|  |  |
| --- | --- |
|  |  |

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 3](#_Toc606296459)

[2. Analisi dei Requisiti 4](#_Toc1289394997)

[3. Progettazione concettuale 5](#_Toc2081466291)

[4. Progettazione logica 6](#_Toc2147004904)

[5. Progettazione fisica 8](#_Toc518560220)

[Appendice: Implementazione 9](#_Toc403811585)

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | 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. |

# 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:  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 di un atuobus | Fermata inziale | Tratta |
| 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. |

# 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.

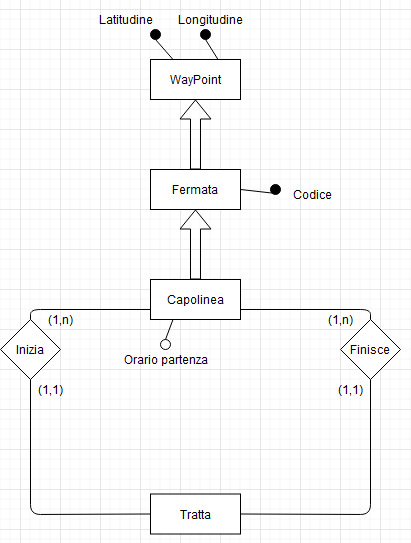
Successivamente ho utilizzato le relazioni (inizia) e (finisce) per indicare la fermata di partenza e di arrivo della tratta, utilizzando in entrambe, delle relazioni zero ad enne , andando ad indicare che per una singola tratta e` presente una sola fermata di partenza e una sola fermata di arrivo, e 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.

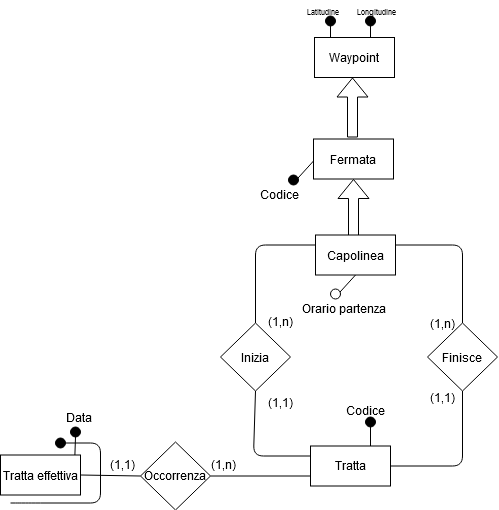
Successivamente visto che i waypoint,da come evince dal testo,sono rappresentati da una latitudine e longitudine,introduco due attributi inerenti a questi dati, i quali in coppia formeranno la sua chiave primara; mediante la generalizzazione anche le entità fermata e capolinea erediteranno tali attribbuti.

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.

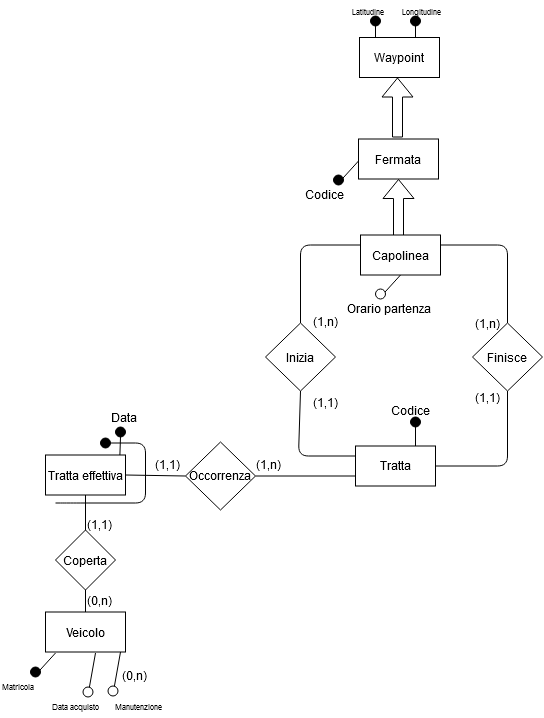
Utilizzo l’associazone (occorrenza) per indicare che in quella tratta astratta possono esserci piu tratte concrete, e che una tratta concreta ovviamente percorrerà una sola tratta.

