

Fase tre

Indice degli argomenti

- [Indice degli argomenti](#)
- [Progetto VRoomA](#)
 - [Componenti del gruppo](#)
 - [Motivazioni](#)
 - [Obiettivi](#)
 - [Raccolta dei dati](#)
 - [Analisi dei requisiti](#)
 - [Glossario delle entità](#)
 - [Glossario dei termini](#)
 - [Glossario delle relazioni](#)
 - [Schemi](#)
 - [Schemi di relazione](#)
 - [Schema Logico](#)
 - [Normalizzazione](#)
 - [Schema Fisico](#)
 - [Generalizzazione](#)
 - [Implementazione Database - MySQL](#)
 - [Creazione delle tabelle](#)
 - [Triggers](#)
 - [Inserimenti](#)
 - [Script di creazione automatica di query](#)
 - [Query](#)
 - [Ottimizzazione](#)
 - [Algebra Relazionale](#)
 - [Calcolo Relazionale](#)
 - [Sicurezza](#)
 - [Views](#)
 - [Creazione Utenti](#)

Progetto VRoomA

Componenti del gruppo

Nome	Cognome	Matricola	Mail
Leonardo	Ascenzi	0310858	leonardo.ascenzi@students.uniroma2.eu
Franco	Salvucci	0306604	franco.salvucci@students.uniroma2.eu
Nicolò	Spadoni	0311175	nicolo.spadoni@students.uniroma2.eu

Motivazioni

Il database che stiamo realizzando è incentrato all'implementazione di un software dedicato all'organizzazione di viaggi tramite taxi.

Obiettivi

L'obiettivo principale di questo sistema è permettere agli utenti di organizzare gli spostamenti tramite taxi a seconda delle proprie esigenze, del tipo di veicolo scelto e del costo della tratta scelta.

Da un punto di vista societario, gli obiettivi sono quelli di valutare la qualità del lavoro degli autisti tramite i feedback forniti dai clienti e migliorare dove possibile il servizio.

Raccolta dei dati

Analisi dei requisiti

I **ruoli aziendali** sono i seguenti:

- Addetti Marketing
- Autisti
- Manutentori

Gli **addetti al marketing** possono inserire, previa autorizzazione da parte degli amministratori della società, delle **promo** che prevedono sconti sulle corse per gli utenti del sistema.

Gli **autisti** potranno scegliere se accettare o rifiutare la corsa, specificando in questo caso la motivazione del rifiuto.

Inoltre potranno lasciare un **feedback** all'utente riguardo il comportamento prima e durante la corsa.

Ogni **autista** ha la propria macchina privata, e può contattare i manutentori aziendali in caso di guasto del veicolo.

Ad ogni **autista** sono assegnati uno o più **turni** di lavoro, con il vincolo che il singolo autista non può essere assegnato a due turni lavorativi che hanno orario inizio e orario fine uguali.

I **manutentori** possono ricevere richieste di assistenza da parte degli autisti e contattare le officine convenzionate per effettuare il lavoro di assistenza.

Le **officine** non fanno parte della società.

Le tipologie di **veicolo** disponibili sono le seguenti:

- Base: 4 posti disponibili
- Plus: 7 posti disponibili, adibito a trasporto di carrozzine per disabili
- Premium: 12 posti disponibili, adibito a trasporto di carrozzine per disabili

Ogni **veicolo**, identificato in modo univoco dalla targa, per poter circolare, deve essere assicurato.

Quando si prenota una **corsa (tratta)** si possono scegliere due punti:

- Punto di Partenza, identificato come Punto di raccolta
- Punto di Arrivo, identificato come Punto di rilascio

Ogni **prenotazione** può essere accettata o rifiutata in base a determinate esigenze dell'**utente** (Es: prenotazione effettuata per errore) e dell'**autista** (Es: indisponibilità al servizio).

Ad ogni tratta completata è associato un feedback che può essere lasciato sia dall'**utente** che dall'**autista**.

Ogni **utente** ha diritto a ricevere **offerte** da poter usare al momento della prenotazione.

Può effettuare illimitate richieste di **prenotazione**, in base alle necessità personali (numero di passeggeri, persone con disabilità, punto di ritiro, punto di rilascio), con il vincolo di una corsa per volta.

A corsa completata l'**utente** può lasciare un **feedback** con un numero di stelle (da 1 a 5) e un commento.

Ogni **utente** deve aggiungere una **carta** con cui effettuare il pagamento relativo alla tratta effettuata, in un

secondo momento potrà aggiungere altri metodi di pagamento secondo le proprie esigenze.
 Ogni **utente** può aggiungere alla lista dei preferiti una qualunque delle tratte effettuate da lui, scegliendo se aggiungere solo la tratta o anche l'autista.
 Ogni **utente** può accedere alla cronologia delle prenotazioni effettuate.

Glossario delle entità

Entità	Descrizione	Attributi	Relazioni Coinvolte
Personale	Membri totali della società	ID , Nome, Cognome, DDN, Numero di Telefono, Email	Addetti Marketing, Manutentori, Autisti
Patente	Describe tutte le info riguardanti la patente degli autisti	Numero Patente , DDS, Categoria	Autisti
Offerte	Serie di offerte che vengono proposte al singolo utente	ID_Offerta , Promo Code, Info Offerta,	Utenti, Addetti Marketing
Manutentori	Addetti alla manutenzione delle auto degli autisti	ID_Manutentore , Qualifica	Personale, Autisti
Autisti	Personale che svolge il ruolo di autista delle auto nella società	ID_Autista , Stipendio	Patente, Manutentori, Veicoli, Turni, Richiesta Prenotazione, Personale, Feedback
Veicoli	Auto utilizzate per il servizio di taxi	Targa , Marca, Modello, Posti disponibili	Autisti, Assicurazione
Turni	Turni lavorativi che riguardano gli autisti	ID_Turno , Orario inizio, Orario fine	Autisti
Richiesta Prenotazione	Richieste di prenotazioni effettuate da parte dell'utente	ID_Richiesta , Punto di raccolta, Punto di rilascio, Data richiesta, Orario richiesta, Numero Passeggeri	Autisti, Utenti, Tratte Complete, Tratte Rifiutate
Utenti	Utenti utilizzatori del servizio taxi	ID_Utente , Nome, Cognome, Email, Password	Carta, Richiesta Prenotazione, Offerte, Feedback, Tratte completate
Feedback	Recensioni lasciate dall'utente e dagli autisti	ID_Feedback , StelleUtente, CommentoUtente, StelleAutista, CommentoAutista	Tratte Completate, Utenti, Autisti
Tratte Completate	Corse effettuate portate a termine con successo	ID_TrattaC , Costo	Richiesta Prenotazione, Feedback, Carta, Utenti
Tratte Rifiutate	Corse rifiutate da parte dell'autista per determinati motivi	ID_TrattaR , Motivazione	Richiesta Prenotazione
Carta	Carta di credito personale dell'utente	Numero Carta , Data di Scadenza, CVV	Utenti, Tratte completate
Assicurazioni	Dati dell'assicurazione associata al singolo veicolo	ID_Assurazione , Data di scadenza, Tipo	Veicoli

Entità	Descrizione	Attributi	Relazioni Coinvolte
Addetti Marketing	Personale addetto al reparto marketing della società	ID_Addetto, Ruolo	Offerte, Personale

Glossario dei termini

Entità	Descrizione	Sinonimi
Personale	Membri totali della società	Organigramma
Patente	Describe tutte le info riguardanti la patente degli autisti	Licenza di Guida
Offerte	Serie di offerte che vengono proposte al singolo utente	Promozioni
Manutentori	Addetti alla manutenzione delle auto degli autisti	Meccanici, Operai
Autisti	Personale che svolge il ruolo di autista delle auto nella società	Driver
Veicoli	Auto utilizzate per il servizio di taxi	Automobili
Turni	Turni lavorativi che riguardano gli autisti	Orario Lavorativo
Richiesta Prenotazione	Richieste di prenotazioni effettuate da parte dell'utente	Prenotazioni
Utenti	Utenti utilizzatori del servizio taxi	Persone
Feedback	Recensioni lasciate dall'utente e dagli autisti	Recensioni
Tratte Completate	Corse effettuate portate a termine con successo	Corse
Tratte Rifiutate	Corse rifiutate da parte dell'autista per determinati motivi	Corse Annullate
Carta	Carta di credito personale dell'utente	Metodo di pagamento
Assicurazioni	Dati dell'assicurazione associata al singolo veicolo	RCA, Polizza assicurativa
Addetti Marketing	Personale addetto al reparto marketing della società	Advertiser

Glossario delle relazioni

Relazione	Descrizione	Entità
VeicoloPossiedeAssicurazione	Relazione che dice che ogni veicolo ha una propria assicurazione	Veicoli (1,1), Assicurazione (1,1)
AutistaGuidaVeicolo	Relazione che dice che ogni autista guida la propria autovettura	Autisti (1,1), Veicoli (1,1)
AutistaPossiedePatente	Relazione che dice che ogni autista, per poter guidare, necessita di una patente	Autisti (1,1), Patente (1,1)
ContattaPerGuasto	Relazione che dice che ogni autista può (non necessariamente) contattare un manutentore per un guasto al veicolo	Autisti (0,N), Manutentori (0,N)
AutistaLasciaFeedback	Relazione che dice che ogni autista può lasciare uno o più feedback relativio a tutti gli aspetti della corsa effettuata	Autisti (1,N), Feedback (1,1)
AssegantoA	Relazione che dice che ogni autista viene assegnato ad una richiesta di prenotazione, in base a determinate circostanze	Autisti (1,1), Richiesta Prenotazione (1,1)
AggiungeOfferta	Relazione che dice che un addetto marketing può aggiungere una o più offerte per gli utenti	Addetti Marketing (1,N), Offerte (1,1)

