Basi di Dati e Conoscenza

Progetto A.A. 2019/2020

SISTEMA INFORMATIVO DI UN AZIENDA DI TRASPORTO PUBBLICO

0259244

Ivan Palmieri

Indice

1. Descrizione del Minimondo

Si intende realizzare il sistema informativo di un'azienda di trasporto pubblico locale. L'azienda è dotata di un parco veicoli che permettono di coprire un determinato insieme di tratte. I veicoli sono caratterizzati da una matricola (codice univoco numerico di quattro cifre). Ogni veicolo è anche associato ad una data di acquisto e ad uno storico di manutenzione.

Ciascuna tratta ha un insieme di fermate identificate da latitudine e longitudine ed associata ad un codice numerico univoco di cinque cifre. La prima e l'ultima fermata sono contrassegnate come capolinea. Inoltre, il percorso tra una fermata e l'altra è identificato da un insieme di waypoint, ciascuno caratterizzato da una latitudine ed una longitudine.

Ciascuna tratta viene coperta da un numero predefinito di veicoli, la cui associazione viene gestita dai gestori del servizio. Ogni capolinea ha un orario di partenze prestabilito. Gli amministratori del servizio gestiscono anche i conducenti, identificati da un codice fiscale, un nome, un cognome, una data di nascita ed un luogo di nascita. Di ogni conducente è di interesse conoscere anche il numero di patente e la data di scadenza della stessa.

I gestori del servizio devono poter gestire l'orario di lavoro dei conducenti, organizzati in turni di otto ore. Un conducente deve effettuare 5 turni a settimana. La gestione dei turni avviene da parte dei gestori del servizio su base mensile. Qualora un conducente si ponga in malattia, i gestori del servizio devono poter indicare che il conducente non ha coperto il turno per malattia e identificare un nuovo conducente cui assegnare la sostituzione del turno.

Ogni veicolo è equipaggiato di un dispositivo GPS che, ogni 5 secondi, comunica le coordinate geografiche in cui si trova il veicolo. Gli utenti del sistema possono accedere al servizio per conoscere, dato il codice di una fermata, a quale distanza si trova un veicolo. La distanza deve essere calcolata andando a prendere in considerazione tutti i waypoint che intercorrono tra la posizione attuale dell'autoveicolo e la fermata di interesse. Si noti che per calcolare la distanza tra due coordinate geografiche è possibile utilizzare la seguente formula, dove r è il raggio della Terra:

d=2r*sin^(-1)(sqrt(sin^(2)((phi_{2}-phi_{1})/(2))+cos(phi_{1})cos(phi_{2})sin^(2)((lambda_{2}-lambda_{1})/(2))))

Quando salgono a bordo, gli utenti del servizio timbrano un biglietto elettronico o un abbonamento sul “validatore intelligente” installato sui veicoli. Nel caso di un biglietto elettronico, questo viene marcato come “utilizzato” all’interno del sistema. L’emissione di nuovi biglietti viene amministrata dai gestori del servizio. Nel caso dell’utilizzo di un abbonamento, il sistema tiene traccia dell’ultimo utilizzo dello stesso.

Quando un autista si trova ad un capolinea, può interrogare il sistema per sapere qual è la prossima partenza prevista del veicolo che sta guidando.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
8	Percorso	Tratta	Precedentemente e' stato dichiarato piu volte con il termine tratta.
25	Autoveicolo	Veicolo	Precedentemente e' stato dichiarato piu volte con il termine veicolo.
7	Ultima fermata	Capolinea Finale	Sia l'ultima fermata che la prima vengono chiamate capolinia.
7	Prima fermata	Capolinea iniziale	Sia l'ultima fermata che la prima vengono chiamate capolinia.
12	Amministratori del servizio	Amministratori	Il termine "del servizio" risulta essere ridondante
11,15, 17,18, 33	Gestori del servizio	Amministratori	Precedentemente sono stati dichiarati con il termine amministatori di servizio.
35	Autista	Conducente	Precedentemente e' stato dichiarato piu volte con il termine conducente.

Specifica disambiguata

Si intende realizzare il sistema informativo di un'azienda di trasporto pubblico locale. L'azienda è dotata di un parco veicoli che permettono di coprire un determinato insieme di tratte. I veicoli sono caratterizzati da una matricola (codice univoco numerico di quattro cifre). Ogni veicolo è anche associato ad una data di acquisto e ad uno storico di manutenzione.

Ciascuna tratta ha un insieme di fermate identificate da latitudine e longitudine ed associata ad un codice numerico univoco di cinque cifre. La prima fermata e l'ultima fermata sono contrassegnate come capolinea iniziale e capolinea finale. Inoltre, la tratta tra una fermata e l'altra è identificato da
un insieme di waypoint, ciascuno caratterizzato da una latitudine ed una longitudine.

Ciascuna tratta viene coperta da un numero predefinito di veicoli, la cui associazione viene gestita dagli amministratori. Ogni capolinea ha un orario di partenze prestabilito. Gli amministratori gestiscono anche i conducenti, identificati da un codice fiscale,

un nome, un cognome, una data di nascita ed un luogo di nascita. Di ogni conducente è di interesse conoscere anche il numero di patente e la data di scadenza della stessa.

Gli amministratori devono poter gestire l'orario di lavoro dei conducenti, organizzati in turni di otto ore. Un conducente deve effettuare 5 turni a settimana. La gestione dei turni avviene da parte degli amministratori su base mensile. Qualora un conducente si ponga in malattia, gli amministratori devono poter indicare che il conducente non ha coperto il turno per malattia e identificare un nuovo conducente cui assegnare la sostituzione del turno.

Ogni veicolo è equipaggiato di un dispositivo GPS che, ogni 5 secondi, comunica le coordinate geografiche in cui si trova il veicolo.I passeggeri del sistema possono accedere al servizio per conoscere, dato il codice di una fermata, a quale distanza si trova un veicolo. La distanza deve essere calcolata andando a prendere in considerazione tutti i waypoint che intercorrono tra la posizione attuale veicolo e la fermata di interesse. Si noti che per calcolare la distanza tra due coordinate geografiche è possibile utilizzare la seguente formula, dove r è il raggio della Terra:

$$d = 2r \arcsin\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}$$

Quando salgono a bordo, i passeggeri del servizio timbrano un biglietto elettronico o un abbonamento sul "validatore intelligente" installato sui veicoli. Nel caso di un biglietto elettronico, questo viene marcato come "utilizzato" all'interno del sistema. L'emissione di nuovi biglietti elettronici e abbonamenti viene amministrata dagli amministratori. Nel caso dell'utilizzo di un abbonamento, il sistema tiene traccia dell'ultimo utilizzo dello stesso.

Quando un conducente si trova ad un capolinea, può interrogare il sistema per sapere qual è la prossima partenza prevista del veicolo che sta guidando.

Glossario dei Termini

Realizzare un dizionario dei termini, compilando la tabella qui sotto, a partire dalle specifiche precedentemente disambiguate

Termine	Descrizione	Sinonimi	Collegamenti
Fermata	Indica un punto di sosta dell'autobus		Tratta
Capolinea	Indica una destinazione o un punto di partenza	Fermata inziale	Tratta

	di un atuobus		
Veicolo	Indica il mezzo pubblico utlizzato dai passeggeri per spostarsi	Autoveicolo	Tratta effettiva,Usato
Tratta	Indica un percorso astratto	Percorso	Tratta effettiva, Way Point, Fermata, Fermata intermedia, Capolinea iniziale, Capolinea finale
Tratta effettiva	Indica la tratta cocnreta che di un veicolo		Veicolo, Tratta, Conducente
Conducente	Indica il guidatore del veicolo	Autista	Tratta effettiva, Turno effettivo
Turno	Indica il turno astratto creato dall'amministratore		Turno effettivo
Turno effettivo	Indica il turno concreto del conducente		Conducente, Turno
WayPonit	È un punto geografico che indica una posizione		Tratta
Abbonamento	E' un biglietto esteso a tutte le tratte		
Biglietto	E' un biglietto valido per una singola tratta		Veicolo
Nuovo	Indica un biglietto non ancora timbrato		
Usato	Indica un timbretto convalidato		Veicolo
Titolo	Indica un qualsiasi tipo di elemento che permette di salire su un veicolo		

Raggruppamento dei requisiti in insiemi omogenei

Per ciascun elemento "più importante" della specifica (riportata anche nel glossario precedente), estrapolare dalla specifica disambiguata le frasi ad esso associate. Compilare una tabella separata per ciascun elemento individuato.

Fermata
Ciascuna tratta ha un insieme di fermate identificate da latitudine e longitudine ed associata ad un codice numerico univoco di cinque cifre. La prima fermata e l'ultima fermata sono contrassegnate come capolinea iniziale e capolinea finale. Ogni capolinea iniziale ha un orario di partenze prestabilito.
Tratta
L'azienda è dotata di un parco veicoli che permettono di coprire un determinato insieme di tratte Ciascuna tratta ha un insieme di fermate identificate da latitudine e longitudine ed associata ad un codice numerico univoco di cinque cifre. Inoltre, la tratta tra una fermata e l'altra è identificato da un insieme di waypoint, ciascuno caratterizzato da una latitudine ed una longitudine. Ciascuna tratta viene coperta da un numero predefinito di veicoli, la cui associazione viene gestita dagli amministratori
Capolinea inziale
La prima fermata e l'ultima fermata sono contrassegnate come capolinea iniziale e capolinea finale. Ogni capolinea ha un orario di partenze prestabilito.
Capolinea finale
La prima fermata e l'ultima fermata sono contrassegnate come capolinea iniziale e capolinea finale. Ogni capolinea ha un orario di partenze prestabilito.
Veicolo
I veicoli sono caratterizzati da una matricola (codice univoco numerico di quattro cifre). Ogni veicolo è anche associato ad una data di acquisto e ad uno storico di manutenzione. Ciascuna tratta viene coperta da un numero predefinito di veicoli, la cui associazione viene gestita dagli amministratori. Ogni veicolo è equipaggiato di un dispositivo GPS che, ogni 5 secondi, comunica le coordinate geografiche in cui si trova il veicolo.
Conducente
Gli amministratori gestiscono anche i conducenti, identificati da un codice fiscale, un nome, un cognome, una data di nascita ed un luogo di nascita. Di ogni conducente è di interesse conoscere anche il numero di patente e la data di scadenza della stessa. Gli amministratori devono poter gestire l'orario di lavoro dei conducenti, organizzati in turni di otto ore. Un conducente deve effettuare 5 turni a settimana. Qualora un conducente si ponga in malattia, gli amministratori devono poter indicare che il conducente non ha coperto il turno per malattia e identificare un nuovo conducente cui assegnare la sostituzione del turno.
Turno
Gli amministratori devono poter gestire l'orario di lavoro dei conducenti, organizzati in turni di otto ore. Un conducente deve effettuare 5 turni a settimana. La gestione dei turni avviene da parte degli amministratori su base mensile. Qualora un conducente si ponga in malattia, gli amministratori devono poter indicare che il conducente non ha coperto il turno per malattia e identificare un nuovo conducente cui assegnare la sostituzione del turno.

WayPoint
Inoltre, la tratta tra una fermata e l’altra è identificato da un insieme di waypoint, ciascuno caratterizzato da una latitudine ed una longitudine. La distanza deve essere calcolata andando a prendere in considerazione tutti i waypoint che intercorrono tra la posizione attuale veicolo e la fermata di interesse
Biglietto
Quando salgono a bordo, i passeggeri del servizio timbrano un biglietto elettronico o un abbonamento sul “validatore intelligente” installato sui veicoli. Nel caso di un biglietto elettronico, questo viene marcato come “utilizzato” all’interno del sistema. L’emissione di nuovi biglietti elettronici e abbonamenti viene amministrata dagli amministratori. Nel caso dell’utilizzo di un abbonamento, il sistema tiene traccia dell’ultimo utilizzo dello stesso.

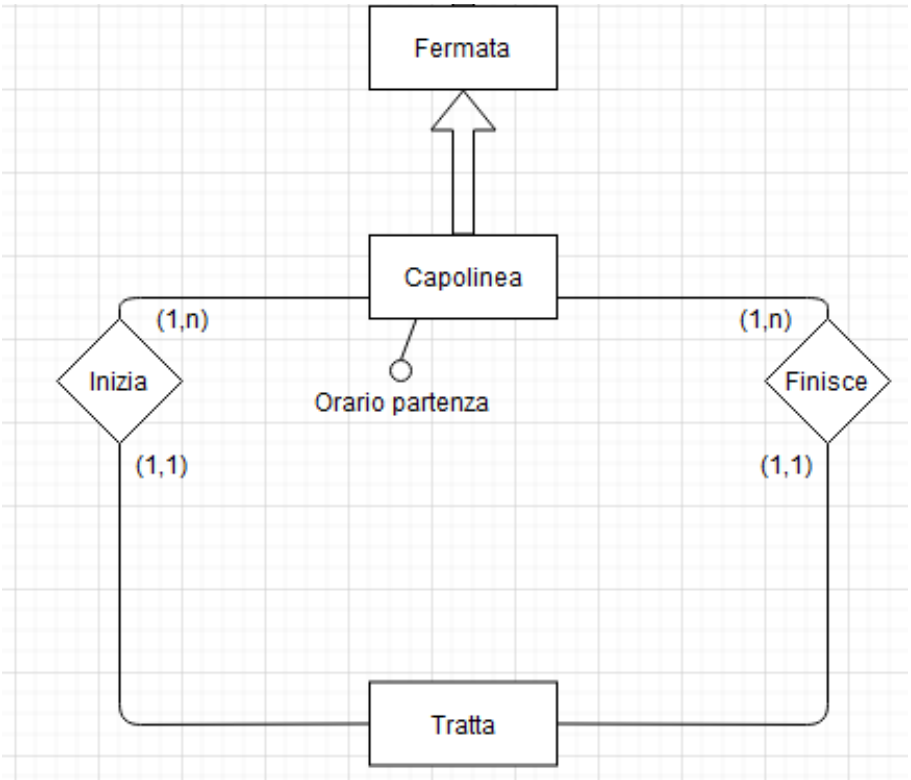
3. Progettazione concettuale

Costruzione dello schema E-R

Ho approcciato una tecnica mista per sviluppare il seguente modello:

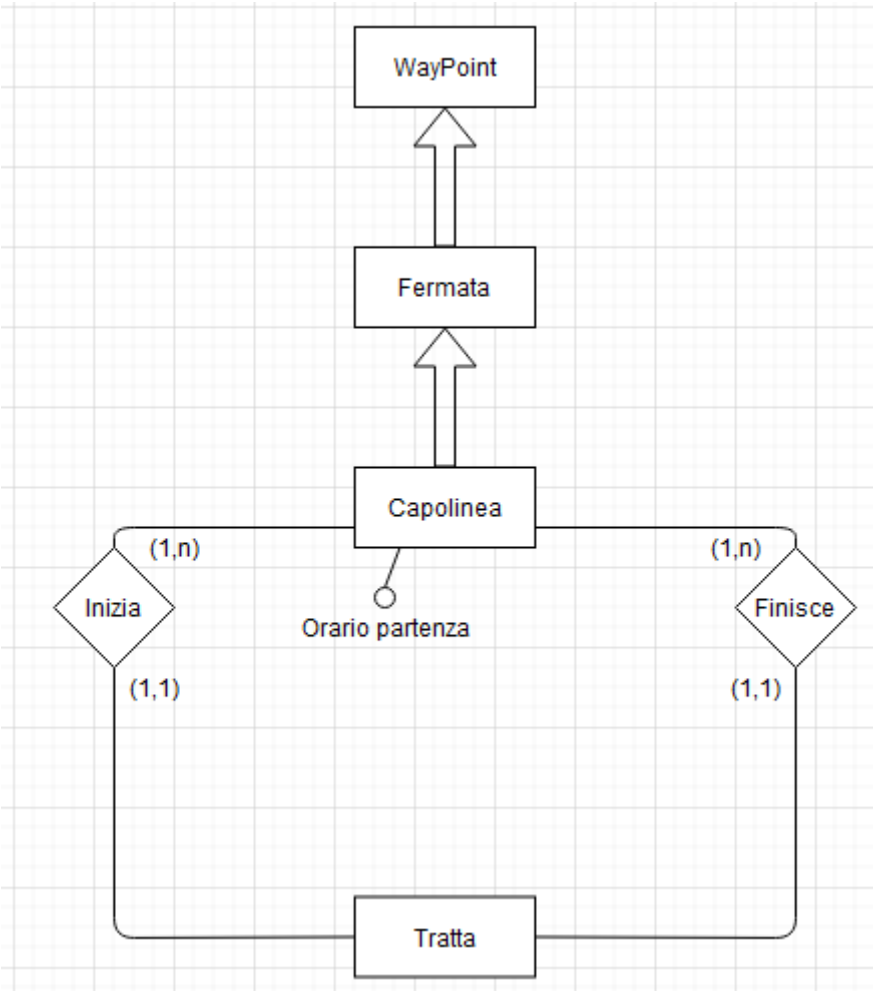
Nel testo si evince che nel sistema in questione e' essenziale tenere conto delle tratte che percorrono i veicoli, quindi ho deciso di creare l'entita (Tratta).

Successivamente nel testo viene scritto che ogni tratta e' caratterizzata da, una fermata di partenza e di arrivo prestabilita , e da varie fermate intermedie presenti nel corso del tragitto, quindi avro sicuramente la necissita di introdurre le entita (Fermata) e (Capolina) , quest'ultima da come si evince nel testo, ha come attributo l'orario di partenza, ed è una generalizzazione della fermata, in quanto il capolinea e' un tipo di fermata.



Andando avanti con la lettura si nota che nelle tratte sono presenti dei waypoint utili a capire a che distanza si trova il mezzo dalla prossima fermata; per rappresentare questo concetto si crea l'entita (Waypoint) collegandola come padre dell'entita fermata, in quanto la fermata puo' essere un waypoint.

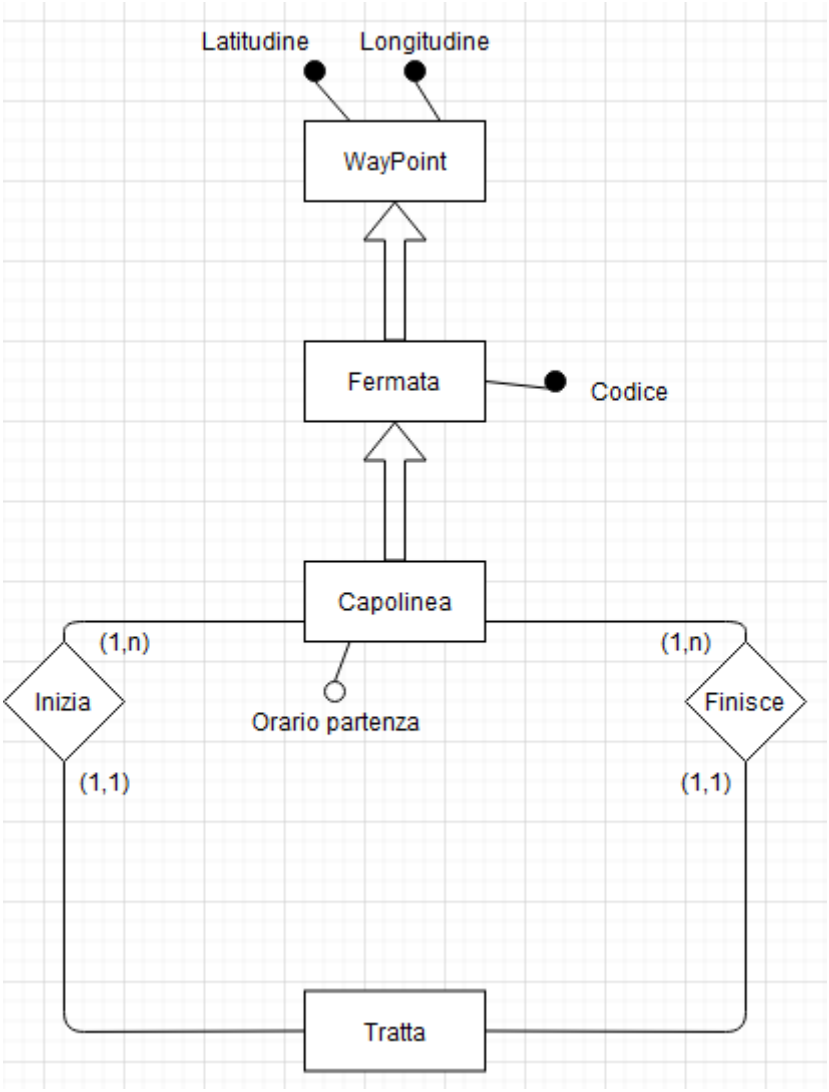
Successivamente ho utilizzato le relazioni (inizia) e (finisce) per indicare la fermata di partenza e di arrivo della tratta, utilizzando in entrambe, delle relazioni zero ad enne , andando ad indicare che per una singola tratta e` presente una sola fermata di partenza e una sola fermata di arrivo, e che quest'ultime possono essere corrispettivamente la fermata inziale e finale di altre tratte.



Entrando maggiormente nel dettaglio si evince dal testo che ogni tratta e' caratterizzata da un codice univoco, il quale diventerà un suo attributo, in particolare un attributo primario, perché diverso ed univoco per ogni tratta.

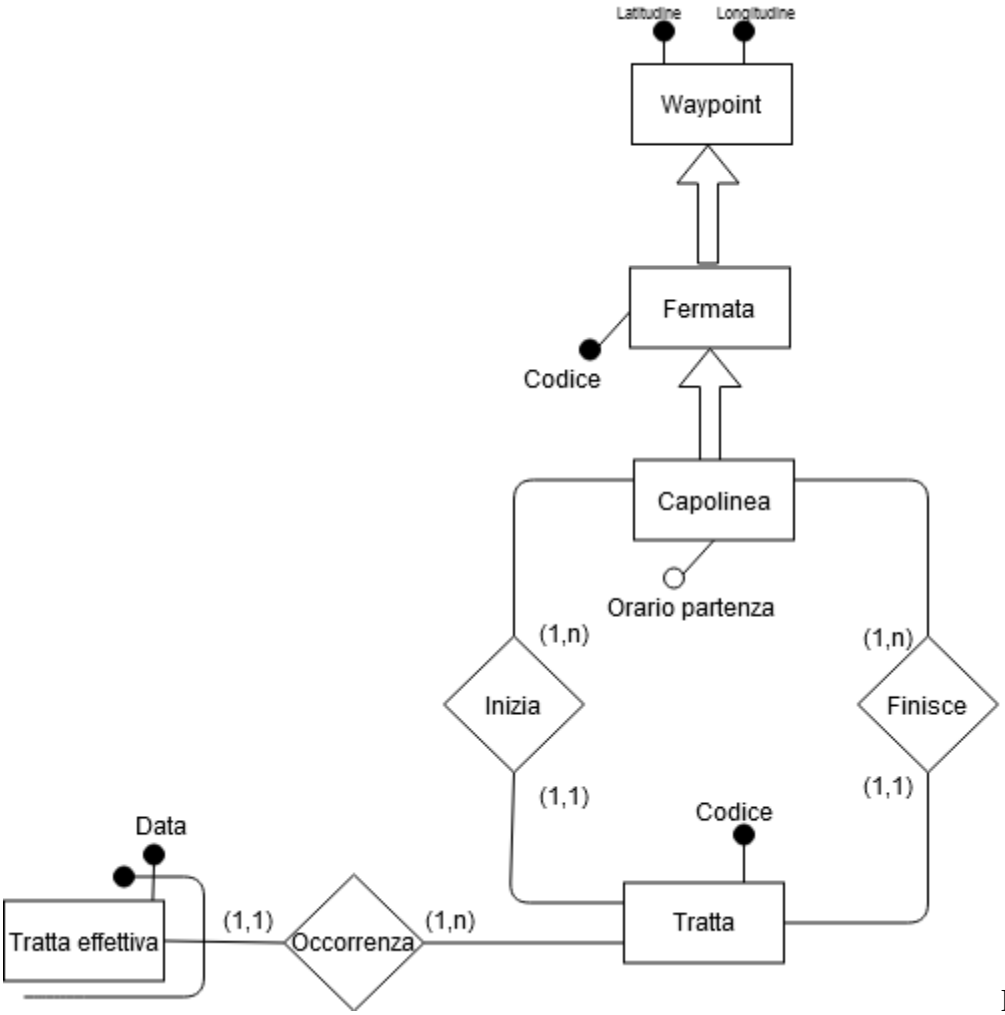
Successivamente visto che i waypoint,da come evince dal testo,sono rappresentati da una latitudine e longitudine,introduco due attributi inerenti a questi dati, i quali in coppia formeranno la sua chiave primara; mediante la generalizzazione anche le entità fermata e capolinea erediteranno tali attribbuti.