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.

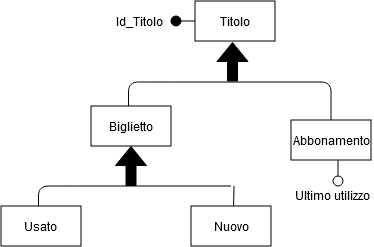
Quindi unisco le due entità mediante l’associazione coperta, la quale indica che un veicolo può coprire piu tratte effettive in diversi giorni e orari, mentre la tratta effetettiva è coperta da un solo veicolo in quel determinato giorno e orario.

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).

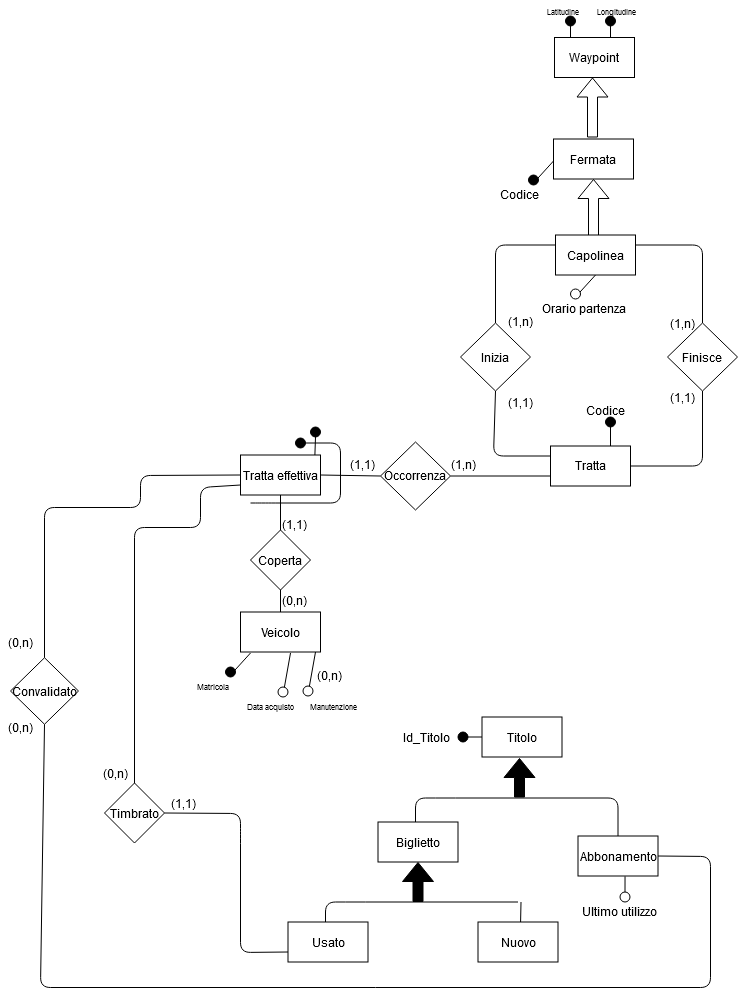
Quindi si inserisce nel diagramma tutta questa generalizzazione dei titoli di viaggio.