Relazione	Descrizione	Entità
UtenteHaOfferta	Relazione che dice che ogni utente può avere (non necessariamente) una o più offerte attive	Utenti (1,1), Offerte (1,N)
UtentePossiedeCarta	Relazione che dice che ogni utente deve possedere almeno una carta con cui effettuare i pagamenti	Utenti (1,N), Carta (1,1)
EffettuaPrenotazione	Relazione che dice che ogni utente effettua una o più prenotazioni	Utenti (1,N), Richiesta Prenotazioni (1,1)
UtenteLasciaFeedback	Relazione che dice che ogni utente può lasciare uno o più feedback relativi alle corse da lui effettuate	Utenti (1,N), Feedback (1,1)
CartaPagaTratta	Relazione che dice che ogni utente, tramite la propria carta, deve pagare le tratte da lui effettuate	Carta (1,N), Tratte Completate (1,1)
TrattaAvereFeedback	Relazione che dice che ogni tratta completata può avere (non necessariamente) un solo feedback, che viene lasciato dagli utenti e dagli autisti	Tratte Completate (0,1), Feedback (1,1)
AutistaAvereTurni	Ogni autista ha un proprio turno lavorativo, ad ogni turno lavorativo vengono assegnati uno o più autisti	Autisti (1,1), Turni (1,N)

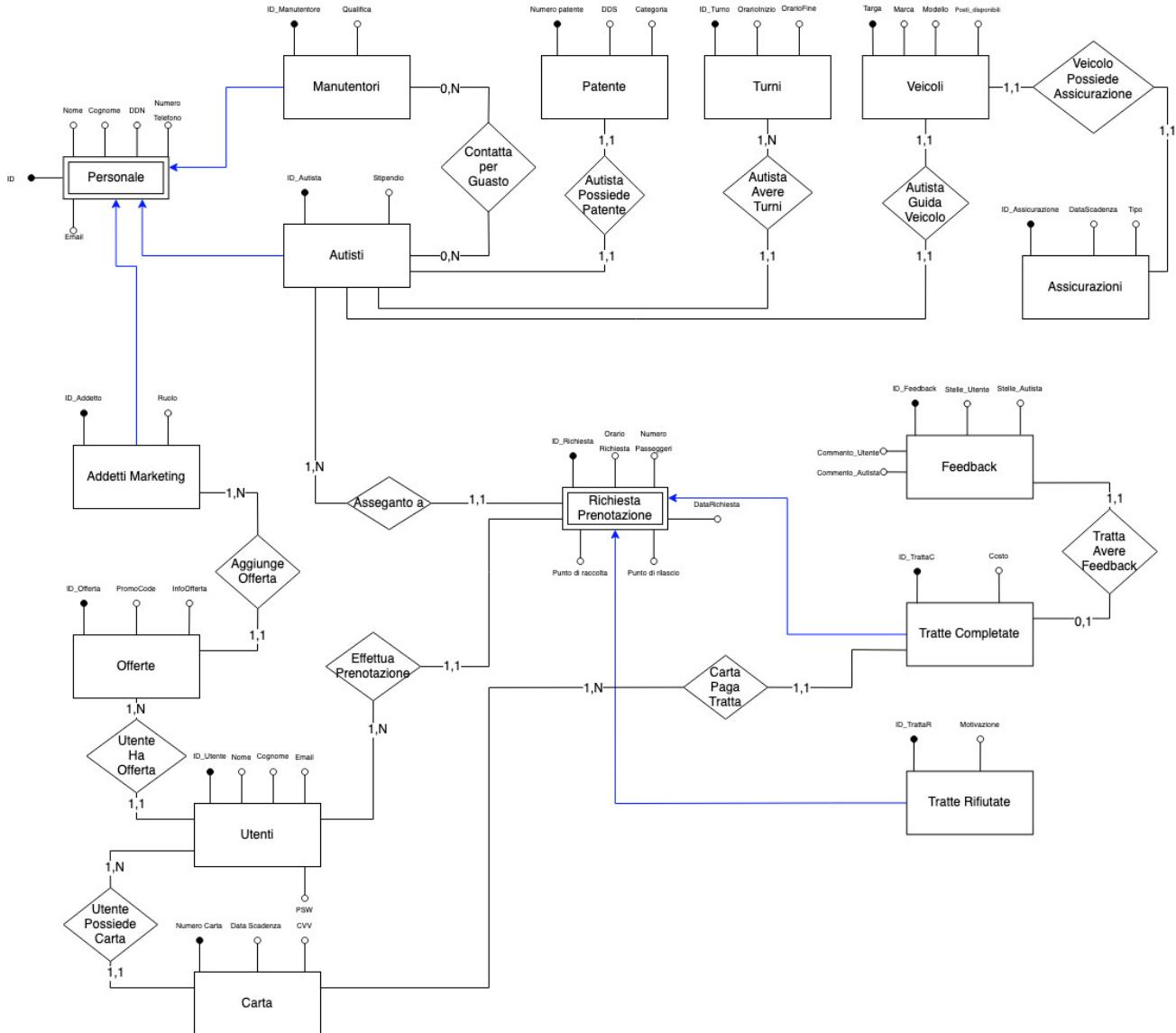
Schemi

Schemi di relazione

Le chiavi primarie sono identificate in **grassetto**, mentre le chiavi secondarie (o esterne) sono scritte in stile *Italic*

- Personale (**ID_Personale**, Nome, Cognome, NumeroTelefono, Email, DDN)
- Autisti (**ID_Autista**, Stipendio, *NumeroPatente*, *Targa*, *ID_Turno*)
- Manutentori (**ID_Manutentore**, Qualifica)
- Addetti Marketing (**ID_Addetto**, Ruolo)
- ContattaPerGuasto (*ID_Manutentore*, *ID_Autista*, Motivo)
- Patente (**NumeroPatente**, DDS, Categoria)
- Turni (**ID_Turno**, OrarioInizio, OrarioFine)
- Veicoli (**Targa**, Marca, Modello, PostiDisponibili, *ID_Assicurazione*)
- Assicurazioni (**ID_Assicurazione**, DataDiScadenza, Tipo)
- Offerte (**ID_Offerta**, PromoCode, InfoOfferta, *ID_Addetto*)
- Utenti (**ID_Utente**, Nome, Cognome, Email, PSW, *ID_Offerta*)
- Carta (**NumeroCarta**, DataScadenza, CVV, *ID_Utente*)
- Richiesta Prenotazione (**ID_Richiesta**, DataRichiesta, OrarioRichiesta, NumeroPasseggeri, PuntoRaccolta, PuntoRilascio, *ID_Utente*, *ID_Autista*)
- Tratte Complete (**ID_Tratta**, Costo, *NumeroCarta*)
- Tratte Rifiutate (**ID_Tratta**, Motivazione)
- Feedback (**ID_Feedback**, StelleUtente, CommentoUtente, StelleAutista, CommentoAutista, *ID_TrattaCompletata*)