Inoltre sempre dal testo si evince che ogni fermata è identificata da un codice numerico, tale codice diventerà un suo attributo principale.



Il passo successivo è stato quello di aggiungere un attributo all'entità (Tratta) chiamato Codice, utile a distinguere le varie tratte tra loro; successivamente viene diviso il concetto di tratta reale con il suo concetto astratto, andando ad inserire l'entità (tratta effettiva). Noto che l'entità (tratta effettiva) è un'entità che non può vivere autonomamente, ma dipende dall'esistenza della tratta; da questa presupposizione decido di rendere tratta effettiva un entita debole avente come attributi primari : la data e la tratta ereditata dalla proprietà dell'entità debole.

Utilizzo l'associazone (occorrenza) per indicare che in quella tratta astratta possono esserci piu tratte concrete, e che una tratta concreta ovviamente percorrerà una sola tratta.

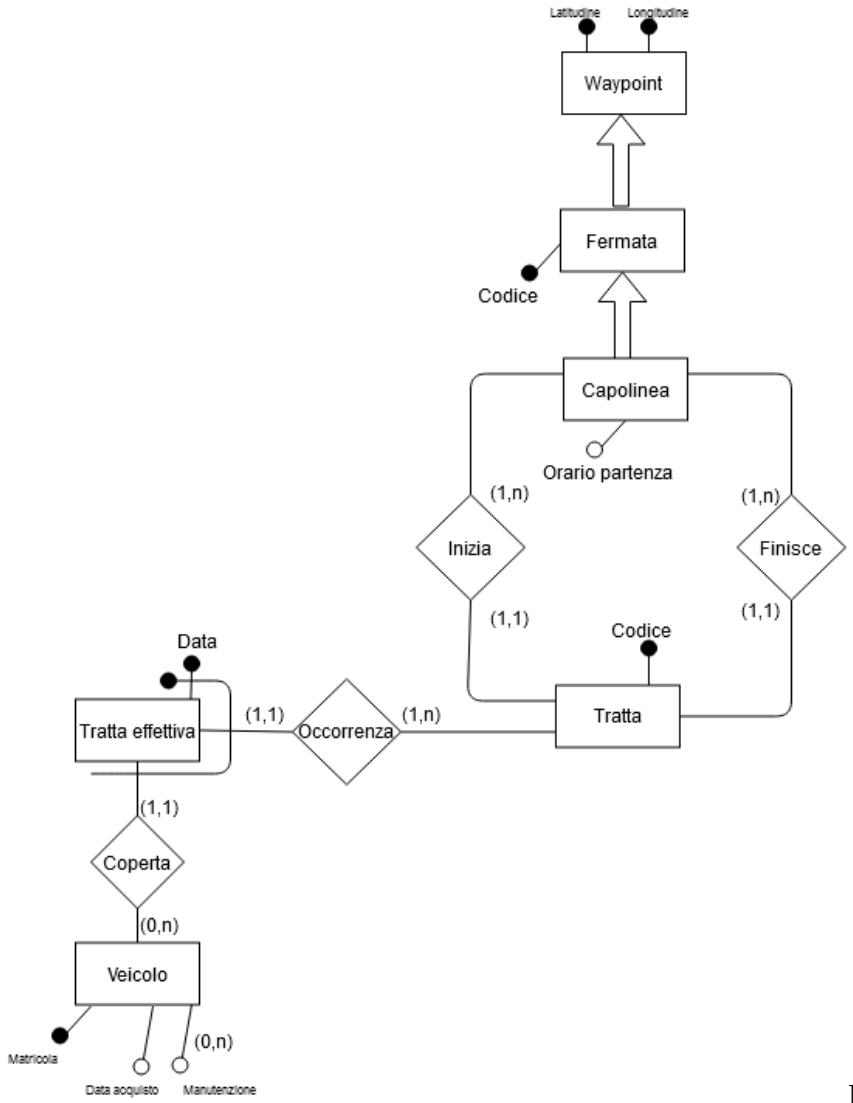


Dal testo ottengo l'informazione che ogni tratta è coperta da un certo numero di veicoli, quindi viene inserita l'entità veicolo caratterizzata da una data di acquisto, da uno storico di manutenzioni e da una matricola, la quale sarà la sua primarykey in quanto è univoca per ogni veicolo.

L'attributo manutenzioni, oltre ad esserre un attributo (0,n), in qaunto possono essere fatte più manutenzioni,viene messo come opzionale, in quanto un veicolo potrebbe non aver avuto mai una manutenzione.

Quindi unisco le due entità mediante l'associazione coperta, la quale indica che un veicolo può coprire piu tratte effettive in diversi giorni e orari, mentre la tratta effetettiva è coperta da un solo veicolo in quel determinato giorno e orario.

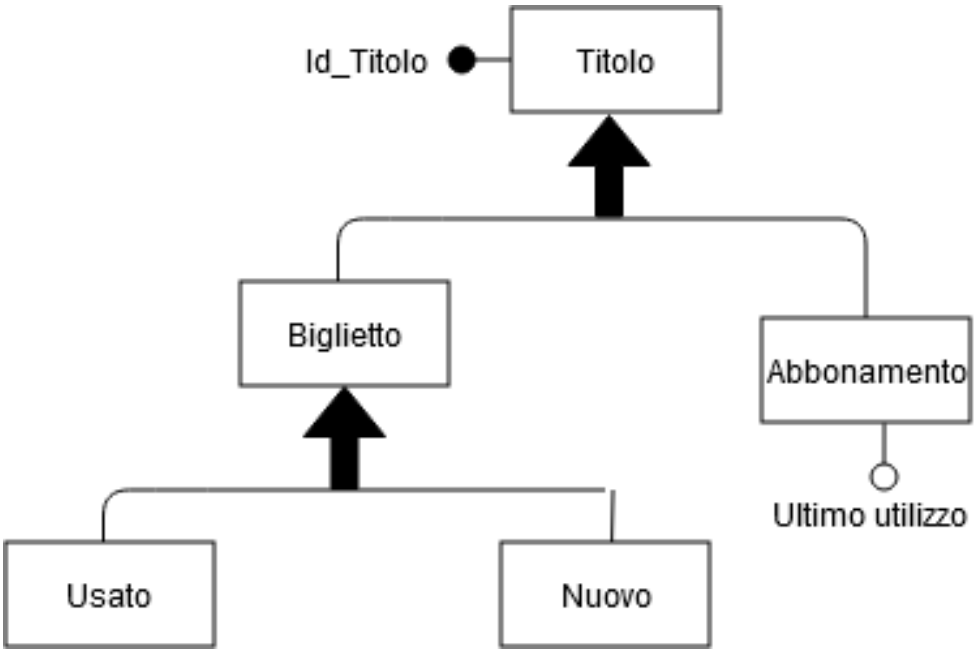
Ovviamente l'entità tratta effettiva dipenderà anche dalla tratta veicolo in quanto, senza di esso, la tratta effettiva non può esistere.



Nel testo viene definito il concetto di biglietto andando ad indicare che su ogni veicolo è installato un convalidatore intelligente che avrà il compito di convalidare gli opportuni biglietti.

Da questa presupposizione viene inserita l'entità (Titolo), ma da come si evince nel testo il titolo può essere di due tipi: (Abbonamento) del quale ci interessa l'ultimo utilizzo e (Biglietto elettronico), quest'ultimo può essere generalizzato in (Usato) se timbrato, oppure (Nuovo).

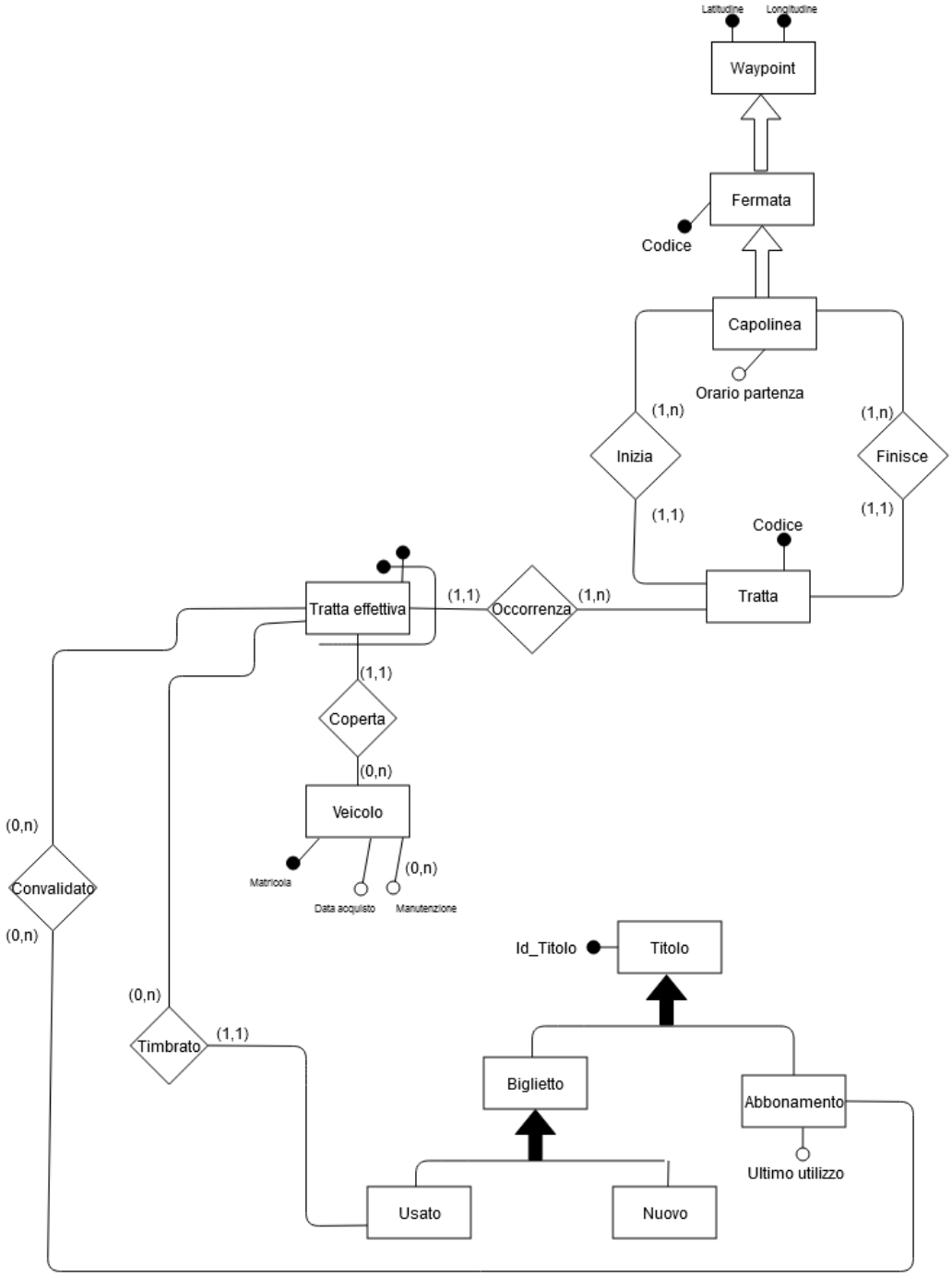
Quindi si inserisce nel diagramma tutta questa generalizzazione dei titoli di viaggio.



Possiamo finalmente collegare il biglietto utilizzato con l'entità tratta effettiva mediante l'associazione (Timbrato) in quanto un biglietto viene timbrato su una tratta effettiva e non sul singolo veicolo, in modo tale che se tale veicolo nella stessa giornata dovesse fare più tratte, tale biglietto è valido solo per una determinata tratta.

Tale condsireazione viene fatta anche verso l'entità Abbonamento, in quanto un abbonamento viene convalidato su un veicolo per una determiata tratta, e non per tutte le tratte che effettuerà tale veicolo.

In fine viene aggiunto l’attributo primario Id_Titolo utile per distinguere i vari titoli tra loro



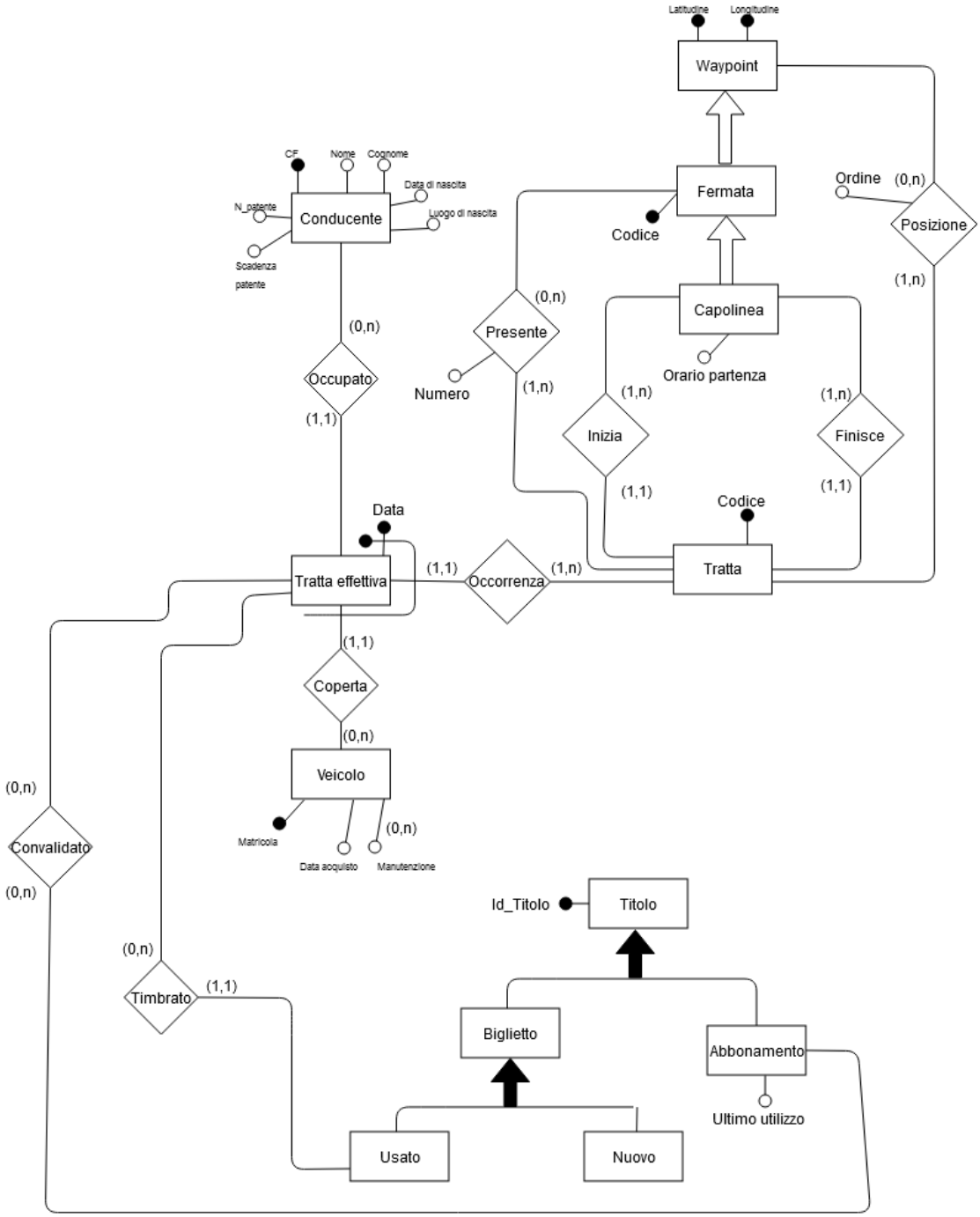
Da come si evince dal testo, bisogna tenere contro dei waypoint presente nelle tratte; Ovviamente un waypoint avrà un codice numero diverso a seconda della tratta, per raccolgiere l’informazione inerente ad un waypoint su una specifica tratta, viene introdotta l’associazione (Posizione) che collega le entità waypoint e tratta, andando a specificare che in una tratta ci possono essere più waypoint, e che quel waypoint può essere inerente a più tratte.

Oltre a questa informazione, per raccogliere i dati delle fermate presenti in una specifica tratta, viene inserita la relazione (Presente) che permette di capire quale fermate sono presenti in una tratta e il

loro corrispetivo ordine, specificando che in una tratta ovviamente possono essere da 1 a n fermate, mentre invece una fermata può appartenere a 0 o n tratte; viene inserito lo zero nel caso di fermate inutilizzate

A questo punto ci concentriamo nella definizione di (Conducente), perché da come si evince nel testo, ogni conducente guida un veicolo su una tratta; per rappresentare quet'ultimo concetto viene creata un entità chaimata (Conducente) rappresentante il conduecente, con i rispettivi attributi ricavati dal testo, tra cui nome,cognome,data di nascita,luogo di nascita,patente.scadenza patente e codice fiscale, quest'ultimo viene scelto come primarykey in quanto univoco per ogni conducente.

Viene collegato il conducente alla tratta effettiva mediante l'associazione (occupato) in quanto un conducente può occupare nel corso della sua cariera zero tratte effettive, se è un nuovo conducente, o enne tratte effettive, mentre una tratta effettiva ovviamente può essere occupata da un solo conducente.

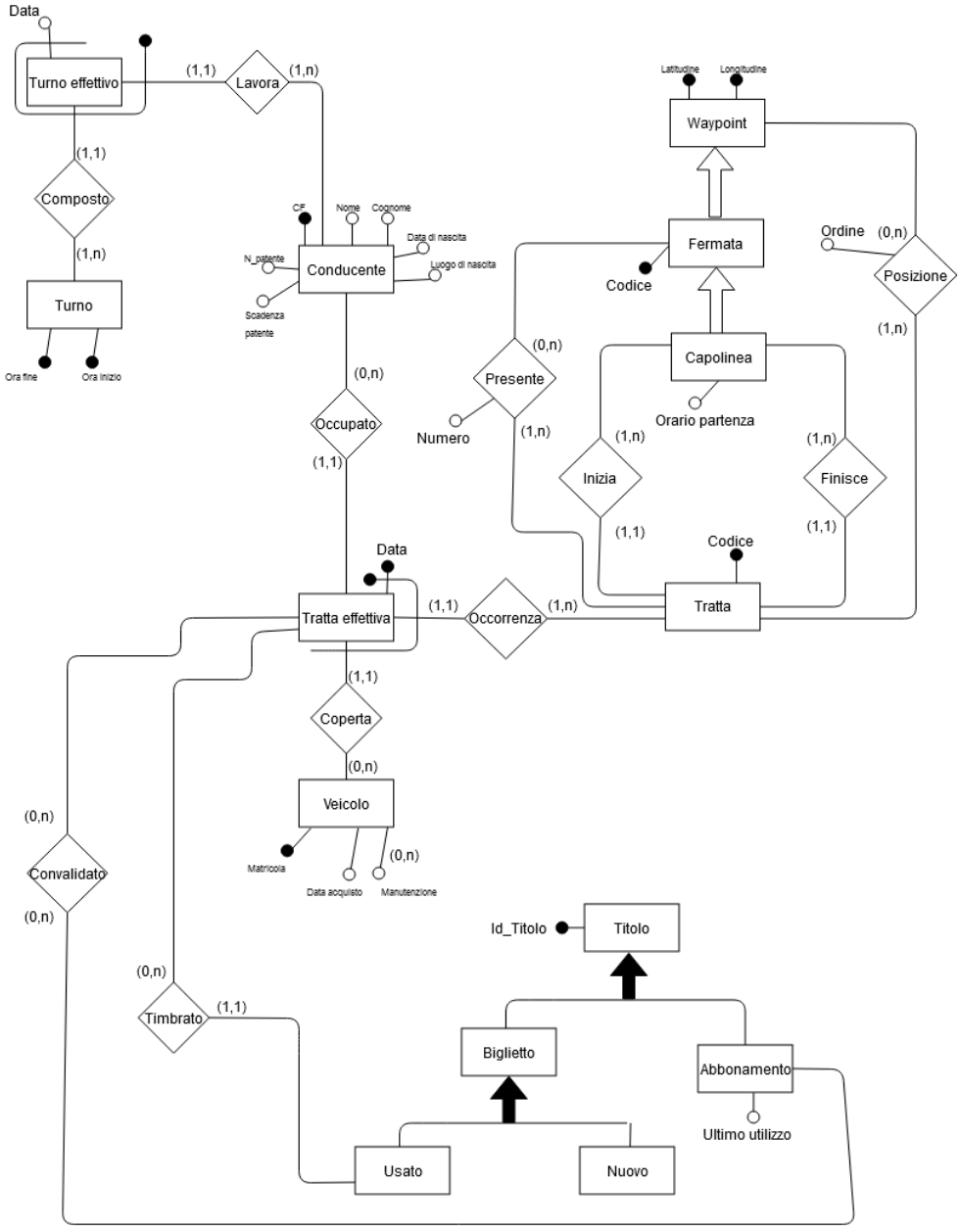


In fine da come si evince nel testo ogni conducente effettua dei turni, quindi viene creata l'entità (Turno) la quale indica il concetto astratto di turno, e l'entità (Turno effettivo) la quala indica il concetto concreto di turno.

Il conducente viene collegato mediante l'associazione (lavora) all'entità turno effettivo andando ad indicare che un conducente può effettuare da zero, in caso di new entry, fino ad enne turni effettivi, mentre il turno effettivo può naturalmente riferisci ad un solo conducente.