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

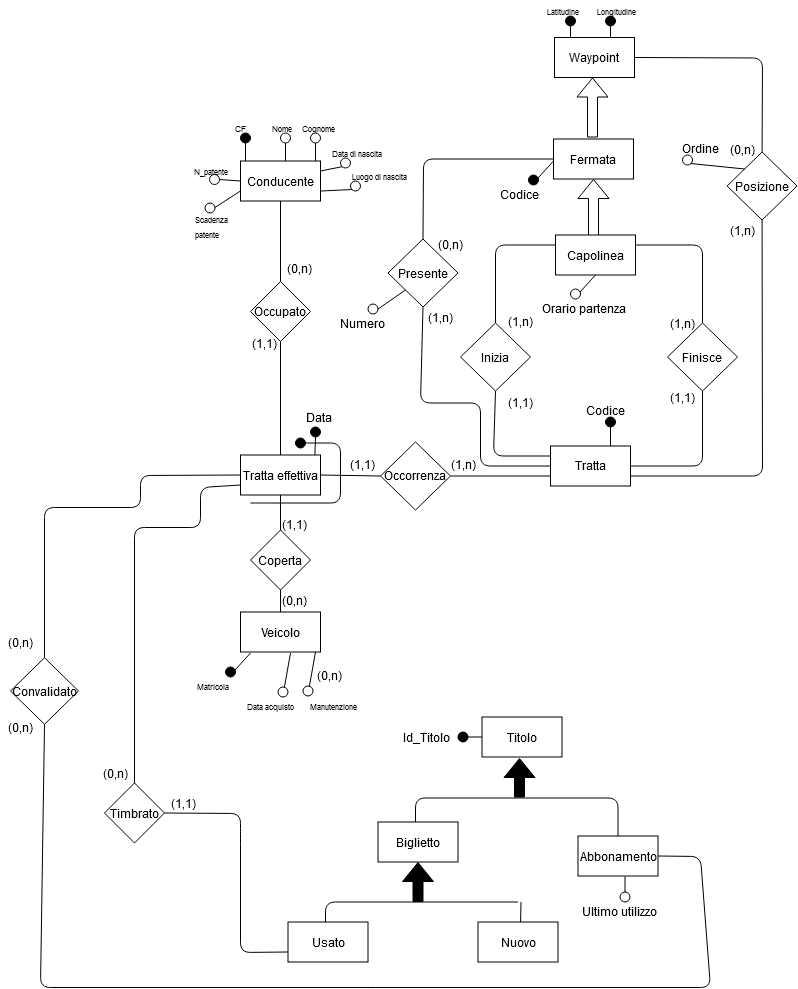


Da come si evince dal testo, bisogna tenere contro dei waypoint presente nelle tratte; Ovviamente un waypoint avrà un codice numero diverso a seconda della tratta, per raccolgiere l’informazione inerente ad un waypoint su una specifica tratta, viene introdotta l’associazione (Posizione) che collega le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

