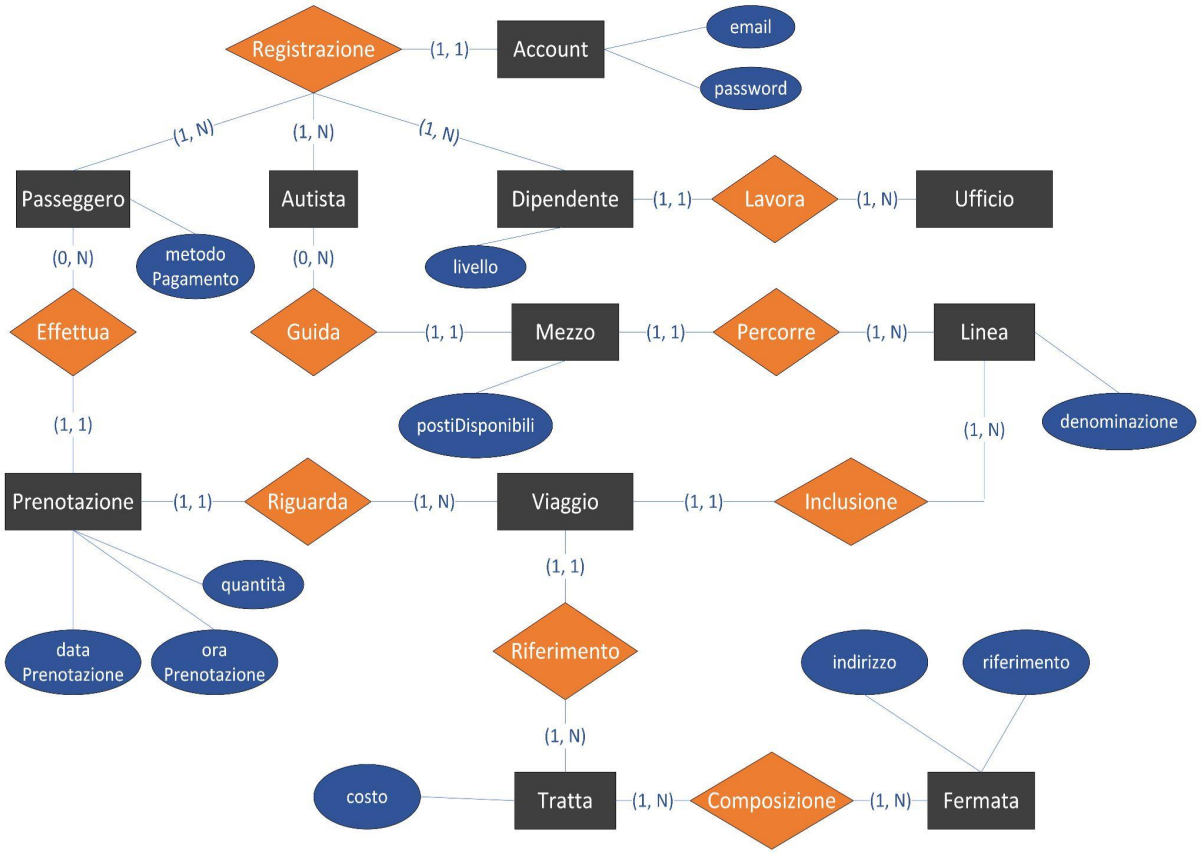
# Analisi delle ridondanze

---

| | | | |
|---|---|---|---|
| 4. | Selezionare il costo totale per un certo numero di biglietti per una tratta | 100.000/giorno | Interattiva |

Si potrebbe aggiungere un campo calcolato, per semplificare le interrogazioni, nell'entità Viaggio, per determinare il costo totale della prenotazione, determinato dalla quantità di posti prenotati e dal costo della tratta.
Essendo un'operazione non particolarmente "pesante" dal punto di vista computazionale ho deciso di non aggiungere un campo calcolato per non creare ridondanza, infatti il prezzo del viaggio può essere semplicemente fornito al passeggero moltiplicando l'attributo costo presente in Tratta per l'attributo quantità presente in Prenotazione, essendo l'entità Viaggio legata, attraverso una relazione, sia con Prenotazione che con Tratta.