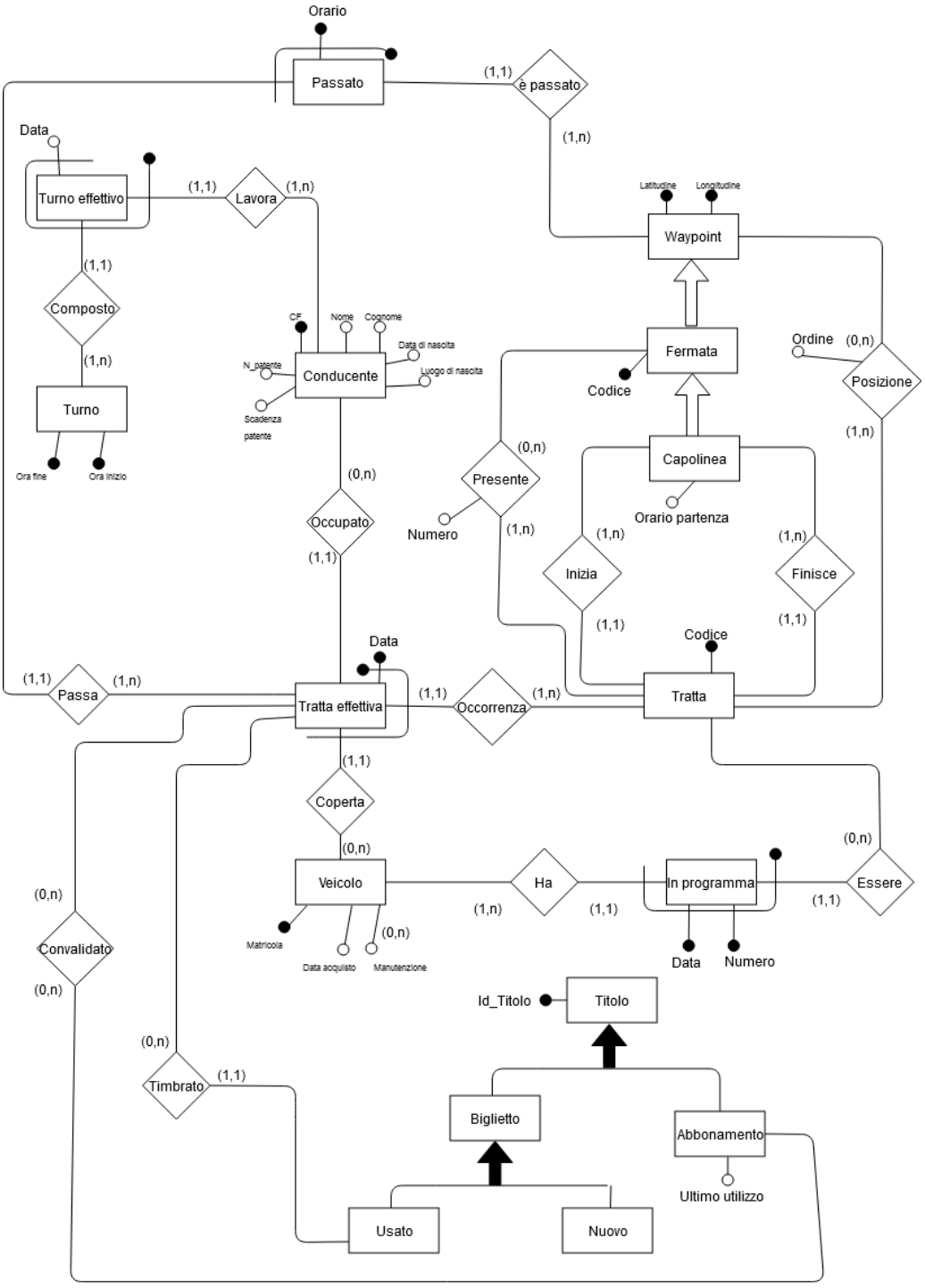
L’entità turno invece viene collegata con l’entità turno effettivo mediante l’associazione composto andando ad indicare che ovviamente in un turno astratto possono esserci effettivamente più turni concreti, mentre invece un turno conreto è riferito ad un unico turno astrattto.

Come l’entità tratta effettiva ovviamente dipendeva dall’esistenza di varie entità come conducente, veicolo, e tratta in quanto senza uno di essi quella tratta effettiva non può esistere, anche l’entità turno effettivo ovviamente dipende dall’esistenza dell’entità tratta e l’entità conducente, diventando quindi un’entità debole dipendente da quest’ultime

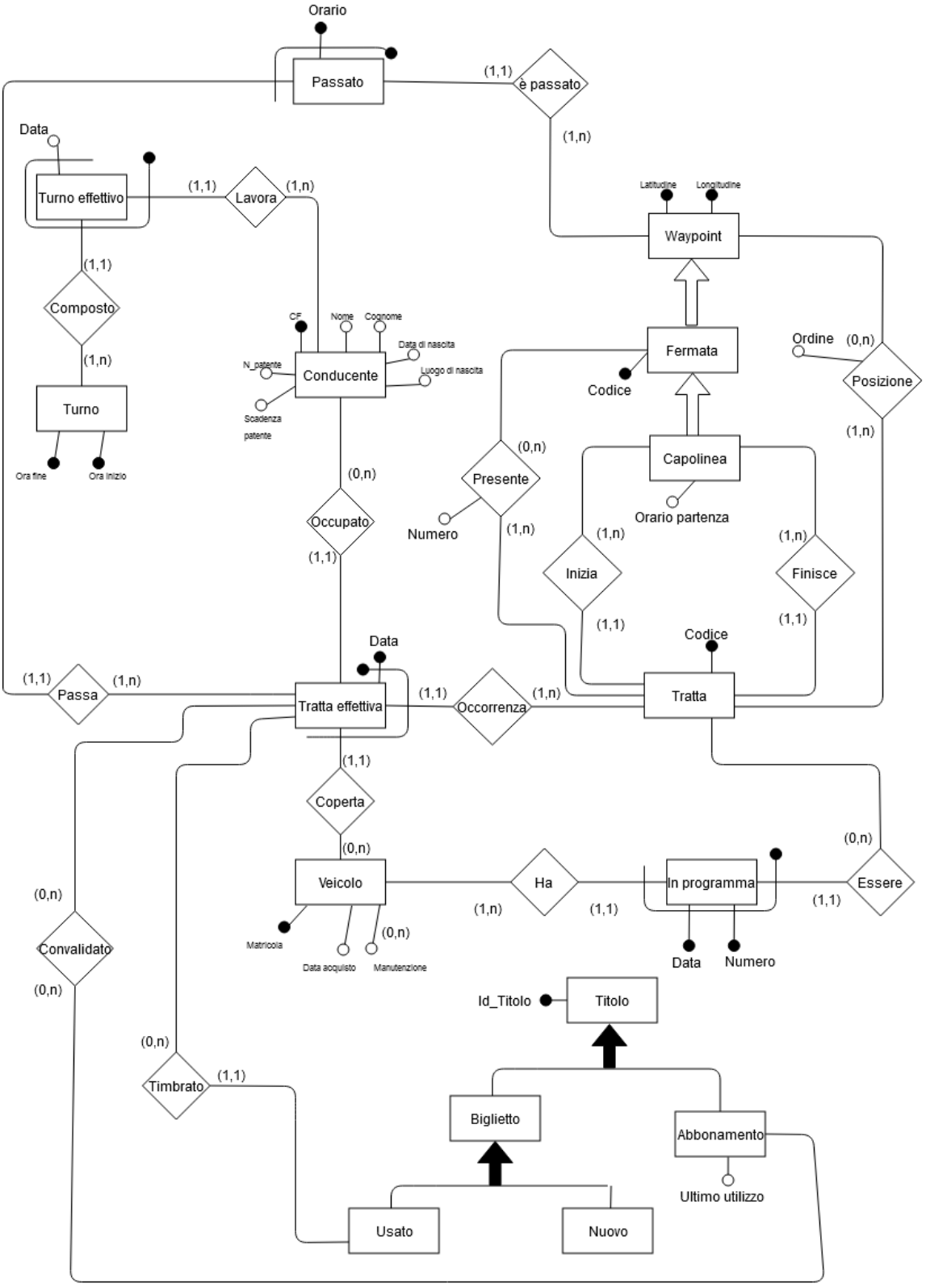


In aggiunta per tenere traccia di quale waypoint sono presenti nella mia tratta, creo una relazione (Passato) che va ad indicare quale waypoint ho appena passato in una determinata tratta, al fine di poter calcolare successivamente la distanza effettiva del veicolo da una fermata.

Come ultima accortezza, visto che il conducente può interrogare il veicolo per vedere quale sarà la prossima tratta che tale veicolo dovrà fare, viene creata la relazione (in programma) che va ad indicare quali tratte dovrà fare un determinato veicolo in una giornata.



Integrazione finale



Viene aggiunto l’attributo primario Id_Titolo per distinguere i vari titoli tra di loro.

Regole aziendali

Di vincolo:

1. Il GPS deve cominicare ogni 5 secondi la sua posizone
2. Ogni conducente deve effettuare 5 turni a settimana
3. Ogni turno deve essere di 8 ore

Di derivazione:

1. La distanza tra due coordinate geografiche si calcola mediate la formula sopra indicata
2. La tratta successiva di un conducente e' ottenuta andando a vedere la prossima tratta relativa al veicolo che sta guidando
3. La distanza di un veicolo dalla fermata si calcola andando a prendere in considerazione l'ultimo waypoint passato

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Fermata	Indica le fermate interne alla tratta	Latitudine,Longitudine,Codice	Latitudine,Longitudine,Codice
Capolinea	Indica la prima/ultima fermata di una tratta	Latitudine,Longitudine,Orario partenze, Codice	Latitudine,Longitudine, Codice
Tratta	Indica il concetto astratto di tratta	Codice	Codice
Waypoint	Indica i punti geografici presenti nelle tratte	Latitudine,Longitudine	Latitudine,Longitudine
Tratta effettiva	Indica la tratta concreta	Codice,CF,Matricola,Data	Codice,Matricola,Data
Veicolo	Indica il mezzo di trasporto	Matricola,Data acquisto,Manutenzione	Matricola
Conducente	Indica la persona che guida il mezzo	CF,Nome,Cognome, Data di nascita, Luogo di nascita,N_patente,Scadenza patente	CF
Turno	Indica il concetto astrattoi di turno	Ora fine,Ora inizio	Ora fine,Ora inizio
Turno effettivo	Indica il concetto concreto di turno	Ora fine,Ora inizio,CF ,Data	Ora fine,Ora inizio,CF, Data

Titolo	Indica la generalizzazione di tutti i titoli di viaggio	Id_Titolo	Id_Titolo
Abbonamento	Indica un tipi di titolo riutilizzabile più volte	Id_Titolo, Ultimo utilizzo	Id_Titolo
Biglietto	Indica un tipo dio titolo utilizzabile una sola volta	Id_Titolo	Id_Titolo
Nuovo	Indica un biglietto non utilizzato	Id_Titolo	Id_Titolo
Usato	Indica un biglietto timbrato	Id_Titolo	Id_Titolo
In programma	Indica le tratte in programma per quel veicolo	Data,Numero,Matricola,Codice	Data,Matricola,Codice
Convalidato	Indica il veicolo sulla quale è stato convalidato l’abbonamento	Matricola,Data,Id_Titolo	Matricola,Data,Id_Titolo
Passato	Indica i WayPoint passato in una tratta	Orario,Latitudine.Longitudine,Matricola,CF,Data,Codice	Orario Matricola,,Data,Codice

4. Progettazione logica

Volume dei dati

Concetto nello schema	Tipo[1]	Volume atteso
Waypoint	E	23.200
Fermata	E	5800
Capolinea	E	200
Tratta	E	260
Tratta effettiva	E	300(85.7%)
Veicolo	E	350
Usato	E	208.000(40%)
Nuovo	E	312.000(60%)
Biglietto	E	520.000(40%)
Abbonamento	E	780.000(60%)
Titolo	E	13.000.000
Conducente	E	11.000
Turno effettivo	E	300(2.8%)
Turno	E	24
Inizia	R	260
Finisce	R	260
Composto	R	300
Posizione	R	23.200 (Per ogni tratta ci sono circa 88 waypoint)
Presente	R	5.800(Per ogni tratta ci sono circa 22 fermate)
Passato	E	46.400(Ongi waypoint è passato in media da due tratte)
Convalidato	E	780.000 (Piu o meno tutti gli abbonamenti hanno una data di convalidazione
In programma	E	1050(Per ogni veicolo ci sono in media

1 Indicare con E le entità, con R le relazioni

		tre tratte al giorno)

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
1 A	**Assegna ad un veicolo una tratta**	**300 al giorno**
2 A	**Visualizza i veicoli in attività**	**300 al giorno** *(Ogni volta che eseguo l'operazione 1)*
3A	**Visualizza i veicoli dentro al deposito**	**300 al giorno** *(Ogni volta che eseguo l'operazione 1)*
4 A	**Assumi un conducente**	**1 al mese**
5 A	**Licenzia un conducente**	**1 ogni anno**
6 A	**Assegna ad un conducente un turno**	**11.000 mese** *(Una volta al mese per ogni conducente)*
7 A	**Cambia conducente in quel turno**	**10 al giorno** *(Ogni volta che un conducente si ammala)*
8 A	**Visualizza i conducenti che non stanno lavorando**	**10 al giorno** *(Ogni volta che un conducente si ammala)*
9 A	**Visualizza i conducenti che stanno lavorando**	**10 al giorno** *(Ogni volta che un conducente si ammala)*
10 P	**Calcola la distanza di un veicolo da una fermata**	**10.000.000 al giorno** *(In media su 13 m. di passeggeri, il 60% sta viaggiando, di questo 60%, il 70% controlla almeno una volta la distanza del veicolo dalla fermata, in media 2 accessi al giorno per persona)*
11 P	**Timbra un biglietto**	**400.000 al giorno** *(In media su 520 mila biglietti l'80% di essi viene timbrato il giorno stesso)*
12 P	**Convalida un abbonamento**	**500.000 al giorno** *(In media su 780 milaabbonamenti il 70% di essi viene convalidato nell'arco della giornata)*
13 A	**Crea un nuovo biglietto**	**700 al giorno**
14 A	**Crea un nuovo abbonamento**	**400 al giorno**
15 C	**Vedi la prossima tratta del veicolo**	**480** *(In media su 300 conducenti che stanno lavorando l'80% esegue l'operazione, in media 2 volta al giorno per ogni conducente)*

Costo delle operazioni

Operazione 1 costo 5			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Veicolo	E	1	L
Tratta	E	1	L
Conducente	E	1	L
Tratta effettiva	E	1	S
Operazione 2 costo 520			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Tratta	E	260	L
Tratta effettiva	E	260	L
Operazione 3 costo 520			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Tratta	E	260	L
Tratta effettiva	E	260	L
Operazione 4 costo 2			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Conducente	E	1	S
Operazione 5 costo 2			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Conducente	E	1	S
Operazione 6 costo 4			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Conducente	E	1	L
Turno	E	1	L
Turno effettivo	E	1	S
Operazione 7 costo 7			

Concetto	Costrutto	Accessi	Tipo
Conducente	E	2	L
Turno effettivo	E	2	S
Turno	E	1	L

Operazione 8 costo 46

Concetto	Costrutto	Accessi	Tipo
Turno	E	23	L
Turno effettivo	E	23	L

Operazione 9 costo 46

Concetto	Costrutto	Accessi	Tipo
Turno	E	23	L
Turno effettivo	E	23	L

Operazione 10 costo 783

Concetto	Costrutto	Accessi	Tipo
Fermata	**E**	**1**	**L**
Presente	R	1	L
Tratta	E	260 (Al peggio tutte le tratte passano per quella fermata)	L
Tratta effettiva	E	260 (Al peggio tutte le tratte passano per quella fermata)	L
Passato	R	260 (Al peggio tutte le tratte passano per quella fermata)	L
WayPoint	E	1	L

Operazione 11 costo 3

Concetto	Costrutto	Accessi	Tipo
Biglietto	E	1	L
Usato	E	1	S

Operazione 12 costo 3

Concetto	Costrutto	Accessi	Tipo

Abbonamento	E	1	L
Convalidato	E	1	S

Operazione 13 costo 4			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Titolo	E	1	S
Biglietto	E	1	S

Operazione 14 costo 4			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Titolo	E	1	S
Abbonamento	E	1	S

Operazione 15 costo 2			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Veicolo	E	**1**	**L**
In programma	**E**	**1**	**L**

1. Costo 5, Al giorno 300 — Costi al giorno 1.500
2. Costo 520, Al giorno 300 — Costi al giorno 156.000
3. Costo 520, Al giorno 300 — Costi al giorno 156.000
4. Costo 2, Al mese 1 — Costi al mese 2
5. Costo 2, All’anno 1 — Costi all’anno 2
6. Costo 4, Al mese 11.000 — Costi al mese 44.000
7. Costo 7, Al giorno 10 — Costi al giorno 70
8. Costo 46, Al giorno 10 — Costi al giorno 460
9. Costo 46, Al giorno 10. — Costi al giorno 460
10. Costo 783, Al giorno 10.000.000 — Costi al giorno 7.830.000.000
11. Costo 3, Al giorno 400.000 — Costi al giorno 1.200.000
12. Costo 3, Al giorno 500.000 — Costi al giorno 1.500.000
13. Costo 4, Al giorno 700 — Costi al giorno 2.800
14. Costo 4, Al giorno 400 — Costi al giotno 1.600
15. Costo 2, Al giorno 480 — Costi al giorno 960

Ristrutturazione dello schema E-R

Andando a ristrutturare il modello E/R si decide di sciogliere le due generalizzazioni (WayPoint) e (Titolo);

La prima ristrutturazione consiste nel ricompattare le entità (Fermata), (Capolinea) e (WayPoint) in un'unica entià (WayPoint) la quale viene collegata a (tratta) per mezzo di quattro relazioni:

1) (Presente) con il relativo numero del waypoint, che rappresenta la posizione di quel waypoint sulla tratta
2) (Fermata) con il relativo numero di fermata che rappresenta il numero della fermata in quella tratta.
3) Inoltre l'entità (WayPoint) continuerà ad avere gli attributi identificatori latitudine e longitudine, in quanto ogni fermata ed ogni capolinea oltre ad essere identificati dal codice, sono identificate anche da latitudine e longitudine, in piu viene aggiunto l'attributo cod_fermata e orario partenza i quali mi serviranno per identificare se tale waypoint rappresenta una fermata, un capolinea o un semplice waypoint

La seconda ristrutturazione consiste nel diversificare i concetti di (Biglietto) ed (Abbonamento) andando ad inserire gli attributi del padre nei figli, in quanto, essendo una generalizzazione totale e visto che il padre da solo non era in relazione con nessuna entità si è deciso di;

1) Assegnare un identificatore all'entità (Biglietto) chiamato (Id_Biglietto) ed un attributo (Stato_Biglietto) che va ad identificare se tale biglietto è stato timbrato o no.
2) Assegnare un identificatore all'entità (Abbonamento) chiamato (Id_Abbonamento) ad un attributo (Ultimo utilizzo) richiesto dal sistema; ovviamente ogni abbonamento potrà essere convalidato più volte su uno stesso veicolo ma ovviamente in giornate diverse, per permettere tale operazione viene inserito un attributo primario (Data) nella relaizone tra abbonamento e veicolo.

Sciogliendo queste generalizzazioni alcuni costi delle operazioni migliorano:

- L'operazione 2 passa da un costo di 520 ad un costo di 350 , diminuendo il suo consumo del 32,7%
- L'operazione 3 passa da un costo di 520 ad un costo di 350 , diminuendo il suo consumo del 32,7%
- L'operazione 10 passa da un costo di 1024 ad un costo di 521 , diminuendo il suo consumo del 50%
- L'operazione 13 passa da un costo di 4 ad un costo di 2 , diminuendo il suo consumo del 50%
- L'operazione 14 passa da un costo di 4 ad un costo di 2 , diminuendo il suo consumo del 50%

In fine viene sciolto l'attributo multiplo presente in (Veicolo) chiamato "Manutenzione" , creando un apposita entita

Trasformazione di attributi e identificatori

Nelle due entità deboli vengono applicate le seguenti modifiche:

1) Nell'entità (Tunro effettivo) gli attributi ereditati dal tunro rimangono tali: "Ora_inzio", "Ora_fine"; mentre l'attributo CF ereditato dall'entità (Conducente) viene rinominato "Conducente" al fine di migliroare la comprensione.
2) Nell'entità (Tratta effettiva) l'attributo CF ereditato da (conducente) viene rinominato "Conducente", l'attributo Matricoloa ereditato da (Veicolo) prende il nome di "Veicolo", ed in fine l'attributo Id_Tratta ereditato da (Tratta) viene rinominato "Tratta".