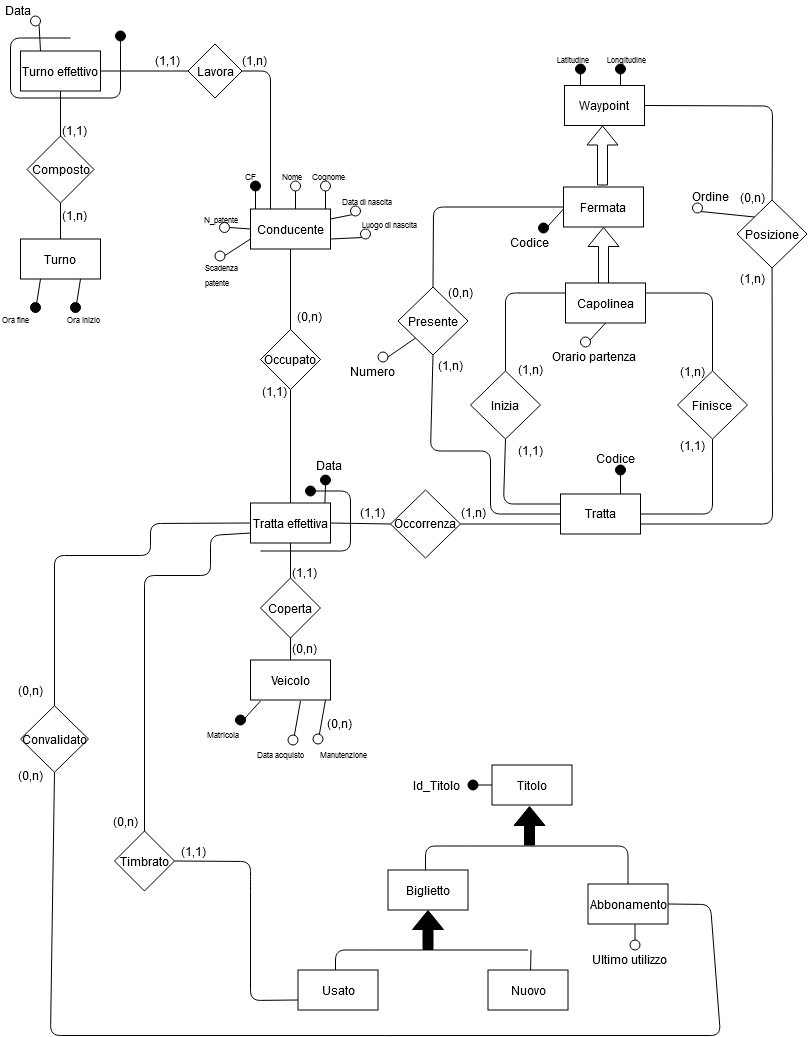


In fine da come si evince nel testo ogni conducente effettua dei turni, quindi viene creata l’entità (Turno) la quale indica il concetto astratto di turno, e l’entità (Turno effettivo) la quala indica il concetto concreto di turno.

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.

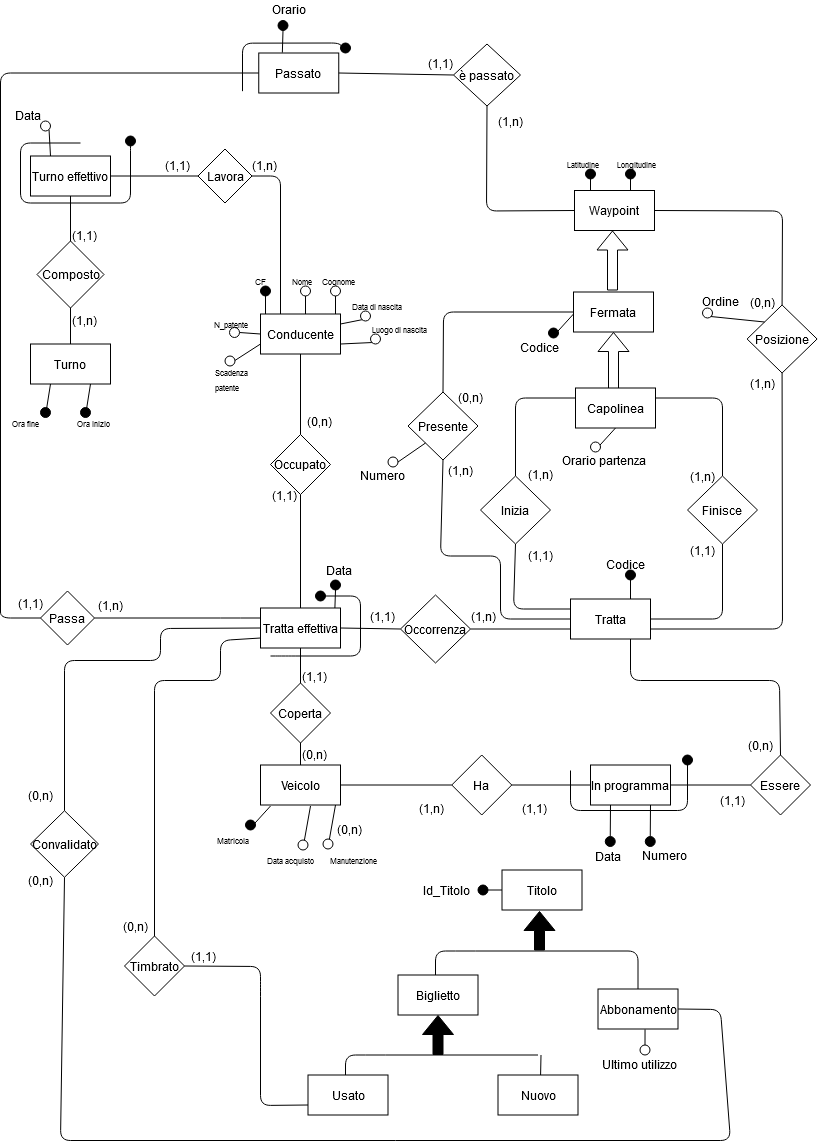
L’entità turno invece viene collegata con l’entità turno effettivo mediante l’associazione composto andando ad indicare che ovviamente in un turno astratto possono esserci effettivamente più turni concreti, mentre invece un turno conreto è riferito ad un unico turno astrattto.