Schema Logico



Normalizzazione

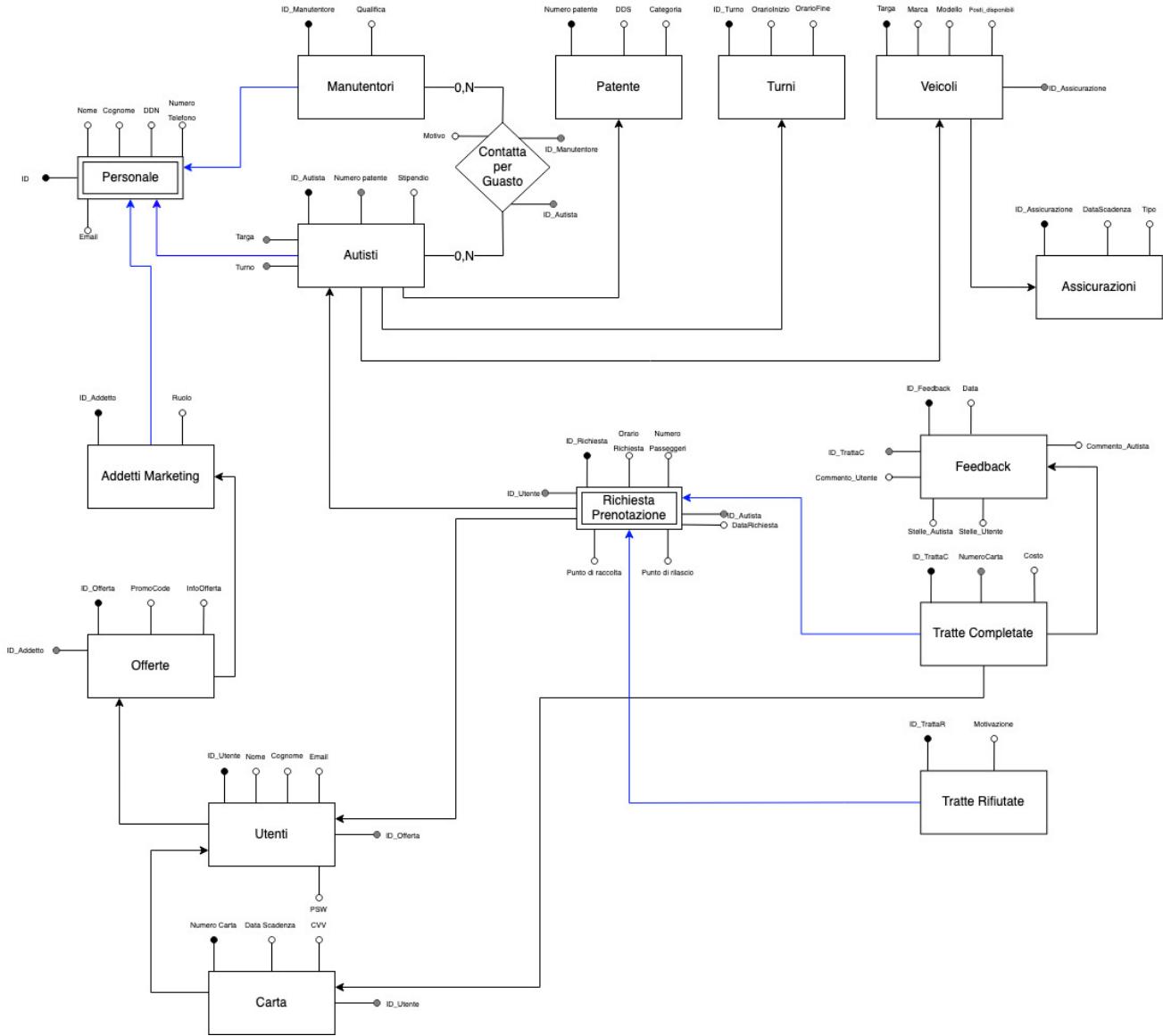
Di seguito si discutono le forme normali dello schema logico:

- **1NF**: tutti gli schemi di relazione nello schema logico sopra riportato sono in **1NF**, poiché tutti gli attributi sono semplici, ovvero contengono soltanto valori atomici indivisibili.
- **2NF**: tutti gli schemi di relazione dello schema logico sono anche già in **2NF**, poiché sono già in **1NF** e nessun attributo presenta alcuna dipendenza parziale. Tutti gli attributi dipendono funzionalmente solo dalla chiave primaria della stessa tabella.
- **3NF**: tutti gli schemi di relazione sono anche in **3NF** perché già in **2NF**, ed inoltre, tutti gli attributi delle tabelle dipendono funzionalmente e direttamente dalla chiave primaria, senza transitività.

Schema Fisico

Abbiamo distinto le frecce che vanno dalle entità figlie a quelle padre mettendole in blu.

Nelle entità, le chiavi secondarie sono indentificate con il pallino grigio, mentre quelle primarie sono indentificate con il pallino nero.



Generalizzazione

Una generalizzazione rappresenta un legame logico tra un'entità genitore e una o più entità figlie, in questo caso le entità genitore sono "Personale" e "Richiesta Prenotazione", ognuna con le rispettive entità figlie, che sono:

1. Personale

1. Autisti
2. Addetti Marketing
3. Manutentori

2. Richiesta prenotazione

1. Tratte completate
2. Tratte rifiutate

Abbiamo tre metodi per rappresentare una generalizzazione a livello fisico:

- Accorpamento del padre nelle entità figlie
- Accorpamento delle entità figlie nel padre
- Sostituzione della generalizzazione con relazioni

Tra questi metodi abbiamo scelto il terzo in quanto da noi considerato il più adeguato. Infatti, il primo metodo avrebbe portato ad una ridondanza di relazioni.

Il secondo metodo necessita dell'aggiunta di un attributo nelle entità "Personale" e "Richiesta Prenotazioni", con il compito di specificare il ruolo del lavoratore (Es. Autisti = 1, Manutentori = 2, etc..), e il tipo di prenotazione (Es. Completata = 1 e Rifiutata = 2), in più si sarebbe dovuto scegliere se perdere informazioni (attributi) dei figli o inserire le informazioni nel padre, quindi aggiungere attributi dei figli al padre. La seconda scelta avrebbe portato ad una quantità non indifferente di valori NULL.

Implementazione Database - MySQL

Creazione delle tabelle

```
CREATE TABLE Personale(
    ID int not null ,
    Nome varchar(50) not null,
    Cognome varchar(50) not null,
    DDN date not null,
    NumeroDiTelefono varchar(50) not null,
    Email varchar(255),
    PRIMARY KEY (ID)
);

CREATE TABLE AddettiMarketing (
    ID_Adetto int not null,
    Ruolo varchar(50),
    PRIMARY KEY (ID_Adetto),
    FOREIGN KEY (ID_Adetto) REFERENCES Personale(ID)
);

CREATE TABLE Patente (
    NumeroPatente varchar(50) not null,
    DDS date not null,
    Categoria varchar(50),
    PRIMARY KEY (NumeroPatente)
);

CREATE TABLE Offerte (
    ID_Offerta int not null ,
    PromoCode int not null,
    InfoOfferta varchar(50) not null,
    ID_Adetto int not null,
    PRIMARY KEY (ID_Offerta),
    FOREIGN KEY (ID_Adetto) REFERENCES AddettiMarketing(ID_Adetto)
);

CREATE TABLE Manutentori (
    ID_Manutentore int not null ,
    Qualifica varchar(50) not null,
    PRIMARY KEY (ID_Manutentore),
    FOREIGN KEY (ID_Manutentore) REFERENCES Personale (ID)
);

CREATE TABLE Assicurazioni (
    ID_Assicurazione int not null ,
    DataScadenza date not null,
    Tipo varchar(50) not null,
    PRIMARY KEY (ID_Assicurazione)
);

CREATE TABLE Veicoli (
    Targa varchar(50) not null,
    Marca varchar(50) not null,
```

```

    Modello varchar(50) not null,
    PostiDisponibili int not null,
    Assicurazione int not null,
    PRIMARY KEY (Targa),
    FOREIGN KEY (Assicurazione) REFERENCES Assicurazione(ID_Assicurazione)
);
CREATE TABLE Turni (
    ID_Turno int not null ,
    OrarioInizio int not null ,
    OrarioFine int not null ,
    PRIMARY KEY (ID_Turno)
);
CREATE TABLE Autisti (
    ID_Autista int not null ,
    NumeroPatente varchar(50) not null ,
    Turno int not null ,
    Targa varchar(50) not null ,
    Stipendio int not null ,
    PRIMARY KEY (ID_Autista),
    FOREIGN KEY (ID_Autista) REFERENCES Personale (ID),
    FOREIGN KEY (NumeroPatente) REFERENCES Patente(NumeroPatente),
    FOREIGN KEY (Turno) REFERENCES Turni(ID_Turno),
    FOREIGN KEY (Targa) REFERENCES Veicoli(Targa)
);
CREATE TABLE Utenti (
    ID_Utente int not null ,
    Nome varchar(50) not null ,
    Cognome varchar(50) not null ,
    Email varchar(255) not null ,
    Password varchar(255) not null ,
    ID_Offerta int not null ,
    PRIMARY KEY (ID_Utente),
    FOREIGN KEY (ID_Offerta) REFERENCES Offerte (ID_Offerta)
);

CREATE TABLE RichiestePrenotazioni (
    ID_Richiesta int not null ,
    PuntoDiRaccolta varchar(50) not null ,
    PuntoDiRilascio varchar(50) not null ,
    DataRichiesta date not null ,
    OrarioRichiesta varchar(50) not null ,
    NumeroPasseggeri int not null ,
    ID_Utente int not null ,
    ID_Autista int not null ,
    PRIMARY KEY (ID_Richiesta),
    FOREIGN KEY (ID_Utente) REFERENCES Utenti(ID_Utente),
    FOREIGN KEY (ID_Autista) REFERENCES Autisti(ID_Autista)
);

CREATE TABLE Carta (
    NumeroCarta varchar(50) not null ,
    DataScadenza date not null ,
    CVV int not null ,
    ID_Utente int not null ,
    PRIMARY KEY (NumeroCarta),

```

```

    FOREIGN KEY (ID_Utente) REFERENCES Utenti(ID_Utente)
);

CREATE TABLE TratteCompletate (
    ID_TrattaC int not null ,
    Costo int not null,
    NumeroCarta varchar(50) not null,
    PRIMARY KEY (ID_TrattaC),
    FOREIGN KEY (ID_TrattaC) REFERENCES RichiestePrenotazioni (ID_Richiesta),
    FOREIGN KEY (NumeroCarta) REFERENCES Carta (NumeroCarta)
);

CREATE TABLE Feedback (
    ID_Feedback int not null ,
    StelleUtente int not null,
    CommentoUtente varchar(255) not null,
    StelleAutista int not null,
    CommentoAutista varchar(255) not null,
    ID_TrattaCompleta int not null,
    PRIMARY KEY (ID_Feedback),
    FOREIGN KEY (ID_TrattaCompleta) REFERENCES TratteCompletate (ID_TrattaC)
);

CREATE TABLE TratteRifiutate (
    ID_TrattaR int not null ,
    Motivazione varchar(255) not null,
    PRIMARY KEY (ID_TrattaR),
    FOREIGN KEY (ID_TrattaR) REFERENCES RichiestePrenotazioni (ID_Richiesta)
);

CREATE TABLE ContattaPerGuasto (
    ID_Manutentore int not null,
    ID_Autista int not null,
    Motivo varchar(255) not null,
    FOREIGN KEY (ID_Manutentore) REFERENCES Manutentori (ID_Manutentore),
    FOREIGN KEY (ID_Autista) REFERENCES Autisti (ID_Autista)
);