Nelle relazioni:

- Nella relazione (timbrato) l'Id_biglietto viene rinominato "Biglietto" e la matricola del veicolo viene rinominata "Veicolo"

- Nella relazione (Convalidato) l'Id_Abbonamento viene rinominato " Abbonamento" e la matricola del veicolo viene rinomata "Veicolo"
- Nella relazione (Effettua) la matricola del veicolo viene rinomata "Veicolo".
- Nella relazione (In programma) la matricola del veicolo viene rinomata "Veicolo" e l'Id_Tratta viene rinominato semplicemente "Tratta"
- Nella relazione (Capolinea) l'Id_Tratta viene rinominato semplicemente "Tratta"
- Nella relazione (Fermata) l'Id_Tratta viene rinominato semplicemente "Tratta"

Traduzione di entità e associazioni

Conducente (CF, Nome, Cognome, Data_nascita, Luogo_nascita, Scadenza_patente, N_patente)

Veicolo (Matricola, Data_acquisto)

Biglietto (Id_Biglietto, Stato_biglietto, Veicolo*,Tratta*,Data*)

Abbonamento (Id_Abbonamento, Ultimo_utilizzo*)

Manutenzione (Tipo)

Tratta (Id_Tratta, Latitudine_I, Lognitudine_I, Latitudine_F, Longitudine_F)

WayPoint (Latitudine, Longitudine, Cod_capolinea*, Cod_Fermata*, Orario_partenze*)

Turno (Ora_Inizio, Ora_fine)

Turno effettivo (Data, Ora_inzio, Ora_fine, Conducente)

Tratta effettiva (Data, Veicolo, Tratta, Conducente)

Passato (Data, Veicolo, Tratta, Orario, Latitudine, Longitudine)

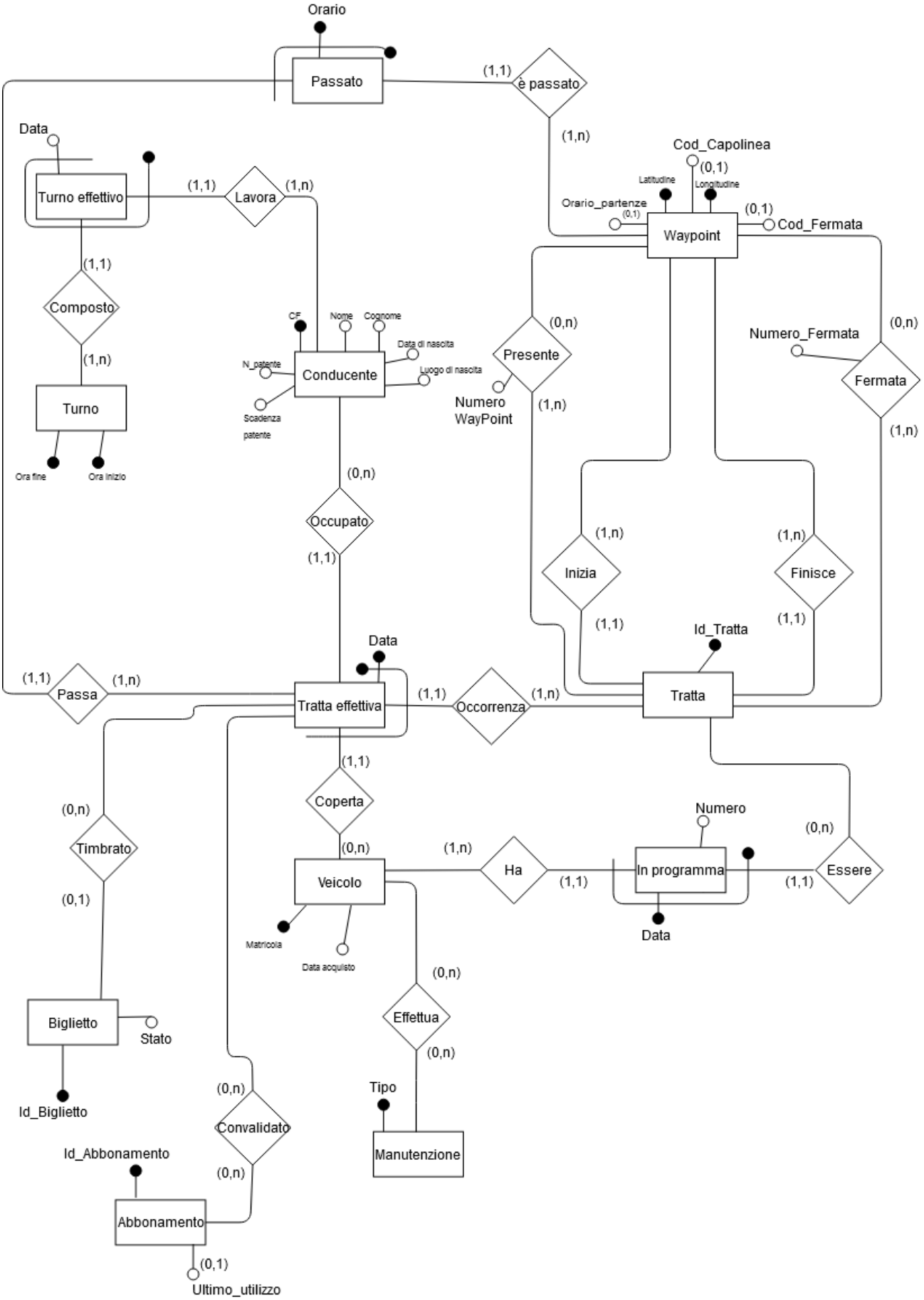
In programma (Veicolo, Tratta, Numero, Data)

Convalidato (Abbonamento, Veicolo, Tratta, Data)

Effettua (Manutenzione, Veicolo)

Presente (Tratta, Latitudine, Longitudine, Numero_Waypoint)

Fermata (Numero_fermata, Latitudine, Longitudine, Tratta)


Orario
Passato
(1,1)
è passato
(1,n)
Data
Turno effettivo
(1,1)
Lavora
(1,n)
Cod_Capolinea
(0,1)
Latitudine
Longitudine
Orario_partenze
(0,1)
(0,1)
Waypoint
Cod_Fermata
(1,1)
Composto
(1,n)
Turno
Ora fine
Ora inizio
CF
Nome
Cognome
Data di nascita
N_patente
Luogo di nascita
Conducente
Scadenza patente
(0,n)
Presente
Numero WayPoint
(1,n)
Numero_Fermata
(0,n)
Fermata
(1,n)
(0,n)
Occupato
(1,1)
(1,n)
Inizia
(1,1)
(1,n)
Finisce
(1,1)
Id_Tratta
Tratta
Data
(1,1)
Passa
(1,n)
Tratta effettiva
(1,1)
Occorrenza
(1,n)
(1,1)
Coperta
(0,n)
Timbrato
(0,1)
Biglietto
Stato
Id_Biglietto
(0,n)
Convalidato
Id_Abbonamento
(0,n)
Abbonamento
(0,1)
Ultimo_utilizzo
(0,n)
Veicolo
Matricola
Data acquisto
(1,n)
Ha
(1,1)
Numero
In programma
Data
(0,n)
Essere
(1,1)
(0,n)
Effettua
(0,n)
Tipo
Manutenzione

Normalizzazione del modello relazionale

Le tabelle sono tutte in 3NF

5. Progettazione fisica

Utenti e privilegi

Utenti:

1) Amministratore Amministratore
2) Conducente Conducente
3) Passeggero Passeggero

Ruoli:

1) Amministratore
2) Conducente
3) Passeggero

Privilegi:

1) Amministratore:
   L'amministratore ha accesso alle tabelle a tutte le tabelle, in particolare può:
   - Assegnare turni ai conducenti (Si evince dal testo)
   - Assegnare veicoli alle tratte (Si evince dal testo)
   - Assumere conducenti
   - Licenziare conducenti
   - Visualizzare i conducenti attivi (Utile per assegnare nuovi turni e per sostituire i conducenti in malattia)
   - Visualizzare i conducenti fermi (Utile per assegnare nuovi turni e per sostituire i conducenti in malattia)
   - Visualizzare i veicoli attivi (Utile per assegnare veicoli alle tratte)
   - Visualizzare i veicoli fermi (Utile per assegnare veicoli alle tratte)

2) Conducente:
   Il conducente ha l'accesso ad un numero ristretto di tabelle in quanto può interrogare il sistema solo per sapere quale sarà la prossima tratta del veicolo che sta guidando
3) Passeggro:
   Il passeggero può accedere al sistema solamente per sapere a quale distanza si trova il veicolo dalla fermata, e per timbrare abbonamenti e biglietti

Strutture di memorizzazione

Manutenzione_veicolo		
Attributo	**Tipo di dato**	**Attributi**[2]
Manutenzione	VARCHAR(45)	PK NN
Veicolo	INT	PK NN

2 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Abbonamento		
Attributo	**Tipo di dato**	**Attributi**[3]
idAbbonamento	int	PK NN
Ultimo_utilizzo	DATETIME	

Convalidato		
Attributo	**Tipo di dato**	**Attributi**[4]
idAbbonamento	int	PK NN
Data	DATE	PK NN
Tratta	INT	PK NN
Veicolo	INT	PK NN

Biglietto		
Attributo	**Tipo di dato**	**Attributi**[5]
idBiglietto	int	PK NN
Stato	Int	
Data	Date	
Tratta	Int	
Veicolo	Int	

In_programma		
Attributo	**Tipo di dato**	**Attributi**[6]

3 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

4 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

5 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

6 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Veicolo	int	PK NN
Trata	int	PK NN
Numero	int	NN
Data	DATE	PK NN

Tratta_effettiva		
Attributo	**Tipo di dato**	**Attributi**[7]
Data	Data	PK NN
Tratta	int	PK NN
Conducente	int	NN
Veicolo	int	NN PK

Veicolo		
Attributo	**Tipo di dato**	**Attributi**[8]
Matricola	int	PK NN
Data_Acquisto	Data	NN

Turno		
Attributo	**Tipo di dato**	**Attributi**[9]
Data_inizio	TIME	PK NN
Data_fine	TIME	PK NN

Tratta		
Attributo	**Tipo di dato**	**Attributi**[10]

7 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

8 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

9 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

idTratta	int	PK NN
Latitudine_inziale	Varchar(45)	NN
Longitudine_inziale	Varchar(45)	NN
Latitudine_finale	Varchar(45)	NN
Longitudine_finale	Varchar(45)	NN

Conducente		
Attributo	**Tipo di dato**	**Attributi**[11]
CF	VARCHAR(16)	PK NN
Nome	VARCHAR(45)	NN
Cognome	VARCHAR(45)	NN
Data_nascita	DATA	NN
Luogo_nascita	VARCHAR(45)	NN
Scadenza_patente	DATA	NN
Numero_patente	VARCHAR(10)	NN
Username	VARCHAR(45)	NN

Turno_effettivo		
Attributo	**Tipo di dato**	**Attributi**[12]
Data_turno	Data	PK NN
Condcuente	VARCHAR(16)	PK NN
Ora_inizio	TIME	PK NN
Ora_fine	TIME	PK NN

10 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

11 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

12 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Fermata		
Attributo	**Tipo di dato**	**Attributi**[13]
IdTratta	int	PK NN
Latitudine	VARCHAR(12)	PK NN
Lognitudine	VARCHAR(12)	PK NN
Numero_fermata	Int	

Presente		
Attributo	**Tipo di dato**	**Attributi**[14]
IdTratta	int	PK NN
Latitudine	VARCHAR(12)	PK NN
Lognitudine	VARCHAR(12)	PK NN
Numero_waypoint	Int	

Fermata		
Attributo	**Tipo di dato**	**Attributi**[15]
Latitudine	VARCHAR(12)	PK NN
Lognitudine	VARCHAR(12)	PK NN
OrarioPartenze	TIME	
Cod capolinea	int	
Cod_fermata	INT	

Utente		
Attributo	**Tipo di dato**	**Attributi**[16]
Username	VARCHAR(45)	PK NN

13 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

14 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

15 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Password	VARCHAR(45)	NN
Ruolo	VARCHAR(45)	NN

Passato		
Attributo	**Tipo di dato**	**Attributi**[17]
WayPoint_Latitudine	VARCHAR(12)	NN
WayPoint_Lognitudine	VARCHAR(12)	NN
Veicolo	TIME	PK NN
Data		PK NN
Orario		PK NN
Tratta		PK NN

Manutenzione		
Attributo	**Tipo di dato**	**Attributi**[18]
Tipo_manutenzione	VARCHAR(45)	PK NN

Indici

In_programma	
Indice	**Tipo**[19]**:**
fk_Veicolo_has_Tratta_Tratta1_idx	Primary
fk_Veicolo_has_Tratta_Veicolo1_idx	Primary

16 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

17 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

18 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

19 IDX = index, UQ = unique, FT = full text, PR = primary.

Convalidato	
Indice	**Tipo[20]:**
fk_Abbonamento_has_Tratta_effettiva_Tratta_effettiva1_idx	Primary
fk_Abbonamento_has_Tratta_effettiva_Abbonamento1_idx	Primary

Biglietto	
Indice	**Tipo[21]:**
fk_Biglietto_Tratta_effettiva1_idx	Secondary

Tratta_effettiva	
Indice	**Tipo[22]:**
fk_Tratta effettiva_Tratta_idx	Primary
fk_Tratta effettiva_Conducente1_idx	Secondary
fk_Tratta effettiva_Veicolo1_idx	Primary

Tratta	
Indice	**Tipo[23]:**
fk_Tratta_WayPoint1_idx	Secondary
fk_Tratta_WayPoint2_idx	Secondary

Turno_effettivo	
Indice	**Tipo[24]:**

20 IDX = index, UQ = unique, FT = full text, PR = primary.

21 IDX = index, UQ = unique, FT = full text, PR = primary.

22 IDX = index, UQ = unique, FT = full text, PR = primary.

23 IDX = index, UQ = unique, FT = full text, PR = primary.

24 IDX = index, UQ = unique, FT = full text, PR = primary.

fk_Turno effettivo_Turno1_idx	Primary
fk_Turno effettivo_Conducente1_idx	Primary

Conducente	
Indice	**Tipo[25]:**
fk_Conducente_Utente1_idx	Primary

Fermata	
Indice	**Tipo[26]:**
fk_Tratta_has_WayPoint_WayPoint2_idx	Primary
fk_Tratta_has_WayPoint_WayPoint2_idx	Primary

Presente	
Indice	**Tipo[27]:**
fk_Tratta_has_WayPoint_WayPoint1_idx	Primary
fk_Tratta_has_WayPoint_WayPoint1_idx	Primary

Passato	
Indice	**Tipo[28]:**
fk_Tratta_effettiva_has_WayPoint_WayPoint1_idx	Secondary
fk_Tratta_effettiva_has_WayPoint_Tratta_effettiva1_idx	Secondary

25 IDX = index, UQ = unique, FT = full text, PR = primary.

26 IDX = index, UQ = unique, FT = full text, PR = primary.

27 IDX = index, UQ = unique, FT = full text, PR = primary.

28 IDX = index, UQ = unique, FT = full text, PR = primary.

Manutenzione_Veicolo	
Indice	**Tipo[29]:**
fk_Manutenzione_has_Veicolo_Veicolo1_idx	Primary
fk_Manutenzione_has_Veicolo_Manutenzione1_idx	Primary

Trigger

CREATE DEFINER = CURRENT_USER TRIGGER `SistemaTrasportoPubblico3`.`Turno_effettivo_BEFORE_INSERT` BEFORE INSERT ON `Turno_effettivo` FOR EACH ROW

BEGIN

if Ora_Inizio <= Ora_fine + 8

then

signal sqlstate '45000';

end if;

non ci sono altri trigger

END

Eventi

Non ci sono eventi specifici

Stored Procedures e transazioni

1) **Assegna_turno_al_conducente**

29 IDX = index, UQ = unique, FT = full text, PR = primary.

```
CREATE PROCEDURE `Assegna_turno_al_conducente` (IN var_conducente
VARCHAR(16),IN var_data DATE , IN var_OraInizio TIME, IN var_OraFine TIME)
BEGIN


        INSERT INTO `turno_effettivo` (`Data_turno`, `Conducente`, `Ora_inizio`,
`Ora_fine`) values (var_data, var_conducente, var_OraInizio, var_OraFine);
END
```

2) **Assegna_veicolo_alla_tratta**

```
CREATE PROCEDURE `Assegna_veicolo_alla_tratta` (IN var_Veicolo INT, In var_Tratta
INT, IN var_Conducente VARCHAR(16),IN var_Data DATE)
BEGIN
                set transaction isolation level repeatable read;
                start transaction;
                IF var_Conducente in (Select Turno_effettivo.Conducente
                                                From `Turno_effettivo`
                                                Where
Turno_effettivo.Conducente = var_Conducente AND Turno_effettivo.Data_turno =
var_Data)
                AND var_Conducente not in( Select Tratta_effettiva.Conducente
                                                FROM `Tratta_effettiva`
                                                WHERE
Tratta_effettiva.Conducente=var_Conducente  and Tratta_effettiva.Data = var_Data )
                                                THEN
                                                IF var_veicolo not in (Select
Tratta_effettiva.Veicolo

                From `Tratta_effettiva`

                Where Tratta_effettiva.Veicolo = var_veicolo and Tratta_effettiva.Data =
var_Data)

                THEN

                INSERT into `Tratta_effettiva` (`Data`, `Tratta`, `Conducente`, `Veicolo`)
values (var_Data,var_Tratta,var_Conducente,var_Veicolo);
                                                ELSE
                                                signal sqlstate '45002' set
message_text = "Veicolo non disponibile";
                                                end if;
                ELSE
                signal sqlstate '45002' set message_text = "Conducenti non disponibile";
                end if;
        commit;
END
```

3) **Assumi_conducente**

```
CREATE PROCEDURE `Assumi_conducente` (IN var_CF VARCHAR(16), IN var_Nome
VARCHAR(45), IN var_Cognome VARCHAR(45), IN var_Data_nascita DATE , IN
var_Luogo_nascita VARCHAR(45), IN var_Scadenza_patente DATE, IN
var_Numero_patente VARCHAR(10), IN var_username VARCHAR(45))
BEGIN
        set transaction isolation level repeatable read;
   start transaction;
        insert into `Conducente` (CF, Nome, Cognome, Data_nascita, Luogo_nascita,
Scadenza_patente, Numero_patente, Utente_Username) values (var_CF, var_Nome,
var_Cognome, var_Data_nascita, var_Luogo_nascita, var_Scadenza_patente,
var_Numero_patente,var_username);
   commit;
END
```

4) **Calcola_distanza_veicolo**

```
CREATE PROCEDURE `Calcola_distanza_veicolo` (IN var_fermata INT)

BEGIN

        SELECT WayPoint.latitudine, WayPoint.longitudine

        fROM `WayPoint`

        WHERE WayPoint.Cod_Fermata = var_fermata;



        SELECT p.WayPoint_Latitudine, p.WayPoint_Longitudine, p.Veicolo

        FROM `Passato` as P

        JOIN `Tratta_effettiva` as TE on (P.Veicolo = TE.Veicolo AND P.Tratta = TE.Tratta
and P.Data = TE.Data)

        JOIN `Tratta` as T on (TE.Tratta = T.IdTratta)

        JOIN `Fermata` as F on (F.IdTratta = T.IdTratta)

        JOIN `WayPoint` as W on (W.latitudine = F.Latitudine AND W.Longitudine =
F.Longitudine)

        WHERE W.Cod_Fermata = var_fermata

        AND TE.Data = CURRENT_DATE

        AND p.Orario = (SELECT  MAX(p.Orario)

                        FROM `Passato` as p

                        JOIN `Tratta_effettiva` as te on (p.Veicolo = te.Veicolo
AND p.Tratta = te.Tratta and p.Data = te.Data)

                        JOIN `Tratta` as t on (te.Tratta = t.IdTratta)

                        JOIN `Fermata` as f on (f.IdTratta = t.IdTratta)
```

JOIN `WayPoint` as w on (w.latitudine = f.Latitudine AND w.Longitudine = f.Longitudine)

WHERE w.Cod_Fermata = var_fermata and te.Data = CURRENT_DATE and te.Veicolo = TE.Veicolo);

END

5) **Calcola_prossima_tratta_veicolo**

```
CREATE PROCEDURE `Calcola_prossima_tratta_veicolo` (IN var_Veicolo INT,OUT
var_Tratta INT)
BEGIN
SELECT T.IdTratta
FROM `In_Programma` as I
JOIN `Tratta` as T on (I.Tratta = T.IdTratta)
JOIN `Veicolo` as V on (I.Veicolo = V.Matricola)
WHERE V.Matricola = var_Veicolo AND (I.Numero = 1 + (    SELECT I.Numero

        FROM In_Programma as I

        JOIN Veicolo as V on (I.Veicolo = V.Matricola)

        JOIN Tratta_Effettiva as TE on (TE.Veicolo = V.Matricola)

        WHERE V.Matricola = var_Veicolo and I.Data = CURRENT_DATE AND I.Tratta =
TE.Tratta))into var_Tratta;
END
```

6) **Cambia_conducente_turno**

```
CREATE PROCEDURE `Cambia_conducente_turno` (IN var_conducente1
VARCHAR(16),IN var_Data DATE,IN var_conducente2 VARCHAR(16))
BEGIN
                declare var_OraInizio TIME;
                declare var_OraFine TIME;
                declare var_Tratta int;
                declare var_Veicolo int;
                if exists (Select turno_effettivo.Conducente
                                FROM `turno_effettivo`
                                WHERE turno_effettivo.Conducente = var_Conducente1 and
turno_effettivo.Data_turno= var_Data) /*ritorna true o false*/
                                then
                                Select turno_effettivo.Ora_inizio, turno_effettivo.Ora_fine
                                FROM `turno_effettivo`
                                WHERE turno_effettivo.Conducente = var_Conducente1 and
turno_effettivo.Data_turno = var_Data into var_OraInizio, var_OraFine; /*ritorna inizio e
fine di quel turno*/
                                if not exists( Select turno_effettivo.Conducente
                                                FROM `turno_effettivo`
```

```
                                                    WHERE
turno_effettivo.Conducente = var_Conducente2 and turno_effettivo.Data_turno=
var_Data)/*ritorna true o false*/
                                                    then
                                                    Delete
                                                    FROM `turno_effettivo`
                                                    where
turno_effettivo.Conducente = var_conducente1 AND turno_effettivo.Data_turno =
var_Data;/*non ritorna nulla*/

                                                    insert into `turno_effettivo`
(Conducente,data_turno, Ora_inizio,Ora_fine)
                                                    values (var_conducente2,
var_Data, var_OraInizio, var_OraFine); /*non ritorna nulla*/


                                                    if exists (Select
tratta_effettiva.Conducente
                                                                FROM
`tratta_effettiva`
                                                                WHERE
tratta_effettiva.Conducente = var_Conducente1 and tratta_effettiva.Data= var_Data)/*ritorna
true o false*/
                                                                THEN
                                                                Select
tratta_effettiva.Tratta, tratta_effettiva.Veicolo
                                                                FROM
`tratta_effettiva`
                                                                WHERE
tratta_effettiva.Conducente = var_Conducente1 and tratta_effettiva.Data = var_Data into
var_Tratta, var_Veicolo; /*ritorna tratta e veicolo di quella tratta*/

                                                                DELETE
                                                                FROM
`tratta_effettiva`
                                                                WHERE
tratta_effettiva.Conducente = var_Conducente1 and tratta_effettiva.Data = var_Data;/*non
ritorna nulla*/

                                                                if not
exists( Select tratta_effettiva.Conducente

                        FROM `tratta_effettiva`

                        WHERE tratta_effettiva.Conducente = var_Conducente2 and
tratta_effettiva.Data= var_Data)/*ritorna true o false*/
```

```
                    THEN

                    insert into `tratta_effettiva` (Data,Tratta,Veicolo,Conducente) values
(var_Data,var_Tratta,var_Veicolo,var_Conducente2);/*non ritorna nulla*/
                                                    ELSE
                                                        signal sqlstate '45002' set
message_text = "Il conducente1 non sta guidando un veicolo in quella data";
                                                        end if;
                                                            ELSE

        signal sqlstate '45002' set message_text = "Il conducente2 gia sta guidando un veicolo
in quella data";
                                                                end
if;

                            ELSE
                                signal sqlstate '45002' set message_text = "Il
conducente2 ha gia un turno in quella data";
                                end if;
        ELSE
            signal sqlstate '45002' set message_text = "Il conducente1 non ha un turno in
quella data";
            end if;

END
```

7) **Convalida_abbonamento**

```
CREATE PROCEDURE `Convalida_abbonamento` (IN var_Abbonamento INT, IN
var_Tratta INT, IN var_Veicolo INT)
BEGIN
        insert into `Convalidato` (idAbbonamento, Data, Tratta,  Veicolo) values
(var_Abbonamento, current_date(),  var_Tratta, var_Veicolo);
        update `Abbonamento`
        set  Ultimo_utilizzo = now()
        WHERE Abbonamento.IdAbbonamento = var_Abbonamento;
END
```

8) **Elimina_conducente**

```
CREATE PROCEDURE `Elimina_conducente` (IN var_CF VARCHAR(16), IN
var_Numero_patente VARCHAR(10))
BEGIN
        Delete
   from `Conducente`
        where Conducente.CF = var_CF   AND  Conducente.Numero_patente =
var_Numero_patente ;
END
```