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

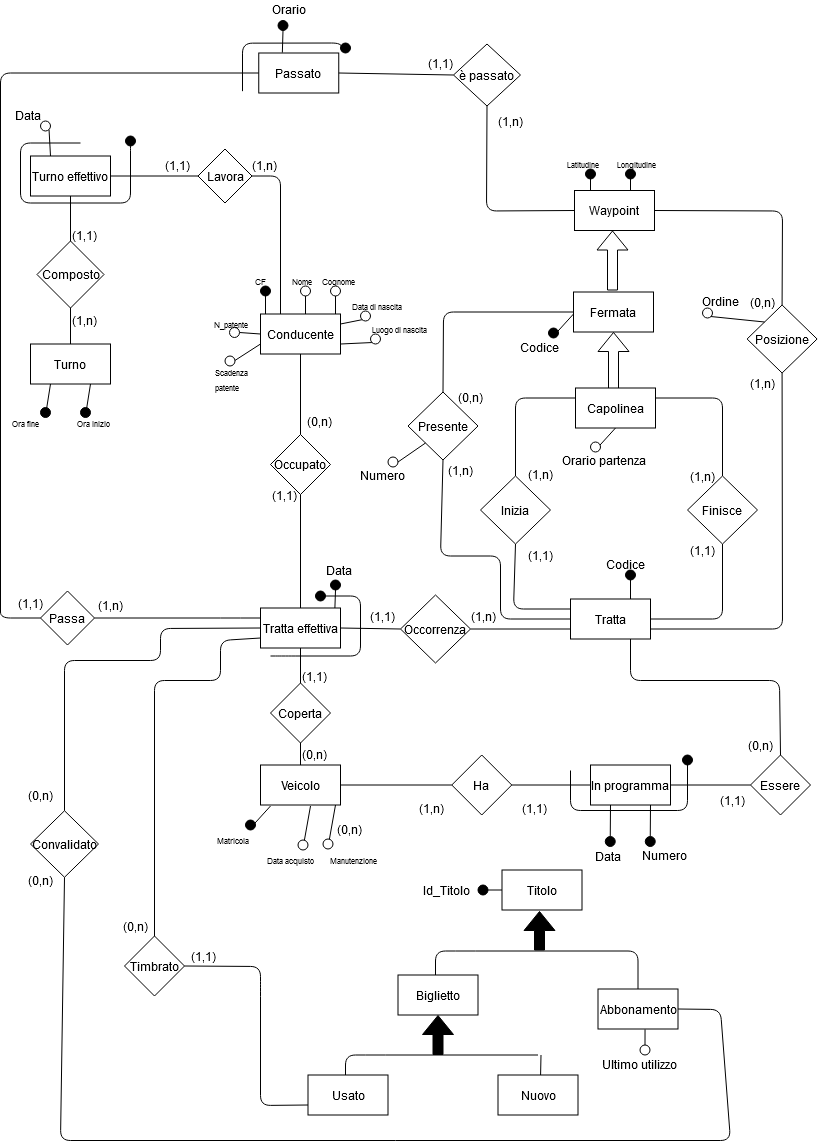


In aggiunta per tenere traccia di quale waypoint sono presenti nella mia tratta, creo una relazione (Passato) che va ad indicare quale waypoint ho appena passato in una determinata tratta, al fine di poter calcolare successivamente la distanza effettiva del veicolo da una fermata.