```

Triggers

I trigger fanno parte del DDL (Data Definition Language), essi seguono il principio ECA, ovvero Event-Condition-Action. Solitamente, un trigger si può attivare prima o dopo un inserimento e hanno 2 livelli di granularità:

1. attivarsi per ogni tupla
2. attivarsi per ogni istruzione DML

Nello specifico in MySQL i trigger operano a livello di riga e si ammette un solo trigger per tabella. Osserviamo inoltre che questi vengono usati per mantenere constraint di ogni tipo, in primis il vincolo di integrità referenziale. Quelli di seguito sono una serie di trigger di esempio necessari per mantenere una serie di vincoli nel nostro database

Controlla Orario Richiesta

```

CREATE TRIGGER `ControllaOrarioRichiesta` BEFORE INSERT ON `RichiestePrenotazioni` FOR EACH ROW BEGIN

-- Controlla se l'orario richiesto esiste già nella tabella
    IF EXISTS (
        SELECT 1
        FROM RichiestePrenotazioni
        WHERE OrarioRichiesta = NEW.OrarioRichiesta AND ID_Utente = NEW.ID_Utente
    ) THEN

        -- Se l'orario esiste già, interrompi l'inserimento
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '[ERRORE],OGNI UTENTE NON PUÒ PRENOTARE PIÙ DI UNA
CORSA NELLO STESSO ORARIO';
    END IF;

END

```

Controlla Turno Lavorativo

```

CREATE TRIGGER `ControllaTurnoLavorativo` BEFORE INSERT ON `Autisti` FOR EACH ROW BEGIN

-- Controlla se l'autista è stato già assegnato al turno richiesto
    IF EXISTS (
        SELECT 1
        FROM Autisti a
        WHERE Turno = NEW.Turno AND ID_Autista = NEW.ID_Autista
    ) THEN

        -- Se si, interrompi l'inserimento
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '[ERRORE],UN AUTISTA NON PUÒ ESSERE ASSEGNATO A DUE
TURNI CHE HANNO LO STESSO ORARIO DI INIZIO E FINE';
    END IF;

END

```

Controlla Carta

```

CREATE TRIGGER `ControllaInserimentiCarta` BEFORE INSERT ON `Carta` FOR EACH ROW BEGIN

-- Controlla se la carta è stato già inserita
    IF EXISTS (
        SELECT 1
        FROM Carta c
        WHERE NumeroCarta = NEW.NumeroCarta
    ) THEN

        -- Se si, interrompi l'inserimento
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '[ERRORE],LA CARTA CHE SI VUOLE AGGIUNGERE È GIÀ
PRESENTE NEL DATABASE';

    END IF;

```

END

Inserimenti

Di seguito vengono riportati alcuni estratti di query per l'inserimento, presi dallo script di creazione automatica delle query

Personale

```
INSERT INTO Personale (ID, Nome, Cognome, DDN, NumeroDiTelefono, Email)
VALUES ('0', 'Alessandra', 'Turci', '1998-01-10', '3367627622', 'Alessandra.Turci@gradenigo.com'),
('1', 'Maria', 'Andreotti', '1976-02-25', '+39 340517754', 'Maria.Andreotti@sgalambro.org'),
('2', 'Elena', 'Toscanini', '1999-04-22', '034354072', 'Elena.Toscanini@casarin.com'),
('3', 'Lorenzo', 'Toscani', '1986-07-18', '0185195557', 'Lorenzo.Toscani@vittadello-franscini.com'),
('4', 'Paride', 'Ferragni', '1982-06-09', '37608479136', 'Paride.Ferragni@tarantino-giannuzzi.net'),
('5', 'Sante', 'Biagi', '1987-10-22', '+39 035138406', 'Sante.Biagi@boccaccio.it'),
('6', 'Renata', 'Gonzaga', '1982-05-22', '0308099892', 'Renata.Gonzaga@gagliardi.eu'),
('7', 'Gaetano', 'Lombroso', '1987-01-08', '04318152253', 'Gaetano.Lombroso@ostinelli-ammaniti.it'),
('8', 'Claudio', 'Bacosi', '1981-08-05', '03555787953', 'Claudio.Bacosi@broschi-pignatti.it'),
('9', 'Arnulfo', 'Borromini', '1975-04-28', '092590091', 'Arnulfo.Borromini@gaito-branciforte.com'),
('10', 'Rosalia', 'Giulietti', '1976-10-12', '0357716989', 'Rosalia.Giulietti@renzi.it'),
('11', 'Durante', 'Ughi', '1998-07-01', '3783935849', 'Durante.Ughi@busoni.eu'),
('12', 'Annamaria', 'Travia', '1997-12-29', '+39 0571009017', 'Annamaria.Travia@bresciani.it'),
('13', 'Guarino', 'Salvini', '1995-05-03', '+39 37138530579', 'Guarino.Salvini@santoro.it'),
('14', 'Eugenia', 'Prati', '1995-05-26', '+39 0375042347', 'Eugenia.Prati@buscetta-ferraris.it'),
('15', 'Virgilio', 'Doglioni', '1990-01-22', '+39 3971845809', 'Virgilio.Doglioni@montecchi.com'),
('16', 'Goffredo', 'Botta', '1999-08-22', '3711811676', 'Goffredo.Botta@napolitano.com'),
('17', 'Raffaellino', 'Strangio', '1983-04-01', '+39 0163724485', 'Raffaellino.Strangio@zamorani.com'),
('18', 'Mariana', 'Viviani', '1994-06-11', '+39 37153448788', 'Mariana.Viviani@guariento-aulenti.net'),
('19', 'Valerio', 'Caffarelli', '1999-10-15', '+39 058506278', 'Valerio.Caffarelli@lussu.it'),
('20', 'Gianfranco', 'Oscuro', '1980-09-26', '+39 0141131737', 'Gianfranco.Oscuro@turchetta.it')
```

Addetti Marketing

```
INSERT INTO AddettiMarketing (ID_Addetto, Ruolo) VALUES
('5900', 'Responsabile'),
('5901', 'Responsabile'),
('5902', 'Analista'),
('5903', 'Responsabile'),
('5904', 'Coordinatore'),
('5905', 'Coordinatore'),
('5906', 'Responsabile'),
('5907', 'Analista'),
('5908', 'Coordinatore'),
```

```
('5909','Responsabile'),  
('5910','Responsabile'),  
('5911','Responsabile'),  
('5912','Coordinatore'),  
('5913','Coordinatore'),  
('5914','Responsabile'),  
('5915','Responsabile'),  
('5916','Analista'),  
('5917','Coordinatore'),  
('5918','Analista'),  
('5919','Analista'),
```

Patente

```
INSERT INTO Patente (NumeroPatente,DDS,Categoria) VALUES  
('1VZAPG5HF','2030-07-10','BE'),  
('1GI06ZZN8','2026-04-06','B96'),  
('SIHTMV046','2026-03-08','B96'),  
('7XDZBD28P','2029-09-05','BE'),  
('MI1VXVJE2','2031-04-09','B96'),  
('LNDKURWCJ','2026-04-13','B96'),  
('ARTNO60B4','2030-11-28','BE'),  
('F4AHC35K1','2025-09-23','B96'),  
('T45K1B5CD','2026-06-19','B'),  
('GBC716IUZ','2027-07-31','B96'),  
('0RIBPMS0S','2027-09-25','B'),  
('7XF1NCD4P','2035-04-08','B'),  
('5UDTLYT7S','2034-11-13','B96'),  
('6CZMPX888','2034-01-02','BE'),
```

Turni

```
IINSERT INTO Turni (ID_Turno,OrarioInizio,OrarioFine) VALUES  
('0','9','17'),  
('1','11','22'),  
('2','10','21'),  
('3','9','22'),  
('4','14','17');
```

Assicurazioni

```
INSERT INTO Assicurazioni (ID_Assicurazione,DataScadenza,Tipo) VALUES  
('0','2024-07-21','Incendio'),  
('1','2023-07-27','Incendio'),  
('2','2023-11-03','Kasko'),  
('3','2024-12-06','Incendio'),  
('4','2023-11-22','Furto'),  
('5','2024-01-27','Furto'),  
('6','2023-11-17','Furto'),  
('7','2023-09-27','Base'),  
('8','2023-09-23','Incendio'),  
('9','2024-03-30','Kasko'),  
('10','2023-06-25','Base'),  
('11','2023-01-26','Kasko'),  
('12','2023-10-31','Base'),  
('13','2023-02-19','Furto'),
```

```
('14','2023-09-28','Base'),
('15','2023-01-10','Base'),
('16','2024-03-16','Incendio'),
('17','2023-11-20','Base'),
('18','2023-06-17','Base'),
('19','2024-02-25','Base'),
('20','2023-11-10','Kasko'),
```