9) **Login**

```
CREATE PROCEDURE `Login` (in var_username varchar(45), in var_pass varchar(45), out
var_role INT)
BEGIN
	declare var_user_role ENUM('amministratore', 'conducente','passeggero');
	select `ruolo` from `utente`
		where `username` = var_username
	and `password` = md5(var_pass)
	into var_user_role;

	-- See the corresponding enum in the client
		if var_user_role = 'amministratore' then
			set var_role = 1;
		elseif var_user_role = 'conducente' then
			set var_role = 2;
		elseif var_user_role = 'passeggero' then
			set var_role = 3;
		else
			set var_role = 4;
		end if;
END
```

10) **Timbra_biglietto**

```
CREATE PROCEDURE `Timbra_biglietto` (IN var_id_Biglietto int, IN var_Tratta INT,  IN
var_Veicolo INT )
BEGIN
	IF EXISTS
			(Select *
			FROM `Tratta_effettiva`
			WHERE Tratta_effettiva.Veicolo = var_veicolo and
Tratta_effettiva.Tratta = var_tratta and Tratta_effettiva.Data = CURRENT_DATE)
	THEN
			UPDATE  `biglietto`
			SET biglietto.Stato = 1,  biglietto.Data=current_date() ,biglietto.Tratta
= var_Tratta  , biglietto.Veicolo = var_Veicolo
			WHERE biglietto.idBiglietto = var_id_Biglietto;
	ELSE
			signal sqlstate '45002' set message_text = "Tratta effettiva non
esistente";
	END IF;
END
```

11) **Visualizza_conducenti_attivi**

```
CREATE PROCEDURE `Visualizza_conducenti_attivi` ()
BEGIN
	set transaction isolation level repeatable read;
   start transaction;
	drop temporary table if exists `Utenti_attivi`;
	create temporary table `Utenti_attivi` (
```

```
        `CF` varchar(45));


    insert into `Utenti_attivi`
        SELECT DISTINCT Conducente.CF
    FROM `Conducente`
        JOIN `Turno_effettivo` on (Conducente.CF = Turno_effettivo.Conducente)
    WHERE Turno_effettivo.Data_turno = CURRENT_DATE
        OR Conducente.cf in (Select Tratta_effettiva.Conducente
                                from `Tratta_effettiva`
                                where Tratta_effettiva.Conducente =
Conducente.cf and Tratta_effettiva.Data = current_date());


    select * from `Utenti_attivi`;
    drop temporary table `Utenti_attivi`;
    commit;

END
```

12) **Visualizza_conducenti_fermi**

```
CREATE PROCEDURE `Visualizza_conducenti_fermi` ()
BEGIN
        drop temporary table if exists `Utenti_fermi`;
        create temporary table `Utenti_fermi` ( `CF` varchar(45));

        set transaction isolation level serializable;
    start transaction;
    insert into `Utenti_fermi`
        SELECT Conducente.CF
    FROM `Conducente`
    WHERE Conducente.CF not in (Select t.Conducente
                                FROM `Turno_effettivo` as `t`
JOIN `Conducente` as `c` on (t.Conducente= c.CF)
                                WHERE c.CF = Conducente.CF
and t.Data_turno = CURRENT_DATE)
        AND Conducente.CF not in
                            (SELECT Tratta_effettiva.Conducente
                                FROM `Tratta_effettiva`
                                WHERE
Tratta_effettiva.Conducente = Conducente.CF AND Tratta_effettiva.Data =
CURRENT_DATE);
        commit;
        select * from `Utenti_fermi`;
    drop temporary table `Utenti_fermi`;
    END
```

13) **Visualizza_veicoli_attivi**

```
CREATE PROCEDURE `Visualizza_veicoli_attivi` ()
```

```
BEGIN
        set transaction isolation level serializable;
    start transaction;
        SELECT veicolo.matricola
    FROM `veicolo` JOIN `Tratta_effettiva` on (Tratta_effettiva.Veicolo = Veicolo.Matricola)
    WHERE Tratta_effettiva.Data = CURRENT_DATE ;
    commit;
END
```

14) **Visualizza_veicoli_fermi**

```
CREATE PROCEDURE `Visualizza_veicoli_fermi` ()
BEGIN
        set transaction isolation level repeatable read;
    start transaction;
        SELECT veicolo.matricola
    FROM `veicolo`
    WHERE  veicolo.matricola  not in (Select t.Veicolo
                                                    FROM `Tratta_effettiva`
as `t` JOIN `veicolo` as `v`  on (t.Veicolo = v.Matricola)
                                                    WHERE v.Matricola =
veicolo.Matricola AND t.Data = CURRENT_DATE);
                commit;
END
```

15) **Emetti abboamenti**

```
CREATE PROCEDURE `Emetti_abbonamenti` ()
BEGIN
        insert into `Abbonamento`() values ();
END
```

16) **Emetti biglietti**

```
CREATE PROCEDURE `Emetti_biglietto` ()
BEGIN
        insert into `Biglietto` (Stato) values ("0");
END
```

17) **Aggiungi utente**

```
CREATE PROCEDURE `Aggiungi_utente` (IN var_username VARCHAR(45), IN
var_password VARCHAR(45), IN var_ruolo VARCHAR(16))
BEGIN
        set transaction isolation level repeatable read;
    start transaction;
        insert into `Utente` (`Username`, `Password`, `Ruolo`) values (var_username,
MD5(var_password), var_ruolo);
    commit;
END
```

Mostrare e commentare le stored procedure che sono state realizzate per implementare la logica applicativa delle operazioni sui dati, evidenziando quando (e perché) sono state realizzate operazioni transazionali complesse.

Appendice: Implementazione

Codice SQL per instanziare il database

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema SistemaTrasportoPubblico3
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema SistemaTrasportoPubblico3
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `SistemaTrasportoPubblico3` DEFAULT CHARACTER SET utf8 ;
USE `SistemaTrasportoPubblico3` ;

-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Veicolo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Veicolo` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Veicolo` (
  `Matricola` INT NOT NULL,
  `Data_acquisto` DATE NOT NULL,
  PRIMARY KEY (`Matricola`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`WayPoint`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`WayPoint` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`WayPoint` (
  `Latitudine` VARCHAR(12) NOT NULL,
  `Longitudine` VARCHAR(12) NOT NULL,
  `OrarioPartenze` TIME NULL,
  `Cod_fermata` INT NULL,
  `Cod_capolinea` INT NULL,
  PRIMARY KEY (`Latitudine`, `Longitudine`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Tratta`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Tratta` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Tratta` (
  `idTratta` INT NOT NULL,
  `Latitudine_inziale` VARCHAR(12) NOT NULL,
  `Longitudine_inziale` VARCHAR(12) NOT NULL,
  `Latitudine_finale` VARCHAR(12) NOT NULL,
  `Longitudine_finale` VARCHAR(12) NOT NULL,
  PRIMARY KEY (`idTratta`),
  INDEX `fk_Tratta_WayPoint1_idx` (`Latitudine_inziale` ASC, `Longitudine_inziale` ASC),
  INDEX `fk_Tratta_WayPoint2_idx` (`Latitudine_finale` ASC, `Longitudine_finale` ASC),
  CONSTRAINT `fk_Tratta_WayPoint1`
    FOREIGN KEY (`Latitudine_inziale` , `Longitudine_inziale`)
```

```
    REFERENCES `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine` , `Longitudine`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tratta_WayPoint2`
    FOREIGN KEY (`Latitudine_finale` , `Longitudine_finale`)
    REFERENCES `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine` , `Longitudine`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Utente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Utente` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Utente` (
  `Username` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NULL,
  `Ruolo` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Username`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Conducente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Conducente` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Conducente` (
  `CF` VARCHAR(16) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
```

```
  `Cognome` VARCHAR(45) NOT NULL,
  `Data_nascita` DATE NOT NULL,
  `Luogo_nascita` VARCHAR(45) NOT NULL,
  `Scadenza_patente` DATE NOT NULL,
  `Numero_patente` VARCHAR(10) NOT NULL,
  `Utente_Username` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CF`),
  INDEX `fk_Conducente_Utente1_idx` (`Utente_Username` ASC),
  CONSTRAINT `fk_Conducente_Utente1`
    FOREIGN KEY (`Utente_Username`)
    REFERENCES `SistemaTrasportoPubblico3`.`Utente` (`Username`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Tratta_effettiva`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Tratta_effettiva` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Tratta_effettiva` (
  `Data` DATE NOT NULL,
  `Tratta` INT NOT NULL,
  `Conducente` VARCHAR(16) NOT NULL,
  `Veicolo` INT NOT NULL,
  PRIMARY KEY (`Data`, `Tratta`, `Veicolo`),
  INDEX `fk_Tratta effettiva_Tratta_idx` (`Tratta` ASC),
  INDEX `fk_Tratta effettiva_Conducente1_idx` (`Conducente` ASC),
  INDEX `fk_Tratta effettiva_Veicolo1_idx` (`Veicolo` ASC),
  CONSTRAINT `fk_Tratta effettiva_Tratta`
    FOREIGN KEY (`Tratta`)
```

```
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Tratta effettiva_Conducente1`
    FOREIGN KEY (`Conducente`)
    REFERENCES `SistemaTrasportoPubblico3`.`Conducente` (`CF`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Tratta effettiva_Veicolo1`
    FOREIGN KEY (`Veicolo`)
    REFERENCES `SistemaTrasportoPubblico3`.`Veicolo` (`Matricola`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Biglietto`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Biglietto` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Biglietto` (
  `idBiglietto` INT NOT NULL AUTO_INCREMENT,
  `Stato` INT NULL DEFAULT 0,
  `Data` DATE NULL,
  `Tratta` INT NULL,
  `Veicolo` INT NULL,
  PRIMARY KEY (`idBiglietto`),
  INDEX `fk_Biglietto_Tratta_effettiva1_idx` (`Data` ASC, `Tratta` ASC, `Veicolo` ASC),
  CONSTRAINT `fk_Biglietto_Tratta_effettiva1`
    FOREIGN KEY (`Data` , `Tratta` , `Veicolo`)
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta_effettiva` (`Data` , `Tratta` , `Veicolo`)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Abbonamento`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Abbonamento` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Abbonamento` (
  `idAbbonamento` INT NOT NULL AUTO_INCREMENT,
  `Ultimo_utilizzo` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`idAbbonamento`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Turno`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Turno` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Turno` (
  `Ora_inizio` TIME NOT NULL,
  `Ora_fine` TIME NOT NULL,
  PRIMARY KEY (`Ora_inizio`, `Ora_fine`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Manutenzione`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Manutenzione` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Manutenzione` (
  `Tipo_manutenzione` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Tipo_manutenzione`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Turno_effettivo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Turno_effettivo` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Turno_effettivo` (
  `Data_turno` DATE NOT NULL,
  `Conducente` VARCHAR(16) NOT NULL,
  `Ora_inizio` TIME NOT NULL,
  `Ora_fine` TIME NOT NULL,
  PRIMARY KEY (`Data_turno`, `Conducente`, `Ora_inizio`, `Ora_fine`),
  INDEX `fk_Turno effettivo_Conducente1_idx` (`Conducente` ASC),
  INDEX `fk_Turno effettivo_Turno1_idx` (`Ora_inizio` ASC, `Ora_fine` ASC),
  CONSTRAINT `fk_Turno effettivo_Conducente1`
    FOREIGN KEY (`Conducente`)
    REFERENCES `SistemaTrasportoPubblico3`.`Conducente` (`CF`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Turno effettivo_Turno1`
    FOREIGN KEY (`Ora_inizio` , `Ora_fine`)
    REFERENCES `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio` , `Ora_fine`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Manutenzione_Veicolo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Manutenzione_Veicolo` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Manutenzione_Veicolo` (
  `Manutenzione` VARCHAR(45) NOT NULL,
  `Veicolo` INT NOT NULL,
  PRIMARY KEY (`Manutenzione`, `Veicolo`),
  INDEX `fk_Manutenzione_has_Veicolo_Veicolo1_idx` (`Veicolo` ASC),
  INDEX `fk_Manutenzione_has_Veicolo_Manutenzione1_idx` (`Manutenzione` ASC),
  CONSTRAINT `fk_Manutenzione_has_Veicolo_Manutenzione1`
    FOREIGN KEY (`Manutenzione`)
    REFERENCES `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Manutenzione_has_Veicolo_Veicolo1`
    FOREIGN KEY (`Veicolo`)
    REFERENCES `SistemaTrasportoPubblico3`.`Veicolo` (`Matricola`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`In_programma`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`In_programma` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`In_programma` (
```

```
  `Veicolo` INT NOT NULL,
  `Tratta` INT NOT NULL,
  `Numero` INT NOT NULL,
  `Data` DATE NOT NULL,
  PRIMARY KEY (`Veicolo`, `Tratta`, `Data`),
  INDEX `fk_Veicolo_has_Tratta_Tratta1_idx` (`Tratta` ASC),
  INDEX `fk_Veicolo_has_Tratta_Veicolo1_idx` (`Veicolo` ASC),
  CONSTRAINT `fk_Veicolo_has_Tratta_Veicolo1`
    FOREIGN KEY (`Veicolo`)
    REFERENCES `SistemaTrasportoPubblico3`.`Veicolo` (`Matricola`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Veicolo_has_Tratta_Tratta1`
    FOREIGN KEY (`Tratta`)
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Convalidato`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Convalidato` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Convalidato` (
  `idAbbonamento` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Tratta` INT NOT NULL,
  `Veicolo` INT NOT NULL,
  PRIMARY KEY (`idAbbonamento`, `Data`, `Tratta`, `Veicolo`),
```

```
  INDEX `fk_Abbonamento_has_Tratta_effettiva_Tratta_effettiva1_idx` (`Data` ASC, `Tratta` ASC, `Veicolo` ASC),
  INDEX `fk_Abbonamento_has_Tratta_effettiva_Abbonamento1_idx` (`idAbbonamento` ASC),
  CONSTRAINT `fk_Abbonamento_has_Tratta_effettiva_Abbonamento1`
    FOREIGN KEY (`idAbbonamento`)
    REFERENCES `SistemaTrasportoPubblico3`.`Abbonamento` (`idAbbonamento`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Abbonamento_has_Tratta_effettiva_Tratta_effettiva1`
    FOREIGN KEY (`Data` , `Tratta` , `Veicolo`)
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta_effettiva` (`Data` , `Tratta` , `Veicolo`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Passato`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Passato` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Passato` (
  `Data` DATE NOT NULL,
  `Tratta` INT NOT NULL,
  `Veicolo` INT NOT NULL,
  `WayPoint_Latitudine` VARCHAR(12) NOT NULL,
  `WayPoint_Longitudine` VARCHAR(12) NOT NULL,
  `Orario` TIME NOT NULL,
  PRIMARY KEY (`Data`, `Tratta`, `Veicolo`, `Orario`),
    INDEX `fk_Tratta_effettiva_has_WayPoint_WayPoint1_idx` (`WayPoint_Latitudine` ASC, `WayPoint_Longitudine` ASC),
    INDEX `fk_Tratta_effettiva_has_WayPoint_Tratta_effettiva1_idx` (`Data` ASC, `Tratta` ASC, `Veicolo` ASC),
```

```
  CONSTRAINT `fk_Tratta_effettiva_has_WayPoint_Tratta_effettiva1`
    FOREIGN KEY (`Data` , `Tratta` , `Veicolo`)
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta_effettiva` (`Data` , `Tratta` , `Veicolo`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Tratta_effettiva_has_WayPoint_WayPoint1`
    FOREIGN KEY (`WayPoint_Latitudine` , `WayPoint_Longitudine`)
    REFERENCES `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine` , `Longitudine`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Presente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Presente` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Presente` (
  `idTratta` INT NOT NULL,
  `Latitudine` VARCHAR(12) NOT NULL,
  `Longitudine` VARCHAR(12) NOT NULL,
  `Numero_WayPoint` INT NULL,
  PRIMARY KEY (`idTratta`, `Latitudine`, `Longitudine`),
  INDEX `fk_Tratta_has_WayPoint_WayPoint1_idx` (`Latitudine` ASC, `Longitudine` ASC),
  INDEX `fk_Tratta_has_WayPoint_Tratta1_idx` (`idTratta` ASC),
  CONSTRAINT `fk_Tratta_has_WayPoint_Tratta1`
    FOREIGN KEY (`idTratta`)
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Tratta_has_WayPoint_WayPoint1`
```

```
    FOREIGN KEY (`Latitudine` , `Longitudine`)
    REFERENCES `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine` , `Longitudine`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SistemaTrasportoPubblico3`.`Fermata`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SistemaTrasportoPubblico3`.`Fermata` ;

CREATE TABLE IF NOT EXISTS `SistemaTrasportoPubblico3`.`Fermata` (
  `idTratta` INT NOT NULL,
  `Latitudine` VARCHAR(12) NOT NULL,
  `Longitudine` VARCHAR(12) NOT NULL,
  `Numero_fermata` INT NULL,
  PRIMARY KEY (`idTratta`, `Latitudine`, `Longitudine`),
  INDEX `fk_Tratta_has_WayPoint_WayPoint2_idx` (`Latitudine` ASC, `Longitudine` ASC),
  INDEX `fk_Tratta_has_WayPoint_Tratta2_idx` (`idTratta` ASC),
  CONSTRAINT `fk_Tratta_has_WayPoint_Tratta2`
    FOREIGN KEY (`idTratta`)
    REFERENCES `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Tratta_has_WayPoint_WayPoint2`
    FOREIGN KEY (`Latitudine` , `Longitudine`)
    REFERENCES `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine` , `Longitudine`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
USE `SistemaTrasportoPubblico3` ;


-- -----------------------------------------------------

-- procedure Assegna_turno_al_conducente

-- -----------------------------------------------------


USE `SistemaTrasportoPubblico3`;

DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Assegna_turno_al_conducente`;


DELIMITER $$

USE `SistemaTrasportoPubblico3`$$

CREATE PROCEDURE `Assegna_turno_al_conducente` (IN var_conducente VARCHAR(16),IN var_data DATE , IN var_OraInizio TIME, IN var_OraFine TIME)

BEGIN



        INSERT INTO `turno_effettivo` (`Data_turno`, `Conducente`, `Ora_inizio`, `Ora_fine`) values (var_data, var_conducente, var_OraInizio, var_OraFine);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure Assegna_veicolo_alla_tratta

-- -----------------------------------------------------


USE `SistemaTrasportoPubblico3`;

DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Assegna_veicolo_alla_tratta`;


DELIMITER $$

USE `SistemaTrasportoPubblico3`$$

CREATE PROCEDURE `Assegna_veicolo_alla_tratta` (IN var_Veicolo INT, In var_Tratta INT, IN var_Conducente VARCHAR(16),IN var_Data DATE)
```

```
BEGIN
        set transaction isolation level repeatable read;
        start transaction;
        IF var_Conducente in (Select Turno_effettivo.Conducente
                                                From `Turno_effettivo`
                                                Where Turno_effettivo.Conducente = var_Conducente AND Turno_effettivo.Data_turno = var_Data)
        AND var_Conducente not in( Select Tratta_effettiva.Conducente
                                                FROM `Tratta_effettiva`
                                                WHERE Tratta_effettiva.Conducente=var_Conducente  and Tratta_effettiva.Data = var_Data )
                                                THEN
                                                IF var_veicolo not in (Select Tratta_effettiva.Veicolo
    From `Tratta_effettiva`
    Where Tratta_effettiva.Veicolo = var_veicolo and Tratta_effettiva.Data = var_Data)
    THEN
    INSERT into `Tratta_effettiva` (`Data`, `Tratta`, `Conducente`, `Veicolo`) values (var_Data,var_Tratta,var_Conducente,var_Veicolo);
                                                ELSE
                                                signal sqlstate '45002' set message_text = "Veicolo non disponibile";
                                                end if;
        ELSE
        signal sqlstate '45002' set message_text = "Conducenti non disponibile";
        end if;
  commit;
END$$

DELIMITER ;
```

```
-- -----------------------------------------------------
-- procedure Assumi_conducente
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Assumi_conducente`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Assumi_conducente` (IN var_CF VARCHAR(16), IN var_Nome VARCHAR(45), IN var_Cognome VARCHAR(45), IN var_Data_nascita DATE , IN var_Luogo_nascita VARCHAR(45), IN var_Scadenza_patente DATE, IN var_Numero_patente VARCHAR(10), IN var_username VARCHAR(45))
BEGIN
	set transaction isolation level repeatable read;
   start transaction;
	insert into `Conducente` (CF, Nome, Cognome, Data_nascita, Luogo_nascita, Scadenza_patente, Numero_patente, Utente_Username) values (var_CF, var_Nome, var_Cognome, var_Data_nascita, var_Luogo_nascita, var_Scadenza_patente, var_Numero_patente,var_username);
   commit;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Elimina_conducente
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Elimina_conducente`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
```

```
CREATE PROCEDURE `Elimina_conducente` (IN var_CF VARCHAR(16), IN var_Numero_patente VARCHAR(10))
BEGIN
	Delete
   from `Conducente`
	where Conducente.CF = var_CF AND Conducente.Numero_patente = var_Numero_patente ;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Calcola_distanza_veicolo
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Calcola_distanza_veicolo`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Calcola_distanza_veicolo` (IN var_fermata INT)
BEGIN
		SELECT WayPoint.latitudine, WayPoint.longitudine
		fROM `WayPoint`
		WHERE WayPoint.Cod_Fermata = var_fermata;


		SELECT p.WayPoint_Latitudine, p.WayPoint_Longitudine, p.Veicolo
		FROM `Passato` as P
		JOIN `Tratta_effettiva` as TE on (P.Veicolo = TE.Veicolo AND P.Tratta = TE.Tratta and P.Data = TE.Data)
		JOIN `Tratta` as T on (TE.Tratta = T.IdTratta)
		JOIN `Fermata` as F on (F.IdTratta = T.IdTratta)
```

```
		JOIN `WayPoint` as W on (W.latitudine = F.Latitudine AND W.Longitudine = F.Longitudine)
		WHERE W.Cod_Fermata = var_fermata
		AND TE.Data = CURRENT_DATE
		AND p.Orario = (SELECT  MAX(p.Orario)
				FROM `Passato` as p
				JOIN `Tratta_effettiva` as te on (p.Veicolo = te.Veicolo AND p.Tratta = te.Tratta and p.Data = te.Data)
				JOIN `Tratta` as t on (te.Tratta = t.IdTratta)
				JOIN `Fermata` as f on (f.IdTratta = t.IdTratta)
				JOIN `WayPoint` as w on (w.latitudine = f.Latitudine AND w.Longitudine = f.Longitudine)
				WHERE w.Cod_Fermata = var_fermata and te.Data = CURRENT_DATE and te.Veicolo = TE.Veicolo);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Cambia_conducente_turno
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Cambia_conducente_turno`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Cambia_conducente_turno` (IN var_conducente1 VARCHAR(16),IN var_Data DATE,IN var_conducente2 VARCHAR(16))
BEGIN
		declare var_OraInizio TIME;
		declare var_OraFine TIME;
		declare var_Tratta int;
		declare var_Veicolo int;
```