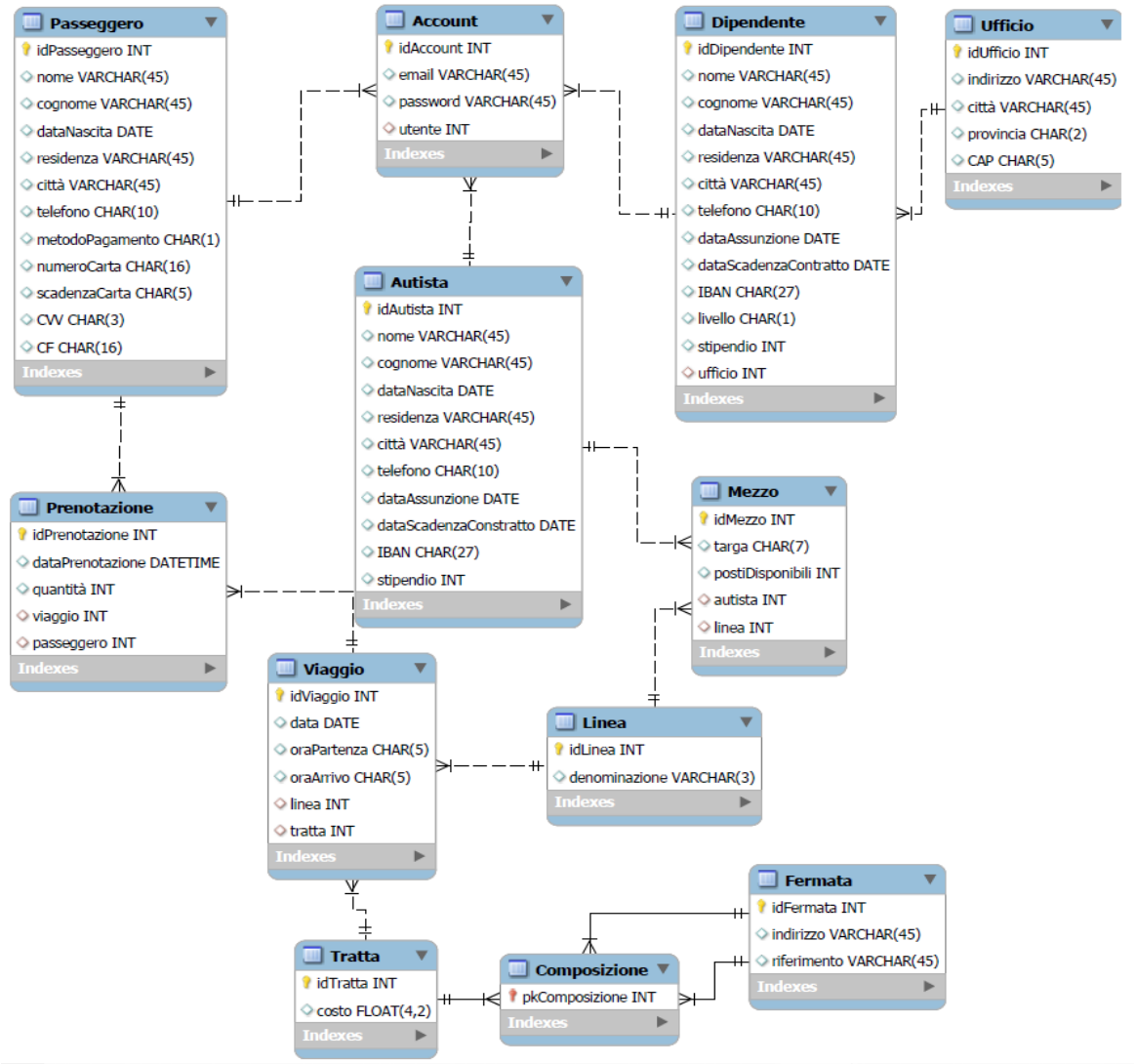
# Schema E-R ristrutturato




# Schema logico

**Account** (idAccount, email, password, utente*)
**Passeggero** (idPasseggero, nome, cognome, dataNascita, residenza, città, telefono, metodoPagamento, numeroCarta, scadenzaCarta, CVV, CF)
**Autista** (idAustista, nome, cognome, dataNascita, residenza, città, telefono, dataAssunzione, dataScadenzaContratto, IBAN, stipendio)

**Dipendente** (idDipendente, nome, cognome, dataNascita, residenza, città, telefono, dataAssunzione, dataScadenzaContratto, IBAN, livello, stipendio, ufficio*)
**Ufficio** (idUfficio, indirizzo, città, provincia, CAP)
**Mezzo** (idMezzo, targa, postiDisponibili, autista*, linea*)
**Linea** (idLinea, denominazione)
**Prenotazione** (idPrenotazione, dataPrenotazione, quantità, viaggio*, passeggero*)
**Viaggio** (idViaggio, data, oraPartenza, oraArrivo, linea*, tratta*)
**Tratta** (idTratta, costo)
**Composizione** (tratta*, fermataPartenza*, fermataArrivo*)
**Fermata** (idFermata, indirizzo, riferimento)



# Forme normali

---

Tutte le tabelle sono in prima forma normale in quanto tutte le colonne sono atomiche.

Tutte le tabelle sono in seconda forma normale in quanto sono in 1NF e non ci sono chiavi primarie composte, per questo motivo ho deciso di utilizzare un Id, per ogni tabella, come chiave primaria.
L'unica tabella che ha una chiave composta è Composizione, originata dalla relazione molti a molti tra Tratta e Fermata, ma, in ogni caso, anche Composizione è in 2NF perché è in 1NF ed ogni colonna dipende in senso stretto dalla chiave primaria.