Veicoli

```
INSERT INTO Veicoli (Targa,Marca,Modello,PostiDisponibili,ID_Assicurazione) VALUES
('SC670UV','Audi','RS7','2','0'),
('AY173TM','BMW','Panda','11','1'),
('ZD988ED','Audi','Q8','2','2'),
('PS408RD','Fiat','Panda','5','3'),
('VS694BO','Audi','Panda','2','4'),
('BB5820Y','Audi','Punto','3','5'),
('CM575GU','Fiat','Punto','10','6'),
('YR313NC','Fiat','Q8','3','7'),
('AF288JI','BMW','Q8','4','8'),
('CF779IO','Fiat','RS7','6','9'),
('QX126ME','Range Rover','Q8','12','10'),
('WK748BW','BMW','Q8','3','11'),
('BL407OS','Range Rover','RS7','5','12'),
('OR153EO','BMW','Punto','1','13'),
('YX590FW','Range Rover','RS7','8','14'),
('PW990ED','Audi','RS7','3','15'),
('UH792HC','BMW','Q8','8','16'),
('XV292AR','Seat','Q8','2','17'),
('LX404VB','Audi','Panda','1','18'),
('NG689AU','Seat','Panda','12','19'),
```

Autisti

```
INSERT INTO Autisti (ID_Autista,NumeroPatente,Turno,Targa,Stipendio) VALUES
('0','1VZAPG5HF','3','QF3650B','1200'),
('1','1GIO6ZZN8','2','ZW508CB','800'),
('2','SIHTMV046','1','VS694BO','1200'),
('3','7XDZBD28P','2','AD555MK','1200'),
('4','MI1VXXJE2','3','US482YF','800'),
('5','LNDKURWCJ','1','XJ474EH','1100'),
('6','ARTNO60B4','2','XF186JW','1100'),
('7','F4AHC35K1','1','RX184KI','900'),
('8','T45K1B5CD','1','FE998XV','1100'),
('9','GBC716IUZ','2','FL184DP','1100'),
('10','0RIBPMS0S','2','NR192MK','1100'),
('11','7XF1NCD4P','2','VT628VM','900'),
('12','5UDTLYT7S','3','HZ030UQ','1100'),
('13','6CZMPX888','1','YJ817VA','800'),
('14','FLU0C3Y7N','4','EC6560S','900'),
('15','0VJ5IZUF4','1','VS897AB','900'),
('16','FCVLNXPZA','3','GY188RG','800'),
('17','OWVH2ITCE','2','TU882AL','800'),
('18','0J8RQR58D','1','HS782WY','900'),
('19','0F83LL0NU','3','ZZ498WK','800'),
('20','YSE3RS33J','3','VG466BR','1100'),
```

Manutentori

```
INSERT INTO Manutentori (ID_Manutentore,Qualifica) VALUES ('3000','Carrozziere'),  
('3001','Gommista'),  
('3002','Gommista'),  
('3003','Carrozziere'),  
('3004','Elettrauto'),  
('3005','Gommista'),  
('3006','Meccanico'),  
('3007','Elettrauto'),  
('3008','Gommista'),  
('3009','Carrozziere'),  
('3010','Meccanico'),  
('3011','Carrozziere'),  
('3012','Meccanico'),  
('3013','Meccanico'),  
('3014','Carrozziere'),  
('3015','Carrozziere'),  
('3016','Carrozziere'),  
('3017','Meccanico'),  
('3018','Carrozziere'),  
('3019','Carrozziere'),
```

ContattaPerGuasto

```
INSERT INTO ContattaPerGuasto (ID_Manutentore, ID_Autista, Motivo) VALUES  
('3303','162','Specchietto rotto'),  
('3437','63','Radiatore bucato'),  
('4525','10','Errore centralina'),  
('4824','194','Radiatore bucato'),  
('3366','105','Semiasse distrutto'),  
('5492','66','Spia dell motore accesa'),  
('5810','148','La macchina non parte'),  
('3252','29','Errore centralina'),  
('4064','131','Radiatore bucato'),  
('4431','34','Semiasse distrutto'),  
('4797','105','La macchina non parte'),  
('5322','50','Spia dell motore accesa'),  
('3278','90','Semiasse distrutto'),  
('3679','14','Spia dell motore accesa'),  
('4430','147','Semiasse distrutto'),  
('3519','25','Rottura degli ammortizzatori'),  
('3201','179','Differenziale rotto'),  
('4127','91','Errore centralina'),  
('3907','145','Specchietto rotto'),  
('3652','12','Problema con il FAP'),  
('5179','114','Gomma Bucata'),  
('3404','2','Semiasse distrutto'),
```

Offerte

```
INSERT INTO Offerte (ID_Offerta, PromoCode, InfoOfferta, ID_Addetto) VALUES  
('0','273824','Credito 5€','5907'),  
('1','364933','Sconto 15%','5969'),  
('2','136714','Credito 10€','5942'),  
('3','655866','Credito 5€','5970'),
```

```
('4','931497','Credito 10€','5966'),  
('5','624579','Credito 5€','5998'),  
('6','295792','Sconto 20%','5900'),  
('7','488267','Credito 10€','5952'),  
('8','752354','Credito 10€','5941'),  
('9','436112','Credito 10€','5961'),  
('10','635915','Sconto 15%','5994'),  
('11','292342','Sconto 10%','5909'),  
('12','139624','Sconto 20%','5961'),  
('13','666711','Credito 5€','5920'),  
('14','716143','Sconto 20%','5983');
```

Utenti

```
INSERT INTO Utenti (ID_Utente, Nome, Cognome, Email, Password, ID_Offerta) VALUES  
('0','Geronimo','Lucarelli','Geronimo.Lucarelli@ligorio.it','UllKLuaR','12'),  
('1','Eva','Montesano','Eva.Montesano@morandi.it','s7w7DABQr','10'),  
('2','Umberto','Pizzo','Umberto.Pizzo@fioravanti.net','TUyC3gN2f','12'),  
('3','Licia','Interminelli','Licia.Interminelli@manunta-tasso.com','1wbysw17I','10'),  
('4','Atenulf','Alfonsi','Atenulf.Alfonsi@liguori.it','M2Wuw90jZ','12'),  
('5','Carla','Cadorna','Carla.Cadorna@goldstein-troisi.it','LLpEc0ipa','5'),  
('6','Marissa','Cavalcanti','Marissa.Cavalcanti@faugno.it','pxKgaxGJ4','1'),  
('7','Manuel','Mogherini','Manuel.Mogherini@chindamo.net','MvvB8NhDy','10'),  
('8','Stella','Tencalla','Stella.Tencalla@toscanini-palladio.it','tnNk04ipx','8'),  
('9','Venancio','Sgarbi','Venancio.Sgarbi@dellucci.net','b0Q7Qk4Zw','8'),  
('10','Donna','Tassoni','Donna.Tassoni@luciani.it','L7r4biloI','9'),  
('11','Piero','Marinetti','Piero.Marinetti@travaglio-ferrucci.it','Mq2D9oi3h','4'),  
('12','Mario','Colletti','Mario.Colletti@lattuada.com','of1QG204m','6'),  
('13','Silvia','Giulietti','Silvia.Giulietti@asprucci-tozzi.com','m1Fg0hvY5','5'),  
('14','Pomponio','Scarpa','Pomponio.Scarpa@castellitto.com','Esio46MMH','13'),  
('15','Leopoldo','Guariento','Leopoldo.Guariento@armellini.com','SIA2coy92','3'),  
('16','Viridiana','Beffa','Viridiana.Beffa@travaglia.com','LvmqCIG2Y','1'),  
('17','Fausto','Gualtieri','Fausto.Gualtieri@cristoforetti.org','ndvVU7RdX','6'),  
('18','Sabatino','Comisso','Sabatino.Comisso@virgilio.org','GcJhrfI47','3'),  
('19','Adelasia','Pavanello','Adelasia.Pavanello@santoro.com','JJUn7EmBU','3'),  
('20','Germana','Doglioni','Germana.Doglioni@foletti.it','wJsuPsgQv','3'),
```

Carta

```
INSERT INTO Carta (NumeroCarta, DataScadenza, CVV, ID_Utente) VALUES  
('5516 6245 1261 0132','2030-10-18','479','0'),  
('4757 0060 3749 3269','2031-01-15','850','1'),  
('4147 7481 0581 8262','2033-02-16','871','2'),  
('4241 7064 9449 0308','2027-05-14','909','3'),  
('4130 0669 8284 2854','2030-01-19','573','4'),  
('4767 3839 2746 2434','2029-04-20','340','5'),  
('4381 7526 0880 0555','2033-01-19','228','6'),  
('5648 4762 3809 8495','2032-09-17','883','7'),  
('5399 5927 8149 2142','2029-07-13','475','8'),  
('4191 1402 6218 6567','2032-02-16','420','9'),  
('4909 5358 3453 7202','2030-05-20','248','10'),  
('5684 5971 5717 0779','2029-10-27','574','11'),  
('4334 5438 9255 3174','2029-08-21','017','12'),  
('5349 9153 1786 1035','2029-05-17','280','13'),  
('4399 5092 3169 1653','2032-04-08','884','14'),  
('4238 1558 8766 1848','2032-05-03','735','15'),  
('4092 3571 3675 4525','2031-11-17','452','16'),
```

```
('5892 6823 9691 1048','2027-02-18','391','17'),  
('4528 2796 0798 3218','2034-08-21','453','18'),  
('5280 7228 9635 4347','2028-04-23','090','19'),  
('4292 6680 9680 7438','2029-06-05','773','20'),  
('5949 3770 6304 7987','2030-10-27','700','21'),  
('4597 3672 2445 1647','2031-05-29','484','22'),  
('5126 4874 4373 9777','2028-05-03','603','23'),
```