```
if exists (Select turno_effettivo.Conducente
        FROM `turno_effettivo`
        WHERE turno_effettivo.Conducente = var_Conducente1 and turno_effettivo.Data_turno= var_Data) /*ritorna true o false*/
        then
        Select turno_effettivo.Ora_inizio, turno_effettivo.Ora_fine
        FROM `turno_effettivo`
        WHERE turno_effettivo.Conducente = var_Conducente1 and turno_effettivo.Data_turno = var_Data into var_OraInizio, var_OraFine; /*ritorna inizio e fine di quel turno*/
        if not exists( Select turno_effettivo.Conducente
                FROM `turno_effettivo`
                WHERE turno_effettivo.Conducente = var_Conducente2 and turno_effettivo.Data_turno= var_Data)/*ritorna true o false*/
                then
                Delete
                FROM `turno_effettivo`
                where turno_effettivo.Conducente = var_conducente1 AND turno_effettivo.Data_turno = var_Data;/*non ritorna nulla*/

                insert into `turno_effettivo` (Conducente,data_turno, Ora_inizio,Ora_fine)
                values (var_conducente2, var_Data, var_OraInizio, var_OraFine); /*non ritorna nulla*/


                if exists (Select tratta_effettiva.Conducente
                        FROM `tratta_effettiva`
                        WHERE tratta_effettiva.Conducente = var_Conducente1 and tratta_effettiva.Data= var_Data)/*ritorna true o false*/
                        THEN
                        Select tratta_effettiva.Tratta, tratta_effettiva.Veicolo
```

```
FROM
`tratta_effettiva`
WHERE
tratta_effettiva.Conducente = var_Conducente1 and tratta_effettiva.Data = var_Data into var_Tratta, var_Veicolo; /*ritorna tratta e veicolo di quella tratta*/

DELETE
FROM
`tratta_effettiva`
WHERE
tratta_effettiva.Conducente = var_Conducente1 and tratta_effettiva.Data = var_Data;/*non ritorna nulla*/

if not exists( Select tratta_effettiva.Conducente
FROM `tratta_effettiva`
WHERE tratta_effettiva.Conducente = var_Conducente2 and tratta_effettiva.Data= var_Data)/*ritorna true o false*/
THEN
insert into `tratta_effettiva` (Data,Tratta,Veicolo,Conducente) values (var_Data,var_Tratta,var_Veicolo,var_Conducente2);/*non ritorna nulla*/
ELSE
signal sqlstate '45002' set message_text = "Il conducente1 non sta guidando un veicolo in quella data";
end if;
ELSE
signal sqlstate '45002' set message_text = "Il conducente2 gia sta guidando un veicolo in quella data";
end if;

ELSE
signal sqlstate '45002' set message_text = "Il conducente2 ha gia un turno in quella data";
end if;
```

```
        ELSE
                signal sqlstate '45002' set message_text = "Il conducente1 non ha un turno in quella data";
                end if;

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Convalida_abbonamento
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Convalida_abbonamento`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Convalida_abbonamento` (IN var_Abbonamento INT, IN var_Tratta INT, IN var_Veicolo INT)
BEGIN
        insert into `Convalidato` (idAbbonamento, Data, Tratta,  Veicolo) values (var_Abbonamento, current_date(),  var_Tratta, var_Veicolo);
        update `Abbonamento`
        set  Ultimo_utilizzo = now()
        WHERE Abbonamento.IdAbbonamento = var_Abbonamento;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Calcola_prossima_tratta_veicolo
-- -----------------------------------------------------
```

```
USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Calcola_prossima_tratta_veicolo`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Calcola_prossima_tratta_veicolo` (IN var_Veicolo INT,OUT var_Tratta INT)
BEGIN
SELECT T.IdTratta
FROM `In_Programma` as I
JOIN `Tratta` as T on (I.Tratta = T.IdTratta)
JOIN `Veicolo` as V on (I.Veicolo = V.Matricola)
WHERE V.Matricola = var_Veicolo AND (I.Numero = 1 + (    SELECT I.Numero
                                                         FROM
In_Programma as I
                                                         JOIN
Veicolo as V on (I.Veicolo = V.Matricola)
                                                         JOIN
Tratta_Effettiva as TE on (TE.Veicolo = V.Matricola)
                                                         WHERE
V.Matricola = var_Veicolo and I.Data = CURRENT_DATE AND I.Tratta = TE.Tratta))into var_Tratta;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Timbra_biglietto
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Timbra_biglietto`;
```

```
DELIMITER $$

USE `SistemaTrasportoPubblico3`$$

CREATE PROCEDURE `Timbra_biglietto` (IN var_id_Biglietto int, IN var_Tratta INT,  IN var_Veicolo INT )

BEGIN

	IF EXISTS

			(Select *

			FROM `Tratta_effettiva`

			WHERE Tratta_effettiva.Veicolo = var_veicolo and Tratta_effettiva.Tratta = var_tratta and Tratta_effettiva.Data = CURRENT_DATE)

	THEN

			UPDATE  `biglietto`

			SET biglietto.Stato = 1,  biglietto.Data=current_date() ,biglietto.Tratta = var_Tratta  , biglietto.Veicolo = var_Veicolo

			WHERE biglietto.idBiglietto = var_id_Biglietto;

	ELSE

			signal sqlstate '45002' set message_text = "Tratta effettiva non esistente";

	END IF;

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure Visualizza_conducenti_attivi

-- -----------------------------------------------------


USE `SistemaTrasportoPubblico3`;

DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Visualizza_conducenti_attivi`;


DELIMITER $$

USE `SistemaTrasportoPubblico3`$$

CREATE PROCEDURE `Visualizza_conducenti_attivi` ()

BEGIN
```

```
        set transaction isolation level repeatable read;
    start transaction;
        drop temporary table if exists `Utenti_attivi`;
        create temporary table `Utenti_attivi` (
        `CF` varchar(45));


    insert into `Utenti_attivi`
        SELECT DISTINCT Conducente.CF
    FROM `Conducente`
        JOIN `Turno_effettivo` on (Conducente.CF = Turno_effettivo.Conducente)
    WHERE Turno_effettivo.Data_turno = CURRENT_DATE
        OR Conducente.cf in (Select Tratta_effettiva.Conducente
                                    from `Tratta_effettiva`
                                    where Tratta_effettiva.Conducente = Conducente.cf and
Tratta_effettiva.Data = current_date());


    select * from `Utenti_attivi`;
    drop temporary table `Utenti_attivi`;
    commit;

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Visualizza_conducenti_fermi
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Visualizza_conducenti_fermi`;
```

```
DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Visualizza_conducenti_fermi` ()
BEGIN
	drop temporary table if exists `Utenti_fermi`;
	create temporary table `Utenti_fermi` ( `CF` varchar(45));

	set transaction isolation level serializable;
   start transaction;
   insert into `Utenti_fermi`
	SELECT Conducente.CF
   FROM `Conducente`
   WHERE Conducente.CF not in (Select t.Conducente
                                                FROM `Turno_effettivo` as `t` JOIN `Conducente` as `c` on (t.Conducente= c.CF)
                                                WHERE c.CF = Conducente.CF and t.Data_turno = CURRENT_DATE)
	AND Conducente.CF not in
                                        (SELECT Tratta_effettiva.Conducente
                                                FROM `Tratta_effettiva`
                                                WHERE Tratta_effettiva.Conducente = Conducente.CF AND Tratta_effettiva.Data = CURRENT_DATE);
	commit;
	select * from `Utenti_fermi`;
   drop temporary table `Utenti_fermi`;
   END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Visualizza_veicoli_attivi
-- -----------------------------------------------------
```

```
USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Visualizza_veicoli_attivi`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Visualizza_veicoli_attivi` ()
BEGIN
        set transaction isolation level serializable;
    start transaction;
        SELECT veicolo.matricola
    FROM `veicolo` JOIN `Tratta_effettiva` on (Tratta_effettiva.Veicolo = Veicolo.Matricola)
    WHERE Tratta_effettiva.Data = CURRENT_DATE ;
    commit;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Visualizza_veicoli_fermi
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Visualizza_veicoli_fermi`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Visualizza_veicoli_fermi` ()
BEGIN
        set transaction isolation level repeatable read;
    start transaction;
        SELECT veicolo.matricola
```

```
   FROM `veicolo`
   WHERE  veicolo.matricola  not in (Select t.Veicolo
                                    FROM `Tratta_effettiva` as `t`
JOIN `veicolo` as `v`  on (t.Veicolo = v.Matricola)
                                    WHERE v.Matricola =
veicolo.Matricola AND t.Data = CURRENT_DATE);
                commit;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Login
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Login`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Login` (in var_username varchar(45), in var_pass varchar(45), out
var_role INT)
BEGIN
        declare var_user_role ENUM('amministratore', 'conducente','passeggero');
        select `ruolo` from `utente`
                where `username` = var_username
      and `password` = md5(var_pass)
      into var_user_role;

      -- See the corresponding enum in the client
                if var_user_role = 'amministratore' then
                        set var_role = 1;
                elseif var_user_role = 'conducente' then
```

```
                set var_role = 2;
        elseif var_user_role = 'passeggero' then
                set var_role = 3;
        else
                set var_role = 4;
        end if;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Emetti_abbonamenti
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Emetti_abbonamenti`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Emetti_abbonamenti` ()
BEGIN
    insert into `Abbonamento`() values ();
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Emetti_biglietto
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Emetti_biglietto`;
```

```
DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Emetti_biglietto` ()
BEGIN
	insert into `Biglietto` (Stato) values ("0");
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure Aggiungi_utente
-- -----------------------------------------------------

USE `SistemaTrasportoPubblico3`;
DROP procedure IF EXISTS `SistemaTrasportoPubblico3`.`Aggiungi_utente`;

DELIMITER $$
USE `SistemaTrasportoPubblico3`$$
CREATE PROCEDURE `Aggiungi_utente` (IN var_username VARCHAR(45), IN var_password VARCHAR(45), IN var_ruolo VARCHAR(16))
BEGIN
	set transaction isolation level repeatable read;
  start transaction;
	insert into `Utente` (`Username`, `Password`, `Ruolo`) values (var_username, MD5(var_password), var_ruolo);
  commit;
END$$

DELIMITER ;
SET SQL_MODE = '';
DROP USER IF EXISTS Amministratore;
```

```
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'Amministratore' IDENTIFIED BY 'amministratore';


SET SQL_MODE = '';

DROP USER IF EXISTS Conducente;

SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'Conducente' IDENTIFIED BY 'conducente';


SET SQL_MODE = '';

DROP USER IF EXISTS Passeggero;

SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'Passeggero' IDENTIFIED BY 'passeggero';



SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;


-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Veicolo`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Veicolo` (`Matricola`, `Data_acquisto`) VALUES (1, '2006-5-4');

INSERT INTO `SistemaTrasportoPubblico3`.`Veicolo` (`Matricola`, `Data_acquisto`) VALUES (2, '2005-8-4');

INSERT INTO `SistemaTrasportoPubblico3`.`Veicolo` (`Matricola`, `Data_acquisto`) VALUES (3, '2000-6-9');
```

```
COMMIT;


-- -----------------------------------------------------
-- Data for table `SistemaTrasportoPubblico3`.`WayPoint`
-- -----------------------------------------------------
START TRANSACTION;
USE `SistemaTrasportoPubblico3`;
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('21.568746', '12.457415', NULL, 1, NULL);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('15.265498', '52.548745', '16:00:00', NULL, 1);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('12.568547', '85.698547', NULL, NULL, NULL);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('52.567895', '43.357586', NULL, 2, NULL);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('98.653242', '54.563245', NULL, NULL, NULL);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('10.254856', '43.245874', '13:00:00', NULL, 2);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('11.256547', '52.632412', NULL, NULL, NULL);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('14.495850', '35.586837', NULL, 3, NULL);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('43.543543', '54.543545', '10:00:00', NULL, 3);
INSERT INTO `SistemaTrasportoPubblico3`.`WayPoint` (`Latitudine`, `Longitudine`, `OrarioPartenze`, `Cod_fermata`, `Cod_capolinea`) VALUES ('54.543545', '76.986747', '19:00:00', NULL, 4);
```

```
COMMIT;


-- -----------------------------------------------------
-- Data for table `SistemaTrasportoPubblico3`.`Tratta`
-- -----------------------------------------------------
START TRANSACTION;
USE `SistemaTrasportoPubblico3`;
INSERT INTO `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`, `Latitudine_inziale`, `Longitudine_inziale`, `Latitudine_finale`, `Longitudine_finale`) VALUES (1, '15.265498', '52.548745', '10.254856', '43.245874');
INSERT INTO `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`, `Latitudine_inziale`, `Longitudine_inziale`, `Latitudine_finale`, `Longitudine_finale`) VALUES (2, '10.254856', '43.245874', '15.265498', '52.548745');
INSERT INTO `SistemaTrasportoPubblico3`.`Tratta` (`idTratta`, `Latitudine_inziale`, `Longitudine_inziale`, `Latitudine_finale`, `Longitudine_finale`) VALUES (3, '43.543543', '54.543545', '54.543545', '76.986747');
COMMIT;


-- -----------------------------------------------------
-- Data for table `SistemaTrasportoPubblico3`.`Utente`
-- -----------------------------------------------------
START TRANSACTION;
USE `SistemaTrasportoPubblico3`;
INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('mauro', '0c88028bf3aa6a6a143ed846f2be1ea4', 'AMMINISTRATORE');
INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('mattia', '0c88028bf3aa6a6a143ed846f2be1ea4', 'CONDUCENTE');
INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('giovanni', '0c88028bf3aa6a6a143ed846f2be1ea4', 'CONDUCENTE');
INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('antonio', '0c88028bf3aa6a6a143ed846f2be1ea4', 'PASSEGGERO');
```

```
INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('franco', '0c88028bf3aa6a6a143ed846f2be1ea4', 'PASSEGGERO');

INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('lauro', '0c88028bf3aa6a6a143ed846f2be1ea4', 'CONDUCENTE');

INSERT INTO `SistemaTrasportoPubblico3`.`Utente` (`Username`, `Password`, `Ruolo`) VALUES ('ivan', '0c88028bf3aa6a6a143ed846f2be1ea4', 'CONDUCENTE');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Conducente`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Conducente` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Luogo_nascita`, `Scadenza_patente`, `Numero_patente`, `Utente_Username`) VALUES ('PLMVNI128H32', 'mattia', 'Palmieri', '1998-10-28', 'Tivoli', '2025-10-10', 'PTIGH576', 'mattia');

INSERT INTO `SistemaTrasportoPubblico3`.`Conducente` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Luogo_nascita`, `Scadenza_patente`, `Numero_patente`, `Utente_Username`) VALUES ('MDJEBN85JFJ3', 'giovanni', 'Rossi', '1996-10-12', 'Milano', '2027-10-12', 'PODED597', 'giovanni');

INSERT INTO `SistemaTrasportoPubblico3`.`Conducente` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Luogo_nascita`, `Scadenza_patente`, `Numero_patente`, `Utente_Username`) VALUES ('KDJEJCNF7RM1', 'lauro', 'verdi', '1990-10-12', 'Roma', '2027-10-12', 'NDSJXMEU', 'lauro');

INSERT INTO `SistemaTrasportoPubblico3`.`Conducente` (`CF`, `Nome`, `Cognome`, `Data_nascita`, `Luogo_nascita`, `Scadenza_patente`, `Numero_patente`, `Utente_Username`) VALUES ('NSNSJWU73ND', 'ivan', 'Palma', '1978-10-12', 'Tivoli', '2025-10-10', 'NDJQIU32', 'ivan');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Tratta_effettiva`
```

```
-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Tratta_effettiva` (`Data`, `Tratta`, `Conducente`, `Veicolo`) VALUES ('2020-02-20', 1, 'PLMVNI128H32', 1);

INSERT INTO `SistemaTrasportoPubblico3`.`Tratta_effettiva` (`Data`, `Tratta`, `Conducente`, `Veicolo`) VALUES ('2020-02-20', 2, 'MDJEBN85JFJ3', 2);


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Biglietto`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Biglietto` (`idBiglietto`, `Stato`, `Data`, `Tratta`, `Veicolo`) VALUES (1, 0, NULL, NULL, NULL);

INSERT INTO `SistemaTrasportoPubblico3`.`Biglietto` (`idBiglietto`, `Stato`, `Data`, `Tratta`, `Veicolo`) VALUES (2, 0, NULL, NULL, NULL);

INSERT INTO `SistemaTrasportoPubblico3`.`Biglietto` (`idBiglietto`, `Stato`, `Data`, `Tratta`, `Veicolo`) VALUES (3, 0, NULL, NULL, NULL);

INSERT INTO `SistemaTrasportoPubblico3`.`Biglietto` (`idBiglietto`, `Stato`, `Data`, `Tratta`, `Veicolo`) VALUES (4, 0, NULL, NULL, NULL);


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Abbonamento`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;
```

```
INSERT INTO `SistemaTrasportoPubblico3`.`Abbonamento` (`idAbbonamento`, `Ultimo_utilizzo`) VALUES (1, NULL);

INSERT INTO `SistemaTrasportoPubblico3`.`Abbonamento` (`idAbbonamento`, `Ultimo_utilizzo`) VALUES (2, NULL);

INSERT INTO `SistemaTrasportoPubblico3`.`Abbonamento` (`idAbbonamento`, `Ultimo_utilizzo`) VALUES (3, NULL);


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Turno`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('8:00:00', '16:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('9:00:00', '17:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('10:00:00', '18:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('11:00:00', '19:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('12:00:00', '20:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('13:00:00', '21:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('14:00:00', '22:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('15:00:00', '23:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('16:00:00', '1:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('17:00:00', '2:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('18:00:00', '3:00:00');
```

```
INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('19:00:00', '4:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('20:00:00', '5:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('21:00:00', '6:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('22:00:00', '7:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('23:00:00', '8:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('1:00:00', '9:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('2:00:00', '10:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('3:00:00', '11:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('4:00:00', '12:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('5:00:00', '13:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('6:00:00', '14:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno` (`Ora_inizio`, `Ora_fine`) VALUES ('7:00:00', '15:00:00');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Manutenzione`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Cambio dell\'olio');

INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Tagliando');
```

```
INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Cambio pasticche dei freni');

INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Cambio pistone');

INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Modifiche estetiche');

INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Revisione');

INSERT INTO `SistemaTrasportoPubblico3`.`Manutenzione` (`Tipo_manutenzione`) VALUES ('Cambio ruota');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Turno_effettivo`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Turno_effettivo` (`Data_turno`, `Conducente`, `Ora_inizio`, `Ora_fine`) VALUES ('2020-02-18', 'PLMVNI128H32', '12:00:00', '20:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Turno_effettivo` (`Data_turno`, `Conducente`, `Ora_inizio`, `Ora_fine`) VALUES ('2020-02-18', 'MDJEBN85JFJ3', '12:00:00', '20:00:00');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`In_programma`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`In_programma` (`Veicolo`, `Tratta`, `Numero`, `Data`) VALUES (1, 1, 1, '2020-02-18');
```

```
INSERT INTO `SistemaTrasportoPubblico3`.`In_programma` (`Veicolo`, `Tratta`, `Numero`, `Data`) VALUES (1, 2, 2, '2020-02-18');

INSERT INTO `SistemaTrasportoPubblico3`.`In_programma` (`Veicolo`, `Tratta`, `Numero`, `Data`) VALUES (2, 2, 1, '2020-02-18');

INSERT INTO `SistemaTrasportoPubblico3`.`In_programma` (`Veicolo`, `Tratta`, `Numero`, `Data`) VALUES (2, 3, 2, '2020-02-18');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Passato`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Passato` (`Data`, `Tratta`, `Veicolo`, `WayPoint_Latitudine`, `WayPoint_Longitudine`, `Orario`) VALUES ('2020-02-20', 1, 1, '21.568746', '12.457415', '16:00:00');

INSERT INTO `SistemaTrasportoPubblico3`.`Passato` (`Data`, `Tratta`, `Veicolo`, `WayPoint_Latitudine`, `WayPoint_Longitudine`, `Orario`) VALUES ('2020-02-20', 1, 1, '15.265498', '52.548745', '16:25:32');


COMMIT;



-- -----------------------------------------------------

-- Data for table `SistemaTrasportoPubblico3`.`Fermata`

-- -----------------------------------------------------

START TRANSACTION;

USE `SistemaTrasportoPubblico3`;

INSERT INTO `SistemaTrasportoPubblico3`.`Fermata` (`idTratta`, `Latitudine`, `Longitudine`, `Numero_fermata`) VALUES (1, '21.568746', '12.457415', 1);