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 |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| **Concetto nello schema** | **Tipo[[1]](#footnote-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 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 400Costi 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)

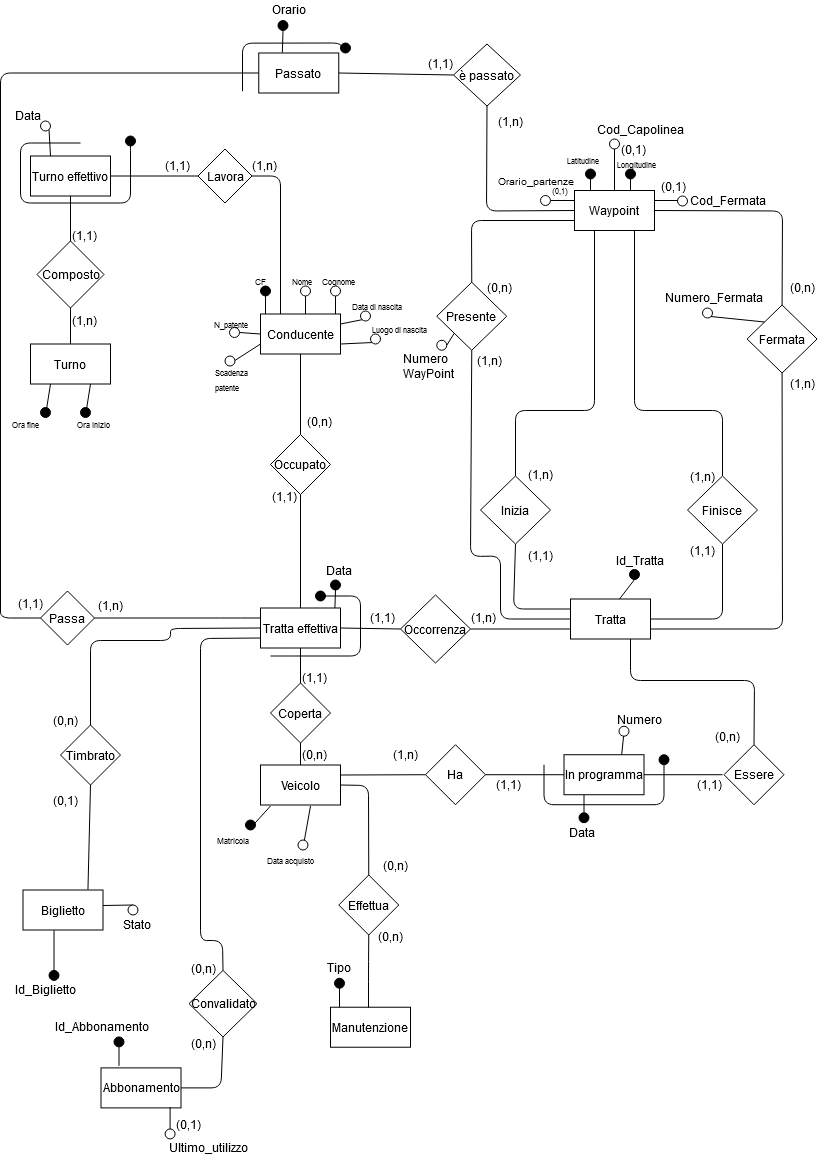
In programma (Veicolo, Tratta, Numero, Data)