Richieste Prenotazioni

```
INSERT INTO RichiestePrenotazioni  
(ID_Richiesta,PuntoDiRaccolta,PuntoDiRilascio,DataRichiesta,OrarioRichiesta,NumeroPasseggeri,  
ID_Utente,ID_Autista) VALUES  
('0','Tor Vergata','San Lorenzo','2023-12-16','22','4','2953','2206'),  
('1','Centocelle','San Lorenzo','2023-01-08','20','4','2019','783'),  
('2','Tor Vergata','San Lorenzo','2023-12-01','11','4','2114','1682'),  
('3','Termini','Garbatella','2022-07-01','22','6','2725','1767'),  
('4','Centocelle','Finocchio','2023-10-10','22','9','1668','2550'),  
('5','Centocelle','Finocchio','2023-06-04','20','3','3890','448'),  
('6','Eur','San Basilio','2023-12-10','14','6','1799','1478'),  
('7','Colosseo','Garbatella','2022-03-24','10','4','1186','1816'),  
('8','Tor Vergata','Finocchio','2022-09-23','21','8','4653','585'),  
('9','Eur','San Basilio','2022-03-10','9','6','3440','694'),  
('10','Colosseo','Finocchio','2022-04-25','21','7','583','1208'),  
('11','Termini','San Basilio','2023-11-23','14','8','4143','2366'),  
('12','Colosseo','San Basilio','2022-08-12','20','8','2857','554'),  
('13','Centocelle','Primavalle','2023-06-13','14','8','2052','1420'),  
('14','Colosseo','Garbatella','2022-11-09','11','4','3713','1738'),  
('15','Eur','Ostia','2023-08-05','11','6','961','1822'),  
('16','Tor Vergata','San Lorenzo','2023-03-16','15','6','1959','801'),  
('17','Eur','Garbatella','2023-10-14','21','12','631','1677'),  
('18','Tor Vergata','San Lorenzo','2023-06-10','10','12','1331','1928'),  
('19','Colosseo','Ostia','2022-08-30','16','8','853','1029'),  
('20','Centocelle','Garbatella','2023-08-26','21','9','909','1922'),  
('21','Colosseo','Primavalle','2023-05-06','9','4','3118','2427'),  
('22','Anagnina','Garbatella','2022-09-30','11','11','2969','1521'),  
('23','Eur','Finocchio','2022-10-29','16','7','109','782'),  
('24','Colosseo','Primavalle','2022-06-30','9','11','4021','677'),  
('25','Termini','Finocchio','2023-10-24','16','6','3443','2174'),  
('26','Tor Vergata','Garbatella','2023-09-05','15','2','2704','584'),  
('27','Anagnina','San Lorenzo','2022-09-09','14','12','912','2570'),  
('28','Anagnina','Primavalle','2022-01-17','16','12','3096','1222'),  
('29','Anagnina','Ostia','2022-11-17','22','3','4189','2245'),  
('30','Tor Vergata','San Basilio','2023-08-26','16','4','95','1239'),
```

Tratte Completate

```
INSERT INTO TratteCompletate(ID_TrattaC,Costo,NumeroCarta) VALUES ('0','50','4089 5151  
5662 4069'),  
('1','25','5047 0846 1593 2618'),  
('2','50','4061 4269 4847 7335'),  
('3','115','5318 1436 0940 0684'),  
('4','50','5975 4893 1583 8147'),  
('5','65','5684 5078 3830 4961'),  
('6','25','5366 8615 7812 6293'),  
('7','50','4722 5730 3302 9435'),  
('8','35','4543 4190 3841 6987'),
```

```
('9','35','5480 6038 6779 5241'),  
('10','115','5855 4204 7208 8110'),  
('11','25','4341 8786 4075 8180'),  
('12','50','4576 1500 6391 0947'),  
('13','25','5037 0899 6806 9872'),  
('14','25','4045 9340 1401 7460'),  
('15','50','5708 7914 3067 7696'),  
('16','50','4282 0309 1668 9769'),  
('17','25','4868 5174 4338 4566'),  
('18','50','4955 5482 5347 8387'),  
('19','65','4733 5758 5742 4031'),  
('20','65','4116 6449 8486 0800'),  
('21','25','4826 7536 1058 0146'),  
('22','115','5941 4951 3635 7089'),
```

Feedback

```
INSERT INTO  
Feedback(ID_Feedback,StelleUtente,CommentoUtente,StelleAutista,CommentoAutista,ID_TrattaCompletata) VALUES  
('0','1','Non lo prenderò mai più!', '1', 'L utente offende', '7125'),  
('1','3','Tutto nella norma', '3', 'Utente ok', '10044'),  
('2','2','La prossima volta preferirei un altro autista', '2', 'Non rispetta l autista', '13925'),  
('3','3','Nulla di particolare', '3', 'Utente ok', '1942'),  
('4','1','Esperienza orribile', '1', 'Utente scortese!', '1666'),  
('5','1','Non lo prenderò mai più!', '1', 'L utente offende', '1417'),  
('6','4','Veicolo molto pulito e comodo.', '4', 'Utente gentile', '1418'),  
('7','1','Non lo prenderò mai più!', '1', 'L utente offende', '4084'),  
('8','4','Esperienza normale', '4', 'Utente gentile', '852'),  
('9','1','Non lo prenderò mai più!', '1', 'Utente scortese!', '12330'),  
('10','5','Autista veramente cordiale', '5', 'Molto bravo e cortese', '12209'),  
('11','5','Ottima esperienza, lo dirò a tutti', '5', 'Molto bravo e cortese', '10205'),  
('12','2','La prossima volta preferirei un altro autista', '2', 'Non rispetta l'autista', '10421'),  
('13','4','Esperienza normale', '4', 'Utente gentile', '2178'),  
('14','5','Autista veramente cordiale', '5', 'Molto bravo e cortese', '2990'),  
('15','3','Tutto nella norma', '3', 'Nulla di particolare', '766'),  
('16','1','Esperienza orribile', '1', 'Utente scortese!', '684'),  
('17','1','Non lo prenderò mai più!', '1', 'Utente scortese!', '7668'),  
('18','2','La prossima volta preferirei un altro autista', '2', 'Utente ritardatario', '11378'),  
('19','5','Ottima esperienza, lo dirò a tutti', '5', 'Utente veramente genuino', '13213'),  
('20','4','Esperienza normale', '4', 'Utente rispettoso.', '10766'),  
('21','2','Non mi è piaciuto lo stile di guida', '2', 'Non rispetta l autista', '11735'),  
('22','3','Tutto nella norma', '3', 'Nulla di particolare', '5012'),  
('23','1','Non lo prenderò mai più!', '1', 'L utente offende', '3562'),  
('24','3','Nulla di particolare', '3', 'Nulla di particolare', '11570'),  
('25','3','Tutto nella norma', '3', 'Utente ok', '8507'),  
('26','5','Autista veramente cordiale', '5', 'Molto bravo e cortese', '9198'),  
('27','1','Esperienza orribile', '1', 'Utente scortese!', '12949'),  
('28','4','Esperienza normale', '4', 'Utente rispettoso.', '7575'),  
('29','1','Esperienza orribile', '1', 'Utente scortese!', '9106'),
```

Tratte Rifiutate

```

INSERT INTO TratteRifiutate (ID_TrattaR,Motivazione) VALUES ('15000','Indisponibilità al servizio'),
('15001','Indisponibilità al servizio'),
('15002','Troppo lontano'),
('15003','Troppo lontano'),
('15004','Troppo lontano'),
('15005','Indisponibilità al servizio'),
('15006','Indisponibilità al servizio'),
('15007','Problema generale'),
('15008','Problema generale'),
('15009','Problema generale'),
('15010','Troppo lontano'),
('15011','Indisponibilità al servizio'),
('15012','Troppo lontano'),
('15013','Problema generale'),
('15014','Indisponibilità al servizio'),
('15015','Problema generale'),
('15016','Indisponibilità al servizio'),
('15017','Problema generale'),
('15018','Troppo lontano'),
('15019','Troppo lontano'),
('15020','Problema generale'),

```

Script di creazione automatica di query

Per rendere paragonabili i tempi di esecuzione delle query non ottimizzate con quelle ottimizzate, è stato necessario introdurre, nel database, un gran numero di record.

Per velocizzare questo processo, è stato scritto uno script in Python che scrive tutti gli inserimenti generati casualmente, in un semplice file di testo.