INSERT INTO `SistemaTrasportoPubblico3`.`Fermata` (`idTratta`, `Latitudine`, `Longitudine`, `Numero_fermata`) VALUES (1, '52.567895', '43.357586', 2);
```

COMMIT;

Codice del Front-End

Utils.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

void print_stmt_error(MYSQL_STMT* stmt, char* message)
{
	fprintf(stderr, "%s\n", message);
	if (stmt != NULL) {
		fprintf(stderr, "Error %u (%s): %s\n",
			mysql_stmt_errno(stmt),
			mysql_stmt_sqlstate(stmt),
			mysql_stmt_error(stmt));
	}
}


void print_error(MYSQL* conn, char* message)
{
	fprintf(stderr, "%s\n", message);
	if (conn != NULL) {
#if MYSQL_VERSION_ID >= 40101
		fprintf(stderr, "Error %u (%s): %s\n",
			mysql_errno(conn), mysql_sqlstate(conn), mysql_error(conn));
#else
		fprintf(stderr, "Error %u: %s\n",
			mysql_errno(conn), mysql_error(conn));
#endif
	}
}

bool setup_prepared_stmt(MYSQL_STMT** stmt, char* statement, MYSQL* conn)
{
	my_bool update_length = true;

	*stmt = mysql_stmt_init(conn);
	if (*stmt == NULL)
	{
		print_error(conn, "Could not initialize statement handler");
		return false;
	}

	if (mysql_stmt_prepare(*stmt, statement, strlen(statement)) != 0) {
```

```
                print_stmt_error(*stmt, "Could not prepare statement");
                return false;
        }

        mysql_stmt_attr_set(*stmt, STMT_ATTR_UPDATE_MAX_LENGTH, &update_length);

        return true;
}

void finish_with_error(MYSQL* conn, char* message)
{
        print_error(conn, message);
        mysql_close(conn);
        exit(EXIT_FAILURE);
}

void finish_with_stmt_error(MYSQL* conn, MYSQL_STMT* stmt, char* message, bool
close_stmt)
{
        print_stmt_error(stmt, message);
        if (close_stmt)         mysql_stmt_close(stmt);
        mysql_close(conn);
        exit(EXIT_FAILURE);
}

static void print_dashes(MYSQL_RES* res_set)
{
        MYSQL_FIELD* field;
        unsigned int i, j;

        mysql_field_seek(res_set, 0);
        putchar('+');
        for (i = 0; i < mysql_num_fields(res_set); i++) {
                field = mysql_fetch_field(res_set);
                for (j = 0; j < field->max_length + 2; j++)
                        putchar('-');
                putchar('+');
        }
        putchar('\n');
}

static void dump_result_set_header(MYSQL_RES* res_set)
{
        MYSQL_FIELD* field;
        unsigned long col_len;
        unsigned int i;

        /* determine column display widths -- requires result set to be */
        /* generated with mysql_store_result(), not mysql_use_result() */
```

```
	mysql_field_seek(res_set, 0);

	for (i = 0; i < mysql_num_fields(res_set); i++) {
		field = mysql_fetch_field(res_set);
		col_len = strlen(field->name);

		if (col_len < field->max_length)
			col_len = field->max_length;
		if (col_len < 4 && !IS_NOT_NULL(field->flags))
			col_len = 4; /* 4 = length of the word "NULL" */
		field->max_length = col_len; /* reset column info */
	}

	print_dashes(res_set);
	putchar('|');
	mysql_field_seek(res_set, 0);
	for (i = 0; i < mysql_num_fields(res_set); i++) {
		field = mysql_fetch_field(res_set);
		printf(" %-*s |", (int)field->max_length, field->name);
	}
	putchar('\n');

	print_dashes(res_set);
}

void dump_result_set(MYSQL* conn, MYSQL_STMT* stmt, char* title)
{
	int i;
	int status;
	int num_fields;       /* number of columns in result */
	MYSQL_FIELD* fields;  /* for result set metadata */
	MYSQL_BIND* rs_bind;  /* for output buffers */
	MYSQL_RES* rs_metadata;
	MYSQL_TIME* date;
	size_t attr_size;

	/* Prefetch the whole result set. This in conjunction with
	 * STMT_ATTR_UPDATE_MAX_LENGTH set in `setup_prepared_stmt`
	 * updates the result set metadata which are fetched in this
	 * function, to allow to compute the actual max length of
	 * the columns.
	 */
	if (mysql_stmt_store_result(stmt)) {
		fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
		fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
		exit(0);
	}

	/* the column count is > 0 if there is a result set */
	/* 0 if the result is only the final status packet */
```

```
	num_fields = mysql_stmt_field_count(stmt);

	if (num_fields > 0) {
		/* there is a result set to fetch */
		printf("%s\n", title);

		if ((rs_metadata = mysql_stmt_result_metadata(stmt)) == NULL) {
			finish_with_stmt_error(conn, stmt, "Unable to retrieve result metadata\n",
true);
		}
		/*QUI*/
		dump_result_set_header(rs_metadata);

		fields = mysql_fetch_fields(rs_metadata);

		rs_bind = (MYSQL_BIND*)malloc(sizeof(MYSQL_BIND) * num_fields);
		if (!rs_bind) {
			finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n", true);
		}
		memset(rs_bind, 0, sizeof(MYSQL_BIND) * num_fields);

		/* set up and bind result set output buffers */
		for (i = 0; i < num_fields; ++i) {

			// Properly size the parameter buffer
			switch (fields[i].type) {
			case MYSQL_TYPE_DATE:
			case MYSQL_TYPE_TIMESTAMP:
			case MYSQL_TYPE_DATETIME:
			case MYSQL_TYPE_TIME:
				attr_size = sizeof(MYSQL_TIME);
				break;
			case MYSQL_TYPE_FLOAT:
				attr_size = sizeof(float);
				break;
			case MYSQL_TYPE_DOUBLE:
				attr_size = sizeof(double);
				break;
			case MYSQL_TYPE_TINY:
				attr_size = sizeof(signed char);
				break;
			case MYSQL_TYPE_SHORT:
			case MYSQL_TYPE_YEAR:
				attr_size = sizeof(short int);
				break;
			case MYSQL_TYPE_LONG:
			case MYSQL_TYPE_INT24:
				attr_size = sizeof(int);
				break;
			case MYSQL_TYPE_LONGLONG:
```

```
                        attr_size = sizeof(int);
                        break;
                default:
                        attr_size = fields[i].max_length;
                        break;
                }

                // Setup the binding for the current parameter
                rs_bind[i].buffer_type = fields[i].type;
                rs_bind[i].buffer = malloc(attr_size + 1);
                rs_bind[i].buffer_length = attr_size + 1;

                if (rs_bind[i].buffer == NULL) {
                        finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n",
true);

                }
        }

        if (mysql_stmt_bind_result(stmt, rs_bind)) {
                finish_with_stmt_error(conn, stmt, "Unable to bind output parameters\n",
true);
        }

        /* fetch and display result set rows */
        while (true) {
                status = mysql_stmt_fetch(stmt);

                if (status == 1 || status == MYSQL_NO_DATA)
                        break;

                putchar('|');

                for (i = 0; i < num_fields; i++) {

                        if (rs_bind[i].is_null_value) {
                                printf(" %-*s |", (int)fields[i].max_length, "NULL");
                                continue;
                        }

                        switch (rs_bind[i].buffer_type) {

                        case MYSQL_TYPE_VAR_STRING:
                        case MYSQL_TYPE_DATETIME:
                                printf(" %-*s |", (int)fields[i].max_length,
(char*)rs_bind[i].buffer);
                                break;

                        case MYSQL_TYPE_DATE:
                        case MYSQL_TYPE_TIMESTAMP:
```

```
                                        date = (MYSQL_TIME*)rs_bind[i].buffer;
                                        printf(" %d-%02d-%02d |", date->year, date->month, date-
>day);
                                        break;

                                case MYSQL_TYPE_STRING:
                                        printf(" %-*s |", (int)fields[i].max_length,
(char*)rs_bind[i].buffer);
                                        break;

                                case MYSQL_TYPE_FLOAT:
                                case MYSQL_TYPE_DOUBLE:
                                        printf(" %.02f |", *(float*)rs_bind[i].buffer);
                                        break;

                                case MYSQL_TYPE_LONG:
                                case MYSQL_TYPE_SHORT:
                                case MYSQL_TYPE_TINY:
                                        printf(" %-*d |", (int)fields[i].max_length,
*(int*)rs_bind[i].buffer);
                                        break;

                                case MYSQL_TYPE_NEWDECIMAL:
                                        printf(" %-*.02lf |", (int)fields[i].max_length,
*(float*)rs_bind[i].buffer);
                                        break;

                                default:
                                        printf("ERROR: Unhandled type (%d)\n",
rs_bind[i].buffer_type);
                                        abort();
                                }
                        }
                        putchar('\n');
                        print_dashes(rs_metadata);
                }

                mysql_free_result(rs_metadata); /* free metadata */

                /* free output buffers */
                for (i = 0; i < num_fields; i++) {
                        free(rs_bind[i].buffer);
                }
                free(rs_bind);
        }
}
```

Passeggero.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <math.h>
#include "defines.h"
#define MAX_DEGREES 128

struct average_grades {
	int latitudineI;
	int longitudineI;
	double avg;
};
float ftemp;
float ftemp2;

static void Timbra_biglietto(MYSQL* conn) {
	MYSQL_STMT* TimbraB;
	MYSQL_BIND param[3];
	int biglietto;
	int veicolo;
	int tratta;
	printf("Inserisci il numero del biglietto : ");
	scanf_s("%d", &biglietto);
	printf("Su quale veicolo ti trovi : ");
	scanf_s("%d", &veicolo);
	printf("Quale tratta stai percorrendo : ");
	scanf_s("%d", &tratta);

	if (!setup_prepared_stmt(&TimbraB, "call Timbra_biglietto(?, ?, ?)", conn)) {
		print_stmt_error(TimbraB, "Unable to initialize login statement\n");
	}

	// Prepare parameters
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[0].buffer = &biglietto;
	param[0].buffer_length = sizeof(biglietto);

	param[1].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[1].buffer = &tratta;
	param[1].buffer_length = sizeof(tratta);

	param[2].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[2].buffer = &veicolo;
	param[2].buffer_length = sizeof(veicolo);

	if (mysql_stmt_bind_param(TimbraB, param) != 0) {
		finish_with_stmt_error(conn, TimbraB, "Could not bind parameters for career report\
n", true);
	}
	// Run procedure
```

```
	if (mysql_stmt_execute(TimbraB) != 0) {
		print_stmt_error(TimbraB, "An error occurred while retrieving the career report.");
		goto out;
	}
	printf("Hai tibrato il bigletto [%d]\n", biglietto);
	system("pause");
	mysql_stmt_close(TimbraB);
	return;


out:
	mysql_stmt_close(TimbraB);
}
static void Convalida_abbonamento(MYSQL* conn) {
	MYSQL_STMT* TimbraA;
	MYSQL_BIND param[3];
	int abbonamento;
	int veicolo;
	int tratta;
	printf("Inserisci il numero dell'abbonamento : ");
	scanf_s("%d", &abbonamento);
	printf("Su quale veicolo ti trovi : ");
	scanf_s("%d", &veicolo);
	printf("Quale tratta stai percorrendo : ");
	scanf_s("%d", &tratta);

	if (!setup_prepared_stmt(&TimbraA, "call Convalida_abbonamento(?, ?, ?)", conn)) {
		print_stmt_error(TimbraA, "Unable to initialize login statement\n");
	}

	// Prepare parameters
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[0].buffer = &abbonamento;
	param[0].buffer_length = sizeof(abbonamento);

	param[1].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[1].buffer = &tratta;
	param[1].buffer_length = sizeof(tratta);

	param[2].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[2].buffer = &veicolo;
	param[2].buffer_length = sizeof(veicolo);

	if (mysql_stmt_bind_param(TimbraA, param) != 0) {
		finish_with_stmt_error(conn, TimbraA, "Could not bind parameters for career report\
n", true);
	}
	// Run procedure
```

```
	if (mysql_stmt_execute(TimbraA) != 0) {
		print_stmt_error(TimbraA, "An error occurred while retrieving the career report.");
		goto out;
	}
	printf("Hai convalidato l'abbonamento [%d]\n", abbonamento);
	system("pause");
	mysql_stmt_close(TimbraA);
	return;


out:
	mysql_stmt_close(TimbraA);
}
static size_t parse_avgs(MYSQL* conn, MYSQL_STMT* stmt, struct average_grades** ret)
{

	int status;
	size_t row = 0;
	MYSQL_BIND param[3];
	my_bool is_null;
	double avg;
	char latitudine[12];
	char longitudine[12];
	int veicolo;

	if (mysql_stmt_store_result(stmt)) {
		fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
		fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
		exit(0);
	}

	*ret = malloc(mysql_stmt_num_rows(stmt) * sizeof(struct average_grades));

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = latitudine;
	param[0].buffer_length = 12;

	/*Devo restiturilo nella store procedure*/
	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = longitudine;
	param[1].buffer_length = 12;

	param[2].buffer_type = MYSQL_TYPE_LONG;
	param[2].buffer = &veicolo;
	param[2].buffer_length = sizeof(veicolo);

	if (mysql_stmt_bind_result(stmt, param)) {
		finish_with_stmt_error(conn, stmt, "Unable to bind column parameters\n", true);
```

```
	}
	int i = 1;
	/* assemble course general information */
	while (true) {
		status = mysql_stmt_fetch(stmt);

		if (status == 1 || status == MYSQL_NO_DATA)
			break;

		(*ret)[row].latitudineI= latitudine;
		(*ret)[row].longitudineI = longitudine;
		printf_s("La tua fermata si trova qui:\nLatitudine : %s	Longitudine : %s\n",
latitudine, longitudine);

		/*VALORE DELLA FERMATA DOVE SI TROVANO*/
		 ftemp = atof(latitudine);
		 ftemp2 = atof(longitudine);
		row++;
	}

	return row;
}
static void Calcola_distanza_veicolo(MYSQL* conn) {
	MYSQL_STMT* Distanza;
	int status;
	MYSQL_BIND param[1];
	bool first = true;
	struct average_grades* avgs = NULL; /*DEVO TOGLIERE IL NULL MA MI DA
ERRORE*/
	size_t longitudineI = 0;
	size_t latitudineI = 0;
	int Tratta;
	char header[512];

	// Prepare stored procedure call
	if (!setup_prepared_stmt(&Distanza, "call Calcola_distanza_veicolo(?)", conn)) {
		finish_with_stmt_error(conn, Distanza, "Unable to initialize career report statement\
n", false);
	}

	printf("A quale fermata ti trovi ? \n");
	printf("SCELTA: ");
	scanf_s("%d", &Tratta);
	printf("\n\n\n");


	// Prepare parameters
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_LONG;  //IN
```

```
	param[0].buffer = &Tratta;
	param[0].buffer_length = sizeof(Tratta);



	if (mysql_stmt_bind_param(Distanza, param) != 0) {
		finish_with_stmt_error(conn, Distanza, "Could not bind parameters for career report\
n", true);
	}

	// Run procedure
	if (mysql_stmt_execute(Distanza) != 0) {
		print_stmt_error(Distanza, "An error occurred while retrieving the career report.");
		goto out;
	}

	// We have multiple result sets here!
	do {
		// Skip OUT variables (although they are not present in the procedure...)
		if (conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		if (first) {
			parse_avgs(conn, Distanza, &avgs);
			first = false;
		}
		else {
			sprintf_s(header, 512, "\nI veicoli si trovano: " ,avgs[latitudineI].latitudineI,
avgs[longitudineI].longitudineI);
			dump_result_set(conn, Distanza, header);

			latitudineI++;
			longitudineI++;

		}
		// more results? -1 = no, >0 = error, 0 = yes (keep looking)
	next:
		status = mysql_stmt_next_result(Distanza);
		if (status > 0)
			finish_with_stmt_error(conn, Distanza, "Unexpected condition", true);

	} while (status == 0);

out:
	mysql_stmt_close(Distanza);
}

void run_as_passeggero(MYSQL* conn)
{
```

```
        int s;
        int i = 0;
        int numero;
        while (true) {

                printf("----------------------------*** Cosa vuoi fare? ***----------------------------\n\
n");
                printf("1) Timbra un biglietto\n");
                printf("2) Convalida un abbonamento\n");
                printf("3) Calcola distanza veicolo\n");
                printf("4) Logout\n");
                printf("SCELTA: ");
                scanf_s("%i", &numero);
                switch (numero)
                {
                case 1:
                        printf("-------------------------Timbra un
biglietto-----------------------------------------\n");
                        Timbra_biglietto(conn);
                        break;
                case 2:
                        printf("-----------------------------------------Convalida un
abbonamento-----------------------------------------\n");
                        Convalida_abbonamento(conn);
                        break;
                case 3:
                        printf("-----------------------------------------Calcola distanza
veicolo-----------------------------------------\n");
                        Calcola_distanza_veicolo(conn);
                        float lat;
                        float lon;
                        printf("\n\n Inserisci i dati del veicolo di interesse \n");
                        printf("Latitudine: ");
                        scanf_s("%f", &lat);
                        printf("Longitudine: ");
                        scanf_s("%f", &lon);
                        float risultato;
                        risultato = 2 * 6371 * asin(sqrt(((sin(lat - ftemp) / 2) * (sin(lat - ftemp) / 2)) +
cos(ftemp) * cos(lat) * (sin((lon - ftemp2) / 2) * sin((lon - ftemp2) / 2))));
                        printf("\nIl veicolo si trova a %f Km di distanza\n risultato ");
                        break;
                case 4:
                        printf("---------------------------------------
Logout---------------------------------------\n");
                        return;

                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        return;
                }
```