Convalidato (Abbonamento, Veicolo, Tratta, Data)

Effettua (Manutenzione, Veicolo)

Presente (Tratta, Latitudine, Longitudine, Numero\_Waypoint)

Fermata (Numero\_fermata, Latitudine, Longitudine, Tratta)



## Normalizzazione del modello relazionale

Le tabelle sono tutte in 3NF

# 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)

1. 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

1. 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]](#footnote-2)** |
| **Manutenzione** | VARCHAR(45) | PK NN |
| **Veicolo** | INT | PK NN |

|  |  |  |
| --- | --- | --- |
| **Abbonamento** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[3]](#footnote-3)** |
| **idAbbonamento** | int | PK NN |
| **Ultimo\_utilizzo** | DATETIME |  |

|  |  |  |
| --- | --- | --- |
| **Convalidato** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[4]](#footnote-4)** |
| **idAbbonamento** | int | PK NN |
| **Data** | DATE | PK NN |
| **Tratta** | INT | PK NN |
| **Veicolo** | INT | PK NN |

|  |  |  |
| --- | --- | --- |
| **Biglietto** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[5]](#footnote-5)** |
| **idBiglietto** | int | PK NN |
| **Stato** | Int |  |
| **Data** | Date |  |
| **Tratta** | Int |  |
| **Veicolo** | Int |  |

|  |  |  |
| --- | --- | --- |
| **In\_programma** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[6]](#footnote-6)** |
| **Veicolo** | int | PK NN |
| **Trata** | int | PK NN |
| **Numero** | int | NN |
| **Data** | DATE | PK NN |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Tratta\_effettiva** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[7]](#footnote-7)** |
| **Data** | Data | PK NN |
| **Tratta** | int | PK NN |
| **Conducente** | int | NN |
| **Veicolo** | int | NN PK |

|  |  |  |
| --- | --- | --- |
| **Veicolo** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[8]](#footnote-8)** |
| **Matricola** | int | PK NN |
| **Data\_Acquisto** | Data | NN |

|  |  |  |
| --- | --- | --- |
| **Turno** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[9]](#footnote-9)** |
| **Data\_inizio** | TIME | PK NN |
| **Data\_fine** | TIME | PK NN |

|  |  |  |
| --- | --- | --- |
| **Tratta** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[10]](#footnote-10)** |
| **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]](#footnote-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]](#footnote-12)** |
| **Data\_turno** | Data | PK NN |
| **Condcuente** | VARCHAR(16) | PK NN |
| **Ora\_inizio** | TIME | PK NN |
| **Ora\_fine** | TIME | PK NN |

|  |  |  |
| --- | --- | --- |
| **Fermata** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[13]](#footnote-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]](#footnote-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]](#footnote-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]](#footnote-16)** |
| **Username** | VARCHAR(45) | PK NN |
| **Password** | VARCHAR(45) | NN |
| **Ruolo** | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| **Passato** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[17]](#footnote-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]](#footnote-18)** |
| **Tipo\_manutenzione** | VARCHAR(45) | PK NN |

## Indici

|  |  |
| --- | --- |
| **In\_programma** | |
| **Indice** | **Tipo[[19]](#footnote-19):** |
| fk\_Veicolo\_has\_Tratta\_Tratta1\_idx | Primary |
| fk\_Veicolo\_has\_Tratta\_Veicolo1\_idx | Primary |

|  |  |
| --- | --- |
| **Convalidato** | |
| **Indice** | **Tipo[[20]](#footnote-20):** |
| fk\_Abbonamento\_has\_Tratta\_effettiva\_Tratta\_effettiva1\_idx | Primary |
| fk\_Abbonamento\_has\_Tratta\_effettiva\_Abbonamento1\_idx | Primary |