Il file è il seguente

```

import random
from faker import Faker
import string
import decimal
import datetime

fake = Faker("it_IT")

#Funzione Rand.DDN
def genRandomDate():
    start_date = datetime.date(1975, 1, 1)
    end_date = datetime.date(2001, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.DA
def genRandomInsuranceDate():
    start_date = datetime.date(2023, 1, 1)
    end_date = datetime.date(2025, 1, 1)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)

```

```

random_date = start_date + datetime.timedelta(days=rand_days)

return random_date

#Funzione Rand.CD
def genRandomCardDate():
    start_date = datetime.date(2027, 1, 1)
    end_date = datetime.date(2034, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.PD
def genRandomLicenceDate():
    start_date = datetime.date(2025, 1, 1)
    end_date = datetime.date(2035, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.DDR
def genRandomRequestDate():
    start_date = datetime.date(2022, 1, 1)
    end_date = datetime.date(2023, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.Mail
def generateEmail(name, surname):

    domain = fake.domain_name()

    return f"{name}.{surname}@{domain}"

#Funzione Rand.T
def generateTarga():

    SYMBOLS = "ABCDEFGHIJKLMNPQRSTUVWXYZ"
    NUMBERS = "0123456789"
    start = "".join(random.choice(SYMBOLS) for i in range(2))
    mezzo = "".join(random.choice(NUMBERS) for i in range(3))
    fine = "".join(random.choice(SYMBOLS) for i in range(2))

    return start+mezzo+fine

#Funzione Rand.PSW
def generatePsw():

    ALL = "0123456789ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
    psw = "".join(random.choice(ALL) for i in range(9))
    return psw

#Funzione Rand.CN
def generateCardNumber():

```

```

NUMBERS = "0123456789"
number = "".join(random.choice(NUMBERS) for i in range(16))
return number
#Funzione Rand.Star
def checkStelleUtenti(stelle):
    if stelle == 1:
        return 1
    elif stelle == 2:
        return 2
    elif stelle == 3:
        return 3
    elif stelle == 4:
        return 4
    elif stelle == 5:
        return 5

print("Inizio esecuzione...")

print("L'ORDINE DI ESECUZIONE DEI FILE È 1.txt,2.txt,etc...")

print("Inizio Creazione 1.txt")

SYMBOLS = "ABCDEFGHIJKLMNPQRSTUVWXYZ"
NUMBERS = "0123456789"
ALL_SYMBOLS = "0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ"
f = open("1.txt", "w+")
print("----- Inizio Inserimento Personale\n")
random_id = ""
unique_Personale = []

values = []
for i in range(6000):
    data = genRandomDate()
    surname = fake.last_name()
    name = fake.first_name()
    email = str(generateEmail(name, surname))

    random_id = str(i)
    unique_Personale.append(random_id)

    query = "(" + random_id + ", '" + name + "','" + surname + "','" + str(data) + "','" + \
fake.phone_number() + "','" + email + "')"
    values.append(query)
f.write(
    "INSERT INTO Personale (ID, Nome, Cognome, DDN, NumeroDiTelefono, Email) VALUES" +
",\n".join(values) + ";" )
f.write("\n")
print("----- Fine Inserimento Personale\n")
f.close()

print("1.txt Done")

print("Inizio Creazione 2.txt")

f = open("2.txt","w+")

print("----- Inizio Inserimento Addetti Marketing\n")

```

```

unique_AddMark = []

ruoli = ["Responsabile", "Analista", "Coordinatore"]
values_marketing = []
for i in range(100):

    random_ruolo = random.choice(ruoli)
    random_id = unique_Personale[5900+i]

    query = "(" + random_id + ", " + random_ruolo + ")"
    unique_AddMark.append(random_id)
    values_marketing.append(query)
f.write(
    "INSERT INTO AddettiMarketing (ID_Adetto,Ruolo)
VALUES"+",\n".join(values_marketing)+";"
)
f.write("\n")

print("----- Fine Inserimento Addetti Marketing\n")
f.write("\n")
print("----- Inizio Inserimento Patente\n")

patenti = ["B","BE","B96"]
unique_Patente = []
values_patenti = []
for i in range(3000):
    data = genRandomLicenceDate()
    random_numpatente = "".join(random.choice(ALL_SYMBOLS) for i in range(9))
    categoria = "".join(random.choice(patenti) for i in range(1))
    unique_Patente.append(random_numpatente)

    query = "(" + random_numpatente + ", " + str(data) + ", " + categoria + ")"

    values_patenti.append(query)
f.write(
    "INSERT INTO Patente (NumeroPatente,DDS,Categoria) VALUES
"+",\n".join(values_patenti)+";"
)
f.write("\n")
print("----- Fine Inserimento Patente\n")
f.write("\n")
print("----- Inizio Inserimento Turni\n")

unique_Turno = []
ora_inizio = ['9','10','11','14']
ora_fine = ['17','20','21','22']
values_turni = []
for i in range(5):

    #random_turno = "".join(random.choice(NUMBERS) for i in range(1))
    random_turno = str(i)
    inizio = "".join(random.choice(ora_inizio))
    fine = "".join(random.choice(ora_fine))
    unique_Turno.append(random_turno)
    query = "(" + random_turno + ", " + inizio + ", " + fine + ")"

    values_turni.append(query)

```

```

f.write(
    "INSERT INTO Turni (ID_Turno,OrarioInizio,OrarioFine) VALUES
"+",\n".join(values_turni)+";"
)
f.write("\n")
print("----- Fine Inserimento Turni\n")
f.write("\n")
print("----- Inizio Inserimento Assicurazione\n")

unique_Assicurazione = []
values_assicurazione = []
tipo_assicurazione=["Kasko","Furto","Incendio","Base"]
for i in range(3000):
    random_id = str(i)
    data = genRandomInsuranceDate()
    tipo = random.choice(tipo_assicurazione)
    query = "("'+ str(random_id)+ ', "'+ str(data)+ '", "'+ str(tipo)+ ')"
    unique_Assicurazione.append(random_id)
    values_assicurazione.append(query)
f.write(
    "INSERT INTO Assicurazioni (ID_Assicurazione,DataScadenza,Tipo) VALUES
"+",\n".join(values_assicurazione)+";"
)
f.write("\n")
print("----- Fine Inserimento Assicurazione\n")
f.write("\n")
print("----- Inizio Inserimento Veicoli\n")
unique_Veicolo = []
values_veicolo = []
l_marca = ["Fiat","BMW","Audi","Range Rover","Seat"]
l_modello = ["Punto","Panda","Q8","RS7"]
for i in range(3000):
    random_targa = generateTarga()
    random_assicurazione = unique_Assicurazione[i]
    query = "("+ str(random_targa)+ ', "'+ str(random.choice(l_marca))+ ', "'+
str(random.choice(l_modello))+',
', "'+str(random.randint(1,12))+', "'+str(random_assicurazione)+')"
    unique_Veicolo.append(random_targa)
    values_veicolo.append(query)

f.write(
    "INSERT INTO Veicoli (Targa,Marca,Modello,PostiDisponibili,ID_Assicurazione) VALUES
"+",\n".join(values_veicolo)+";"
)
f.write("\n")
print("----- Fine Inserimento Veicoli\n")
f.write("\n")
print("----- Inizio Inserimento Autisti\n")

unique_Autisti = []
values_autisti = []
stipendio = ["1200","1100","900","800"]
for i in range(3000):
    random_id = unique_Personale[i]
    random_patente = unique_Patente[i]
    random_Turno = random.choice(unique_Turno[1:])
    random_targa = random.choice(unique_Veicolo)
    query = "("+ random_id+ ', "'+ random_patente+ ', "'+ random_Turno+

```

```

'''"+random_targa+"', '"+random.choice(stipendio)+"') "
unique_Autisti.append(random_id)
values_autisti.append(query)
f.write(
    "INSERT INTO Autisti (ID_Autista,NumerоПатенте,Турно,Targa,Stipendio) VALUES
"+",\n".join(values_autisti)+";"
)
f.write("\n")
print("----- Fine Inserimento Autisti\n")
f.write("\n")
print("----- Inizio Inserimento Manutentori\n")

unique_Manutentori = []
values_manutentori = []
qualifica = ["Gommista","Elettrauto","Meccanico","Carrozziere"]
for i in range(2900):
    random_id = unique_Personale[3000+i]

    query = "("'+ random_id+ "' , '"+ random.choice(qualifica)+ "')"
    unique_Manutentori.append(random_id)
    values_manutentori.append(query)
f.write(
    "INSERT INTO Manutentori (ID_Manutentore,Qualifica) VALUES
"+",\n".join(values_manutentori)+";"
)
f.write("\n")
print("----- Fine Inserimento Manutentori\n")
f.write("\n")
print("----- Inizio Inserimento ContattaPerGuasto\n")

unique_Contatto = []
values_contatto = []
motivi = ["Gomma Bucata","Spia dell motore accesa","Radiatore bucato","Batteria scarica","Problema con il FAP","Errore centralina","Specchietto rotto","Guarnizione della testata bruciata","Rottura degli ammortizzatori","Semiasse distrutto","Differenziale rotto","La macchina non parte","Cambio pasticche dei freni"]

for i in range(500):
    random_manutentore = random.choice(unique_Manutentori)
    random_autista = random.choice(unique_Autisti[0:200])

    query = "("'+ random_manutentore+ "' , '"+ random_autista+
"', '"+random.choice(motivi)+"')"
    unique_Contatto.append((random_manutentore,random_autista))
    values_contatto.append(query)
f.write(
    "INSERT INTO ContattaPerGuasto (ID_Manutentore,ID_Autista,Motivo) VALUES
"+",\n".join(values_contatto)+";"
)
f.write("\n")
print("----- Fine Inserimento ContattaPerGuasto\n")

print("2.txt Done")
f.close()

print("Inizio Creazione 3.txt")
f = open("3.txt","w+")