```
        }
}
```

Amministratore.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"
#define MAX_DEGREES 128

struct average_grades {
        char CF[16];
        double avg;
};
struct average_gradess {
        int Matricola;
        double avg;
};

static void Assegna_turno_al_conducente(MYSQL* conn) {
        MYSQL_STMT* turno;
        MYSQL_BIND param[4];
        MYSQL_TIME  data;
        char inizio[16];
        char fine[16];
        char Conducente_CF[16];

        printf("Inserisci il CF dell'guidatore : ");
```

```
        scanf_s("%s", Conducente_CF, 16);
        printf("Inserisci il giorno\n");
        printf("Anno : ");
        scanf_s("%d", &data.year, 4);
        printf("Mese : ");
        scanf_s("%d", &data.month, 2);
        printf("Giorno : ");
        scanf_s("%d", &data.day, 2);
        printf("La differenza tra l'orario di inizio e quello di fine deve essere precisamente di 8 ore ! \
n");
        printf("Dalle :   (Inserire nel formato hh:mm:)\n");
        scanf_s("%s", &inizio);
        printf("Alle :   (Inserire nel formato hh:mm)\n");
        scanf_s("%s", &fine);



        if (!setup_prepared_stmt(&turno, "call Assegna_turno_al_conducente(?, ?, ?, ?)", conn)) {
                print_stmt_error(turno, "Unable to initialize login statement\n");
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
        param[0].buffer = Conducente_CF;
        param[0].buffer_length = strlen(Conducente_CF);

        param[1].buffer_type = MYSQL_TYPE_DATE; //IN
        param[1].buffer = (char*)&data;
        param[1].buffer_length = sizeof(data);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING; //IN
```

```
	param[2].buffer = inizio;
	param[2].buffer_length = strlen(inizio);

	param[3].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
	param[3].buffer = fine;
	param[3].buffer_length = strlen(fine);



	if (mysql_stmt_bind_param(turno, param) != 0) {
		finish_with_stmt_error(conn, turno, "Could not bind parameters for career report\n", true);
	}
	// Run procedure
	if (mysql_stmt_execute(turno) != 0) {
		print_stmt_error(turno, "An error occurred while retrieving the career report.");
		goto out;
	}

	printf("Turno assegnato con successo! \n");
	system("pause");
	mysql_stmt_close(turno);
	return;



out:
	mysql_stmt_close(turno);

}
static void Assegna_veicolo_alla_tratta(MYSQL* conn) {
	MYSQL_STMT* veicl_stmt;
	MYSQL_BIND param[4];
	MYSQL_TIME  ts;
```

```
char Conducente[16];
int Veicolo;
int Tratta;
printf("Conducente : ");
scanf_s("%s", Conducente, 16);
printf("Veicolo : ");
scanf_s("%d", &Veicolo);
printf("Tratta : ");
scanf_s("%d",&Tratta);

printf("Inserisci la data di percorrenza\n");
printf("Anno : ");
scanf_s("%d", &ts.year, 4);
printf("Mese : ");
scanf_s("%d", &ts.month, 2);
printf("Giorno : ");
scanf_s("%d", &ts.day, 2);


if (!setup_prepared_stmt(&veicl_stmt, "call Assegna_veicolo_alla_tratta(?, ?, ?, ?)", conn)) {
        print_stmt_error(veicl_stmt, "Unable to initialize login statement\n");
}

// Prepare parameters
memset(param, 0, sizeof(param));



param[0].buffer_type = MYSQL_TYPE_LONG;  //IN
param[0].buffer = &Veicolo;
param[0].buffer_length = sizeof(Veicolo);
```

```
	param[1].buffer_type = MYSQL_TYPE_LONG;  //IN
	param[1].buffer = &Tratta;
	param[1].buffer_length = sizeof(Tratta);

	param[2].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
	param[2].buffer = Conducente;
	param[2].buffer_length = strlen(Conducente);

	param[3].buffer_type = MYSQL_TYPE_DATE;  //IN  DATAAA
	param[3].buffer = (char*)&ts;
	param[3].buffer_length = sizeof(ts);

	if (mysql_stmt_bind_param(veicl_stmt, param) != 0) {
		finish_with_stmt_error(conn, veicl_stmt, "Could not bind parameters for career report\n", true);
	}
	// Run procedure
	if (mysql_stmt_execute(veicl_stmt) != 0) {
		print_stmt_error(veicl_stmt, "An error occurred while retrieving the career report.");
		goto out;
	}

	printf("Hai assegnato al conducente : CF %s la tratta %d con il veicolo %d il giorno %d:%d:%d", Conducente, Tratta, Veicolo, ts.year,ts.month,ts.day);
	system("pause");
	mysql_stmt_close(veicl_stmt);
	return;


out:
	mysql_stmt_close(veicl_stmt);

}
```

```
static void Assumi_conducente(MYSQL* conn) {
	MYSQL_STMT* Assum_cond;
	MYSQL_BIND param[8];
	MYSQL_TIME  ts;
	MYSQL_TIME  tl;
	char CF[16];
	char nome[45];
	char username[45];
	char cognome[45];
	char patente[10];
	char nascita[16];
	char scadenza[16];
	char luogo_nascita[45];

	printf("Inserisci l'username dell'utente da assumere come conducente : ");
	scanf_s("%s", username,45);
	printf("Inserisci il codice fiscale : ");
	scanf_s("%s", CF,16);
	printf("Inserisci il nome : ");
	scanf_s("%s", nome,45);
	printf("Inserisci il cognome : ");
	scanf_s("%s", cognome,45);
	printf("Inserisci la data di nascita\n");
	printf("Anno : ");
	scanf_s("%d", &ts.year,4);
	printf("Mese : ");
	scanf_s("%d", &ts.month,2);
	printf("Giorno : ");
	scanf_s("%d", &ts.day,2);
	printf("Inserisci il luogo di nascita : ");
	scanf_s("%s", luogo_nascita,45);
	printf("Inserisci la data di scadenza della patente \n");
```

```
printf("Anno : ");
scanf_s("%d", &tl.year,4);
printf("Mese : ");
scanf_s("%d", &tl.month,2);
printf("Giorno : ");
scanf_s("%d", &tl.day,2);
printf("Inserisci il numero di patente: ");
scanf_s("%s", patente,10);



if (!setup_prepared_stmt(&Assum_cond, "call Assumi_conducente(?, ?, ?, ?, ?, ?, ?, ?)", conn)) {
        print_stmt_error(Assum_cond, "Unable to initialize login statement\n");
}

// Prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[0].buffer = CF;
param[0].buffer_length = strlen(CF);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[1].buffer = nome;
param[1].buffer_length = strlen(nome);

param[2].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[2].buffer = cognome;
param[2].buffer_length = strlen(cognome);

param[3].buffer_type = MYSQL_TYPE_DATE;  //IN  DATAAA
param[3].buffer = (char*)&ts;
```

```
param[3].buffer_length = sizeof(ts);

param[4].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[4].buffer = luogo_nascita;
param[4].buffer_length = strlen(luogo_nascita);

param[5].buffer_type = MYSQL_TYPE_DATE;  //IN  DATAAA
param[5].buffer = (char*)&tl;
param[5].buffer_length = sizeof(tl);

param[6].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[6].buffer = patente;
param[6].buffer_length = strlen(patente);

param[7].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[7].buffer = username;
param[7].buffer_length = strlen(username);

if (mysql_stmt_bind_param(Assum_cond, param) != 0) {
        finish_with_stmt_error(conn, Assum_cond, "Could not bind parameters for career report\n", true);
}
// Run procedure
if (mysql_stmt_execute(Assum_cond) != 0) {
        print_stmt_error(Assum_cond, "An error occurred while retrieving the career report.");
        goto out;
}

printf("Hai aggiunto : CF %s, Nome: %s, Cognome: %s con l'username :%s", CF, nome,cognome,username);
system("pause");
mysql_stmt_close(Assum_cond);
```

```
        return;


out:
        mysql_stmt_close(Assum_cond);

}
static void Cambia_conducente_turno(MYSQL* conn) {
        MYSQL_STMT* changeturno;
        MYSQL_BIND param[4];
        MYSQL_TIME  data;
        char inizio[16];
        char fine[16];
        char Conducente_CF[16];
        char Conducente_CF2[16];

        printf("Inserisci il CF dell'guidatore da cambiare : ");
        scanf_s("%s", Conducente_CF, 16);

        printf("Inserisci il CF dell'guidatore che lo sostituira' : ");
        scanf_s("%s", Conducente_CF2, 16);
        printf("Inserisci il giorno\n");
        printf("Anno : ");
        scanf_s("%d", &data.year, 4);
        printf("Mese : ");
        scanf_s("%d", &data.month, 2);
        printf("Giorno : ");
        scanf_s("%d", &data.day, 2);



        if (!setup_prepared_stmt(&changeturno, "call Cambia_conducente_turno(?, ?, ?)", conn)) {
```

```
        print_stmt_error(changeturno, "Unable to initialize login statement\n");
    }

    // Prepare parameters
    memset(param, 0, sizeof(param));

    param[0].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
    param[0].buffer = Conducente_CF;
    param[0].buffer_length = strlen(Conducente_CF);

    param[1].buffer_type = MYSQL_TYPE_DATE;  //IN
    param[1].buffer = (char*)&data;
    param[1].buffer_length = sizeof(data);

    param[2].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
    param[2].buffer = Conducente_CF2;
    param[2].buffer_length = strlen(Conducente_CF2);



    if (mysql_stmt_bind_param(changeturno, param) != 0) {
        finish_with_stmt_error(conn, changeturno, "Could not bind parameters for career report\n", true);
    }
    // Run procedure
    if (mysql_stmt_execute(changeturno) != 0) {
        print_stmt_error(changeturno, "An error occurred while retrieving the career report.");
        goto out;
    }

    printf("Turno cambiato con successo! \n");
    system("pause");
```

```
	mysql_stmt_close(changeturno);
	return;



out:
	mysql_stmt_close(changeturno);

}
static size_t parse_avgs(MYSQL* conn, MYSQL_STMT* stmt, struct average_grades** ret)
{

	int status;
	size_t row = 0;
	MYSQL_BIND param[2];
	my_bool is_null;
	double avg;
	char CF[16];

	if (mysql_stmt_store_result(stmt)) {
		fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
		fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
		exit(0);
	}

	*ret = malloc(mysql_stmt_num_rows(stmt) * sizeof(struct average_grades));

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = CF;
	param[0].buffer_length = 16;
```

```
	/*Devo restiturilo nella store procedure*/
	param[1].buffer_type = MYSQL_TYPE_DOUBLE;
	param[1].buffer = &avg;
	param[1].buffer_length = sizeof(avg);

	if (mysql_stmt_bind_result(stmt, param)) {
		finish_with_stmt_error(conn, stmt, "Unable to bind column parameters\n", true);
	}
	int i = 1;
	printf("Conducenti: \n");
	/* assemble course general information */
	while (true) {
		status = mysql_stmt_fetch(stmt);

		if (status == 1 || status == MYSQL_NO_DATA)
			break;

		strcpy_s((*ret)[row].CF,16, CF);
		printf_s("%d) Conducente : %s\n",i,CF);
		i++;
		row++;
	}

	return row;
}
static size_t parse_avgss(MYSQL* conn, MYSQL_STMT* stmt, struct average_gradess** ret)
{

	int status;
	size_t row = 0;
	MYSQL_BIND param[2];
	my_bool is_null;
```

```
double avg;
int Matricola;

if (mysql_stmt_store_result(stmt)) {
        fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
        fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
        exit(0);
}

*ret = malloc(mysql_stmt_num_rows(stmt) * sizeof(struct average_gradess));

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &Matricola;
param[0].buffer_length = 16;

/*Devo restiturilo nella store procedure*/
param[1].buffer_type = MYSQL_TYPE_DOUBLE;
param[1].buffer = &avg;
param[1].buffer_length = sizeof(avg);

if (mysql_stmt_bind_result(stmt, param)) {
        finish_with_stmt_error(conn, stmt, "Unable to bind column parameters\n", true);
}
int i = 1;
printf("Veicoli: \n");
/* assemble course general information */
while (true) {
        status = mysql_stmt_fetch(stmt);

        if (status == 1 || status == MYSQL_NO_DATA)
```

```
                    break;

            (*ret)[row].Matricola = Matricola;
            printf_s("%d) Veicolo : %d\n", i, Matricola);
            i++;
            row++;
        }

        return row;
}
static void Visualizza_conducenti_attivi(MYSQL* conn) {

        MYSQL_STMT* Visualizza_conducenti;
        int status;
        bool first = true;
        struct average_grades *avgs=NULL; /*DEVO TOGLIERE IL NULL MA MI DA ERRORE*/
        size_t conducenti = 0;
        char header[512];

        // Prepare stored procedure call
        if (!setup_prepared_stmt(&Visualizza_conducenti, "call Visualizza_conducenti_attivi()", conn)) {
                finish_with_stmt_error(conn, Visualizza_conducenti, "Unable to initialize career report statement\n", false);
        }



        // Run procedure
        if (mysql_stmt_execute(Visualizza_conducenti) != 0) {
                print_stmt_error(Visualizza_conducenti, "An error occurred while retrieving the career report.");
```

```
		goto out;
	}

	// We have multiple result sets here!
	do {
		// Skip OUT variables (although they are not present in the procedure...)
		if (conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		if (first) {
			parse_avgs(conn, Visualizza_conducenti, &avgs);
			first = false;
		}
		else {
			sprintf_s(header,512, "\nConducente: %s", avgs[conducenti].CF);
			dump_result_set(conn, Visualizza_conducenti, header);
			conducenti++;
		}

		// more results? -1 = no, >0 = error, 0 = yes (keep looking)
	next:
		status = mysql_stmt_next_result(Visualizza_conducenti);
		if (status > 0)
			finish_with_stmt_error(conn, Visualizza_conducenti, "Unexpected condition",
true);

	} while (status == 0);

out:
	mysql_stmt_close(Visualizza_conducenti);
```

```
}
static void Visualizza_conducenti_fermi(MYSQL* conn) {

	MYSQL_STMT* Visualizza_conducenti_f;
	int status;
	bool first = true;
	struct average_grades* avgs = NULL; /*DEVO TOGLIERE IL NULL MA MI DA ERRORE*/
	size_t conducenti = 0;
	char header[512];

	// Prepare stored procedure call
	if (!setup_prepared_stmt(&Visualizza_conducenti_f, "call Visualizza_conducenti_fermi()", conn)) {
		finish_with_stmt_error(conn, Visualizza_conducenti_f, "Unable to initialize career report statement\n", false);
	}



	// Run procedure
	if (mysql_stmt_execute(Visualizza_conducenti_f) != 0) {
		print_stmt_error(Visualizza_conducenti_f, "An error occurred while retrieving the career report.");
		goto out;
	}

	// We have multiple result sets here!
	do {
		// Skip OUT variables (although they are not present in the procedure...)
		if (conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}
```

```
		if (first) {
			parse_avgs(conn, Visualizza_conducenti_f, &avgs);
			first = false;
		}
		else {
			sprintf_s(header, 512, "\nConducente: %s", avgs[conducenti].CF);
			dump_result_set(conn, Visualizza_conducenti_f, header);
			conducenti++;
		}

		// more results? -1 = no, >0 = error, 0 = yes (keep looking)
	next:
		status = mysql_stmt_next_result(Visualizza_conducenti_f);
		if (status > 0)
			finish_with_stmt_error(conn, Visualizza_conducenti_f, "Unexpected condition", true);

	} while (status == 0);

out:
	mysql_stmt_close(Visualizza_conducenti_f);

}
static void Visualizza_veicoli_attivi(MYSQL* conn) {
	MYSQL_STMT* Visualizza_veicoli;
	int status;
	bool first = true;
	struct average_gradess* avgs = NULL;
	size_t veicoli = 0;
	char header[512];
```

```
	// Prepare stored procedure call
	if (!setup_prepared_stmt(&Visualizza_veicoli, "call Visualizza_veicoli_attivi()", conn)) {
		finish_with_stmt_error(conn, Visualizza_veicoli, "Unable to initialize career report statement\n", false);
	}




	// Run procedure
	if (mysql_stmt_execute(Visualizza_veicoli) != 0) {
		print_stmt_error(Visualizza_veicoli, "An error occurred while retrieving the career report.");
		goto out;
	}

	// We have multiple result sets here!
	do {
		// Skip OUT variables (although they are not present in the procedure...)
		if (conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		if (first) {
			parse_avgss(conn, Visualizza_veicoli, &avgs);
			first = false;
		}
		else {
			sprintf_s(header, 512, "\nVeicolo: %i", avgs[veicoli].Matricola);
			dump_result_set(conn, Visualizza_veicoli, header);
			veicoli++;
		}

		// more results? -1 = no, >0 = error, 0 = yes (keep looking)
```

```
	next:
		status = mysql_stmt_next_result(Visualizza_veicoli);
		if (status > 0)
			finish_with_stmt_error(conn, Visualizza_veicoli, "Unexpected condition", true);

	} while (status == 0);

out:
	mysql_stmt_close(Visualizza_veicoli);

} /*SBAGLIA IL CODICE DEI VEICOLI*/
static void Visualizza_veicoli_fermi(MYSQL* conn) {
	MYSQL_STMT* Visualizza_veicoli_f;
	int status;
	bool first = true;
	struct average_gradess* avgs = NULL;
	size_t veicoli = 0;
	char header[512];

	// Prepare stored procedure call
	if (!setup_prepared_stmt(&Visualizza_veicoli_f, "call Visualizza_veicoli_fermi()", conn)) {
		finish_with_stmt_error(conn, Visualizza_veicoli_f, "Unable to initialize career report statement\n", false);
	}



	// Run procedure
	if (mysql_stmt_execute(Visualizza_veicoli_f) != 0) {
		print_stmt_error(Visualizza_veicoli_f, "An error occurred while retrieving the career report.");
		goto out;
```

```
	}

	// We have multiple result sets here!
	do {
		// Skip OUT variables (although they are not present in the procedure...)
		if (conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		if (first) {
			parse_avgss(conn, Visualizza_veicoli_f, &avgs);
			first = false;
		}
		else {
			sprintf_s(header, 512, "\nVeicolo: %i", avgs[veicoli].Matricola);
			dump_result_set(conn, Visualizza_veicoli_f, header);
			veicoli++;
		}

		// more results? -1 = no, >0 = error, 0 = yes (keep looking)
	next:
		status = mysql_stmt_next_result(Visualizza_veicoli_f);
		if (status > 0)
			finish_with_stmt_error(conn, Visualizza_veicoli_f, "Unexpected condition",
true);

	} while (status == 0);

out:
	mysql_stmt_close(Visualizza_veicoli_f);
} /*SBAGLIA IL CODICE DEI VEICOLI*/
static void Elimina_conducente(MYSQL* conn) {
```

```
MYSQL_STMT* Elimina;
MYSQL_BIND param[2];
char codice[16];
char patente[10];
printf("Inserisci il codice fiscale : ");
scanf_s("%s", codice,16);
printf("Inserisci il numero della patente : ");
scanf_s("%s",patente,10);

if (!setup_prepared_stmt(&Elimina, "call Elimina_conducente(?, ?)", conn)) {
        print_stmt_error(Elimina, "Unable to initialize login statement\n");
}

// Prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[0].buffer = codice;
param[0].buffer_length = strlen(codice);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[1].buffer = patente;
param[1].buffer_length = strlen(patente);

if (mysql_stmt_bind_param(Elimina, param) != 0) {
        finish_with_stmt_error(conn, Elimina, "Could not bind parameters for career report\
n", true);
}
// Run procedure
if (mysql_stmt_execute(Elimina) != 0) {
        print_stmt_error(Elimina, "An error occurred while retrieving the career report.");
        goto out;
```

```
	}
	printf("Hai eliminato | CF: %s   Patente: %s\n", codice, patente);
	system("pause");
	mysql_stmt_close(Elimina);
	return;



out:
	mysql_stmt_close(Elimina);

} /*MI CONFERMA IL TUTTO MA NON LO ELIMINA SUL DATABASE*/
static void Emetti_biglietto(MYSQL* conn) {
	MYSQL_STMT* EmettiB=NULL;
	int biglietti;
	printf("Inserisci il numero di biglietti da emettere : ");
	scanf_s("%d", &biglietti);

	for (int j = 0; j < biglietti; j++)
	{
		if (!setup_prepared_stmt(&EmettiB, "call Emetti_biglietto()", conn)) {
			print_stmt_error(EmettiB, "Unable to initialize login statement\n");
		}
		// Run procedure
		if (mysql_stmt_execute(EmettiB) != 0) {
			print_stmt_error(EmettiB, "An error occurred while retrieving the career report.");
			goto out;
		}
	}
	printf("Hai emanato %d biglietti !\n",biglietti);
	system("pause");
	mysql_stmt_close(EmettiB);
```

```
	return;
out:
	mysql_stmt_close(EmettiB);
}
static void Emetti_abbonamenti(MYSQL* conn) {
	MYSQL_STMT* EmettiA=NULL;
	int abbonamenti;
	printf("Inserisci il numero di biglietti da emettere : ");
	scanf_s("%d", &abbonamenti);

	for (int j = 0; j < abbonamenti; j++)
	{
		if (!setup_prepared_stmt(&EmettiA, "call Emetti_abbonamenti()", conn)) {
			print_stmt_error(EmettiA, "Unable to initialize login statement\n");
		}
		// Run procedure
		if (mysql_stmt_execute(EmettiA) != 0) {
			print_stmt_error(EmettiA, "An error occurred while retrieving the career report.");
			goto out;
		}
	}
	printf("Hai emanato %d abbonamenti !\n", abbonamenti);
	system("pause");
	mysql_stmt_close(EmettiA);
	return;
out:
	mysql_stmt_close(EmettiA);
}
static void Aggiungi_utente(MYSQL* conn)
{
	MYSQL_STMT* distanza_stmt;
```

```
MYSQL_BIND param[3];

char username[45];
char password[45];
char ruolo[45];



printf("Inserisci l'username : ");
scanf_s("%s", username,45);
printf("Inserisci la password (e' consigliato usare pippo) : ");
scanf_s("%s", password,45);
printf("Inserisci il ruolo (AMMINISTRATORE|GUIDATORE|PASSEGGERO) : ");
scanf_s("%s", ruolo,45);



if (!setup_prepared_stmt(&distanza_stmt, "call Aggiungi_utente(?, ?, ?)", conn)) {
        print_stmt_error(distanza_stmt, "Unable to initialize login statement\n");
}

// Prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[0].buffer = username;
param[0].buffer_length = strlen(username);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
param[1].buffer = password;
param[1].buffer_length = strlen(password);

param[2].buffer_type = MYSQL_TYPE_VAR_STRING;  //IN
```

```
	param[2].buffer = ruolo;
	param[2].buffer_length = strlen(ruolo);

	if (mysql_stmt_bind_param(distanza_stmt, param) != 0) {
		finish_with_stmt_error(conn, distanza_stmt, "Could not bind parameters for career report\n", true);
	}

	// Run procedure
	if (mysql_stmt_execute(distanza_stmt) != 0) {
		print_stmt_error(distanza_stmt, "An error occurred while retrieving the career report.");
		goto out;
	}

	printf("Hai aggiunto : Username %s, Ruolo: %s\n", username, ruolo);
	system("pause");
	mysql_stmt_close(distanza_stmt);
	return;

out:
	mysql_stmt_close(distanza_stmt);

}

void run_as_administrator(MYSQL* conn)
{
	int s;
	int i = 0;
	int numero;
	while (true) {
```

```
		printf("----------***** Cosa vuoi fare? *****----------\n\n");
		printf("1) Assegna turno ad un conducente\n");
		printf("2) Assegna un veicolo ad una tratta\n");
		printf("3) Assumi un conducente\n");
		printf("4) Cambia turno ad un conducente\n");
		printf("5) Visualizza conducenti attivi\n");
		printf("6) Visualizza conducenti fermi\n");
		printf("7) Visualizza veicoli attivi\n");
		printf("8) Visualizza veicoli fermi\n");
		printf("9) Elimina un conducente\n");
		printf("10) Emetti biglietti\n");
		printf("11) Emetti abbonamenti\n");
		printf("12) Aggiungi utente\n");
		printf("13) Logout\n");
		printf("SCELTA: ");
		scanf_s("%i", &numero);
		switch (numero)
		{
		case 1:
			printf("---------- Assegna_turno_al_conducente----------\n");
			Assegna_turno_al_conducente(conn);
			break;
		case 2:
			printf("---------- Assegna_veicolo_alla_tratta----------\n");
			Assegna_veicolo_alla_tratta(conn);
			break;
		case 3:
			printf("---------- Assumi_conducente----------\n");
```

```
                Assumi_conducente(conn);
                break;
            case 4:
                printf("----------------------------------------------
Cambia_conducente_turno-----------------------------------------------\n");
                Cambia_conducente_turno(conn);
                break;
            case 5:
                printf("----------------------------------------------
Visualizza_conducenti_attivi------------------------------------------\n");
                Visualizza_conducenti_attivi(conn);
                break;
            case 6:
                printf("----------------------------------------------
Visualizza_conducenti_fermi-------------------------------------------\n");
                Visualizza_conducenti_fermi(conn);
                break;
            case 7:
                printf("----------------------------------------------
Visualizza_veicoli_attivi---------------------------------------------\n");
                Visualizza_veicoli_attivi(conn);
                break;
            case 8:
                printf("----------------------------------------------
Visualizza_veicoli_fermi----------------------------------------------\n");
                Visualizza_veicoli_fermi(conn);
                break;
            case 9:
                printf("----------------------------------------------
Elimina_conducente----------------------------------------------------\n");
                Elimina_conducente(conn);
                break;
            case 10:
```

```
                printf("-----------------------------------------
Emetti_biglietto-----------------------------------------------------\n");
                Emetti_biglietto(conn);
                break;
        case 11:
                printf("-----------------------------------------
Emetti_abbonamenti---------------------------------------------------\n");
                Emetti_abbonamenti(conn);
                break;
        case 12:
                printf("-----------------------------------------Aggiungi
utente-----------------------------------------------------\n");
                Aggiungi_utente(conn);
                break;
        case 13:
                printf("-----------------------------------------------------
Logout-----------------------------------------------------\n");
                return;
        default:
                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                break;
        }
    }



}
```

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql.h>

#include "defines.h"
```

```
typedef enum {
	AMMINISTRATORE = 1,
	CONDUCENTE=2,
	PASSEGGERO=3,
} role_t;


struct configuration conf;

static MYSQL* conn;


static role_t attempt_login(MYSQL* conn, char* username, char* password)
{
	MYSQL_STMT *login_procedure;

	MYSQL_BIND param[3]; // Used both for input and output
	int role = 0;

	if (!setup_prepared_stmt(&login_procedure, "call Login(?, ?, ?)", conn)) {
		print_stmt_error(login_procedure, "Unable to initialize login statement\n");

	}

	// Prepare parameters
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
	param[0].buffer = username;
	param[0].buffer_length = strlen(username);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
	param[1].buffer = password;
	param[1].buffer_length = strlen(password);

	param[2].buffer_type = MYSQL_TYPE_LONG; // OUT
	param[2].buffer = &role;
	param[2].buffer_length = sizeof(role);

	if (mysql_stmt_bind_param(login_procedure, param) != 0) { // Note _param
		print_stmt_error(login_procedure, "Could not bind parameters for login");
		goto err;
	}

	// Run procedure
	if (mysql_stmt_execute(login_procedure) != 0) {
		print_stmt_error(login_procedure, "Could not execute login procedure");
		goto err;
	}
```

```
	// Prepare output parameters
	memset(param, 0, sizeof(param));
	param[0].buffer_type = MYSQL_TYPE_LONG; // OUT
	param[0].buffer = &role;
	param[0].buffer_length = sizeof(role);
	if (mysql_stmt_bind_result(login_procedure, param)) {
		print_stmt_error(login_procedure, "Could not retrieve output parameter");
		goto err;
	}

	// Retrieve output parameter
	if (mysql_stmt_fetch(login_procedure)) {
		print_stmt_error(login_procedure, "Could not buffer results");
		goto err;
	}

	mysql_stmt_close(login_procedure);
	return role;

err:
	mysql_stmt_close(login_procedure);

}


main(void) {
	printf("***************	Benevenuto nel nuovo sistema di trasporto pubblico di Ivan
Palmieri	***************\n\n\n");
	role_t role;
	conn = mysql_init(NULL);
	if (conn == NULL) {
		fprintf(stderr, "mysql_init() failed (probably out of memory)\n");
		exit(EXIT_FAILURE);
	}
	/*si connette*/
	if (mysql_real_connect(conn, "localhost", "root", conf.db_password,
"sistematrasportopubblico3", conf.port, NULL, CLIENT_MULTI_STATEMENTS |
CLIENT_MULTI_RESULTS) == NULL) {
		fprintf(stderr, "mysql_real_connect() failed\n");
		mysql_close(conn);
		exit(EXIT_FAILURE);
	}
	char us[25];
	char pass[25];
	int choise;
	while (true)

	{
		printf("*** Cosa vuoi fare? *** \n");
		printf("1) Login\n");
```

```
                printf("2) Esci\n");
                printf("SCELTA: ");
                scanf_s("%i", &choise);
                switch (choise)
                {
                case 1:
                        printf("-----------------------------------------------------
Login-----------------------------------------------------\n");
                        printf("Username: ");
                        scanf_s("%s", &us);
                        printf("Password: ");
                        scanf_s("%s", &pass);


printf("-----------------------------------------------------------------------------------------------------------
----\n");
                        role = attempt_login(conn, us, pass);
                        switch (role)
                        {
                        case PASSEGGERO:
                                printf("Connessione riuscita come passeggero\n");
                                run_as_passeggero(conn);
                                break;

                        case CONDUCENTE:
                                printf("Connessione riuscita come guidatore\n");
                                run_as_guidatore(conn);
                                break;

                        case AMMINISTRATORE:
                                printf("Connessione riuscita come amministratore\n");
                                run_as_administrator(conn);
                                break;
                        default:
                                printf("Credenziali sbagliate\n");

                        }
                        break;
                case 2:


printf("-----------------------------------------------------------------------------------------------------------
------\n");
                        printf("Bye!\n");
                        system("pause");
                        mysql_close(conn);
                        return;
                }
```

```
        }
}
```

Defines.h

```
#pragma once

#include <stdbool.h>
#include <mysql.h>

struct configuration {
        char* host;
        char* db_username;
        char* db_password;
        unsigned int port;
        char* database;

        char username[128];
        char password[128];
};

extern struct configuration conf;


extern void run_as_guidatore(MYSQL* conn);
extern void run_as_passeggero(MYSQL* conn);
extern void run_as_administrator(MYSQL* conn);
```

Amministratore.json

```
{
  "host": "localhost",
  "username": "amministratore",
  "password": "amministratore",
  "port": 3306,
  "database": "sistematrasportopubblico3"
}
```

Conducente.json

```
{
  "host": "localhost",
  "username": "conducente",
  "password": "conducente",
  "port": 3306,
  "database": "sistematrasportopubblico3"
}
```

Passeggero.json


```
{
  "host": "localhost",
  "username": "passeggero",
  "password": "passeggero",
  "port": 3306,
  "database": "sistematrasportopubblico3"
}
```