|  |  |
| --- | --- |
| **Biglietto** | |
| **Indice** | **Tipo[[21]](#footnote-21):** |
| fk\_Biglietto\_Tratta\_effettiva1\_idx | Secondary |

|  |  |
| --- | --- |
| **Tratta\_effettiva** | |
| **Indice** | **Tipo[[22]](#footnote-22):** |
| fk\_Tratta effettiva\_Tratta\_idx | Primary |
| fk\_Tratta effettiva\_Conducente1\_idx | Secondary |
| fk\_Tratta effettiva\_Veicolo1\_idx | Primary |

|  |  |
| --- | --- |
| **Tratta** | |
| **Indice** | **Tipo[[23]](#footnote-23):** |
| fk\_Tratta\_WayPoint1\_idx | Secondary |
| fk\_Tratta\_WayPoint2\_idx | Secondary |

|  |  |
| --- | --- |
| **Turno\_effettivo** | |
| **Indice** | **Tipo[[24]](#footnote-24):** |
| fk\_Turno effettivo\_Turno1\_idx | Primary |
| fk\_Turno effettivo\_Conducente1\_idx | Primary |

|  |  |
| --- | --- |
| **Conducente** | |
| **Indice** | **Tipo[[25]](#footnote-25):** |
| fk\_Conducente\_Utente1\_idx | Primary |

|  |  |
| --- | --- |
| **Fermata** | |
| **Indice** | **Tipo[[26]](#footnote-26):** |
| fk\_Tratta\_has\_WayPoint\_WayPoint2\_idx | Primary |
| fk\_Tratta\_has\_WayPoint\_WayPoint2\_idx | Primary |

|  |  |
| --- | --- |
| **Presente** | |
| **Indice** | **Tipo[[27]](#footnote-27):** |
| fk\_Tratta\_has\_WayPoint\_WayPoint1\_idx | Primary |
| fk\_Tratta\_has\_WayPoint\_WayPoint1\_idx | Primary |

|  |  |
| --- | --- |
| **Passato** | |
| **Indice** | **Tipo[[28]](#footnote-28):** |
| fk\_Tratta\_effettiva\_has\_WayPoint\_WayPoint1\_idx | Secondary |
| fk\_Tratta\_effettiva\_has\_WayPoint\_Tratta\_effettiva1\_idx | Secondary |

|  |  |
| --- | --- |
| **Manutenzione\_Veicolo** | |
| **Indice** | **Tipo[[29]](#footnote-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**

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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **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

1. **Emetti abboamenti**

CREATE PROCEDURE `Emetti\_abbonamenti` ()

BEGIN

insert into `Abbonamento`() values ();

END

1. **Emetti biglietti**

CREATE PROCEDURE `Emetti\_biglietto` ()

BEGIN

insert into `Biglietto` (Stato) values ("0");

END

1. **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"

}

1. Indicare con E le entità, con R le relazioni [↑](#footnote-ref-1)
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. [↑](#footnote-ref-2)
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. [↑](#footnote-ref-3)
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. [↑](#footnote-ref-4)
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. [↑](#footnote-ref-5)
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. [↑](#footnote-ref-6)
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. [↑](#footnote-ref-7)
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. [↑](#footnote-ref-8)
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. [↑](#footnote-ref-9)
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. [↑](#footnote-ref-10)
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. [↑](#footnote-ref-11)
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. [↑](#footnote-ref-12)
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. [↑](#footnote-ref-13)
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. [↑](#footnote-ref-14)
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. [↑](#footnote-ref-15)
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. [↑](#footnote-ref-16)
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. [↑](#footnote-ref-17)
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. [↑](#footnote-ref-18)
19. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-19)
20. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-20)
21. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-21)
22. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-22)
23. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-23)
24. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-24)
25. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-25)
26. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-26)
27. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-27)
28. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-28)
29. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-29)