Tutte le tabelle sono in terza forma normale in quanto sono in 2NF ed ogni attributo dipende solo dalla chiave primaria, tranne la tabella Ufficio che contiene l'attributo CAP, il quale dipende da provincia, città e indirizzo (anche indirizzo perché il CAP non è unico per tutta Trieste). Per risolvere questo problema dovrei creare una tabella di lookup di nome "Luogo" in cui ci sono i tre attributi sopracitati, ma, in questo caso, preferisco non aggiungere un'altra tabella e lasciare Ufficio così com'è per garantire la possibilità di avere accesso diretto a tutte le informazioni relative a tutti gli uffici.

# Comandi SQL significativi

- Selezionare i posti rimanenti per i mezzi di una determinata linea in una certa fascia oraria

```
# Selezionare i posti rimanenti per i mezzi di una determinata linea in una certa fascia oraria
drop procedure posti_rimanenti;
DELIMITER $$
create procedure posti_rimanenti(IN linea_input varchar(3),
IN partenza_input CHAR(5), IN arrivo_input CHAR(5))
BEGIN
    select l.denominazione as linea, m.postiDisponibili as posti, v.data,
    v.oraPartenza as partenza, v.oraArrivo as arrivo
    from Viaggio v
    inner join Linea l
    on v.linea = l.idLinea
    inner join Tratta t
    on v.tratta = t.idTratta
    inner join Mezzo m
    on l.idLinea = m.linea
    where l.denominazione = linea_input
    and v.oraPartenza >= partenza_input and v.oraArrivo <= arrivo_input
    and m.postiDisponibili <> 0;
END $$
DELIMITER ;

set @linea_input = "17/";
set @partenza_input = "00:00";
set @arrivo_input = "23:59";
call posti_rimanenti(@linea_input, @partenza_input, @arrivo_input);
```

- Aggiornare il numero di posti rimanenti in un mezzo dopo una prenotazione effettuata da un passeggero.
Prima della prenotazione devo controllare che, prenotando un certo numero di posti a sedere, quelli rimanenti non diventino un numero minore di zero.

```
# trigger per controllare che, prenotando una determinata quantità di posti,
# il numero di posti rimanenti non diventi <0
drop trigger trg_zero;
DELIMITER $$
CREATE TRIGGER trg_zero
BEFORE INSERT ON prenotazione
FOR EACH ROW
BEGIN
    DECLARE num_posti INT;
    select sum(postiDisponibili) into num_posti
    from Mezzo m
    inner join Linea l
    on m.linea = l.idLinea
    inner join Viaggio v
    on v.linea = l.idLinea
    inner join Prenotazione p
    on p.viaggio = v.idViaggio
    where NEW.viaggio = v.idViaggio;

    if (num_posti - NEW.quantità) < 0 THEN
        signal sqlstate "02000"
        set message_text = "Numero posti rimasti insufficiente";
    end if;
END $$
DELIMITER ;

# trigger per aggiornare il numero di posti rimanenti in un mezzo
# dopo una prenotazione effettuata da un passeggero
drop trigger trg_posti;
DELIMITER $$
CREATE TRIGGER trg_posti
AFTER INSERT ON prenotazione
FOR EACH ROW
BEGIN
    update mezzo as m
    inner join (select idMezzo
    from Mezzo m
    inner join Linea l
    on m.linea = l.idLinea
    inner join Viaggio v
    on v.linea = l.idLinea
    inner join Prenotazione p
    on p.viaggio = v.idViaggio
    where NEW.viaggio = v.idViaggio) as tmp on m.idMezzo = tmp.idMezzo
    set postiDisponibili = postiDisponibili - NEW.quantità;
END $$
DELIMITER ;
```

- Visualizzare le informazioni di recup di una determinata prenotazione, quindi codice, data, orario partenza, orario arrivo, quantità di biglietti acquistati e costo totale

```
# visualizzare informazioni di recup di una determinata prenotazione
# usd per calcolare il costo totale
drop function costo_totale;
DELIMITER $$
create function costo_totale(prenotazione_input INT, quantita_input INT)
returns float (5, 2)
deterministic
BEGIN
    declare costo_tot float (5, 2);
    select (quantita_input * t.costo) into costo_tot
    from Tratta t
    inner join Viaggio v
    on t.idTratta = v.tratta
    inner join Prenotazione p
    on p.viaggio = v.idViaggio
    where p.idPrenotazione = prenotazione_input;
    return costo_tot;
END $$
DELIMITER ;

set @prenotazione_input = 3;

select p.idPrenotazione as codice, v.data, v.oraPartenza as partenza,
v.oraArrivo as arrivo, p.quantità, concat(costo_totale(@tratta_input, p.quantità), "€") as totale
from Prenotazione p
inner join Viaggio v
on p.viaggio = v.idViaggio
inner join Tratta t
on t.idTratta = v.tratta
where p.idPrenotazione = @prenotazione_input;
```