```

```

print("----- Inizio Inserimento Offerte\n")
unique_Offerta = []
offerta = ["Sconto 10%","Sconto 15%","Sconto 20%","Credito 5€","Credito 10€"]
values_offerta = []
for i in range(15):
    random_id = str(i)
    promo = "".join(str(random.randint(1,9)) for i in range(6))
    random_addetto = random.choice(unique_AddMark)
    query = "(""+ random_id+ '", "'+ promo+ '", "'+ random.choice(offerta)+"
", "'+random_addetto+')"'
    unique_Offerta.append(random_id)
    values_offerta.append(query)
f.write(
    "INSERT INTO Offerte (ID_Offerta,PromoCode,InfoOfferta,ID_Addetto) VALUES
"+",\n".join(values_offerta)+";"
)
f.write("\n")
print("----- Fine Inserimento Offerte\n")
f.write("\n")
print("----- Inizio Inserimento Utenti\n")

unique_Utenti = []
values_utenti = []

for i in range(10000):
    random_id = str(i)
    surname = fake.last_name()
    name = fake.first_name()
    email = str(generateEmail(name, surname))
    psw = generatePsw()
    id_off = random.choice(unique_Offerta)
    query = "(""+ random_id+ '", "'+ name+ '", "'+ surname+
", "'+email+ '", "'+psw+ '", "'+id_off+')"'
    unique_Utenti.append(random_id)
    values_utenti.append(query)
f.write(
    "INSERT INTO Utenti (ID_Utente,Nome,Cognome,Email,Password,ID_Offerta) VALUES
"+",\n".join(values_utenti)+";"
)
f.write("\n")
print("----- Fine Inserimento Utenti\n")
f.write("\n")
print("----- Inizio Inserimento Carte\n")

unique_Carta = []
values_carta = []

utente_carta = []
for i in range(10000):

    numero_Carta = str(random.randint(4,5))+"".join(str(random.randint(0,9)) for i in
range(3))+"".join(str(random.randint(0,9)) for i in range(4))+"
"+"".join(str(random.randint(0,9)) for i in range(4))+"
"+"".join(str(random.randint(0,9)) for i in range(4)))
    data_scadenza = genRandomCardDate()
    cvv = "".join(str(random.randint(0,9)) for i in range(3))
    utente = unique_Utenti[i]
    query = "(""+ numero_Carta+ '", "'+ str(data_scadenza)+ '", "'+ cvv+ '", "'+utente+')""

```

```

unique_Carta.append(numero_Carta)
values_carta.append(query)

utente_carta.append((utente,numero_Carta))
f.write(
    "INSERT INTO Carta (NumeroCarta,DataScadenza,CVV,ID_Utente) VALUES
"+",\n".join(values_carta)+";""
)
print("----- Fine Inserimento Carte\n")

print("3.txt Done")
f.close()
print("Inizio creazione 4.txt")
f = open("4.txt","w+")

print("----- Inizio Inserimento RichiestaPrenotazioni\n")

unique_RichPren = []
values_ricpren = []
raccolta = ["Anagnina","Termini","Centocelle","Eur","Tor Vergata","Colosseo"]
rilascio = ["Finocchio","Garbatella","Ostia","San Lorenzo","Primavalle","San Basilio"]
date = []
ora = ['9','10','11','14','15','16','20','21','22']
id_carta_utente = []
for i in range(20000):
    random_id = str(i)
    passeggeri = str(random.randint(1,12))

    tupla = random.choice(utente_carta[0:5000])
    #print(utente)
    utente = tupla[0]
    id_carta_utente.append(utente)
    autista = random.choice(unique_Autisti)
    data = genRandomRequestDate()
    orario = random.choice(ora)
    query = "("+" random_id+", "+ str(random.choice(raccolta))+ ", "+ +
str(random.choice(rilascio))+ ,
", "+str(data)+", "+str(orario)+", "+str(passeggeri)+", "+str(utente)+", "+str(autista)+")"
    unique_RichPren.append(random_id)
    values_ricpren.append(query)
    date.append(data)
f.write(
    "INSERT INTO RichiestePrenotazioni
(ID_Richiesta,PuntoDiRaccolta,PuntoDiRilascio,DataRichiesta,OrarioRichiesta,NumeroPasseggiatori,ID_Utente,ID_Autista) VALUES "+",\n".join(values_ricpren)+";""
)
f.write("\n")
print("----- Fine Inserimento RichiestaPrenotazioni\n")
f.write("\n")
print("----- Inizio Inserimento TratteCompleteate\n")

unique_TrattaC = []
values_trattac = []
costo = ["25","65","115","35","50"]
for i in range(15000):
    random_id = unique_RichPren[i]

```

```

costi = random.choice(costo)
id = int(id_carta_utente[i])

numcarta = utente_carta[id][1]
query = "(""+ random_id + ", "+ str(costi) + ", " + str(numcarta) + ")"
unique_TrattaC.append(random_id)
values_trattac.append(query)
f.write(
    "INSERT INTO TratteCompletate (ID_TrattaC,Costo,NumeroCarta) VALUES
"+",\n".join(values_trattac)+";"
)
f.write("\n")
print("----- Fine Inserimento TratteCompletate\n")
f.write("\n")
print("----- Inizio Inserimento Feedback\n")

unique_Feed = []
values_feed = []
feedback_utente = {
    1: ["Non lo prenderò mai più!","Esperienza orribile"],
    2: ["Non mi è piaciuto lo stile di guida","La prossima volta
preferirei un\'altro autista"],
    3: ["Nulla di particolare","Tutto nella norma"],
    4: ["Veicolo molto pulito e comodo.", "Esperienza normale"],
    5: ["Autista veramente cordiale","Ottima esperienza, lo dirò a
tutti"],
}
feedback_autisti = {
    1: ["Utente scortese!","L\'utente offende"],
    2: ["Utente ritardatario","Non rispetta l\'autista"],
    3: ["Nulla di particolare","Utente ok"],
    4: ["Utente rispettoso.", "Utente gentile"],
    5: ["Utente veramente genuino","Molto bravo e cortese"],
}

for i in range(15000):
    random_id = str(i)

    stelle_random_ut = random.choice(list(feedback_utente.keys()))

    commento_ut = str(random.choice(feedback_utente[stelle_random_ut]))

    stelle_random_aut = checkStelleUtenti(stelle_random_ut)

    commento_aut = str(random.choice(feedback_autisti[stelle_random_aut]))
    random_trattac = random.choice(unique_TrattaC)
    query = "("+ random_id + ", "+ str(stelle_random_ut) + ", " + str(commento_ut) +
    ", " + str(stelle_random_aut) + ", " + str(commento_aut) + ", " + str(random_trattac) + ")"

    unique_Feed.append(random_id)
    values_feed.append(query)
    unique_TrattaC.remove(random_trattac)

f.write(
    "INSERT INTO Feedback
(ID_Feedback,StelleUtente,CommentoUtente,StelleAutista,CommentoAutista,ID_TrattaCompleta)
VALUES "+",\n".join(values_feed)+";"
)

```

```

)
f.write("\n")
print("----- Fine Inserimento Feedback\n")
f.write("\n")
print("----- Inizio Inserimento TratteRifiutate\n")

unique_TrattaR = []
values_trattar = []
motivi = ["Problema generale","Indisponibilità al servizio","Tropo lontano"]
for i in range(5000):
    random_id = unique_RichPren[15000+i]
    motivo = random.choice(motivi)
    query = "("+ random_id+", '"+ str(motivo)+ ")"
    unique_TrattaR.append(random_id)
    values_trattar.append(query)
f.write(
    "INSERT INTO TratteRifiutate (ID_TrattaR,Motivazione) VALUES
"+",\n".join(values_trattar)+";"
)
f.write("\n")
print("----- Fine Inserimento TratteRifiutate\n")
print("4.txt Done")
f.close()

```

Query

- Visualizza tutte le tratte complete, con annesso costo e carta usata per il pagamento, fatte da un determinato utente

```

SELECT ID_Richiesta, PuntoDiRaccolta as Partenza, PuntoDiRilascio as Arrivo, Costo,
c.NumeroCarta as Carta from `RichiestePrenotazioni` rp
JOIN `TratteComplate` tc on rp.`ID_Richiesta` = tc.`ID_TrattaC`
JOIN `Carta` c on tc.`NumeroCarta` = c.`NumeroCarta`
JOIN `Utenti` u on c.`ID_Utente` = u.`ID_Utente`
WHERE u.Nome = 'Carla' AND u.Cognome = 'Raimondi';

```

ID_Richiesta	Partenza	Arrivo	Costo	Carta
4417	Centocelle	Primavalle	65	4048 6114 7559 1284
4428	Centocelle	Ostia	25	4048 6114 7559 1284
6137	Eur	San Lorenzo	65	4048 6114 7559 1284
6818	Colosseo	Finocchio	25	4048 6114 7559 1284
7277	Eur	Finocchio	50	4048 6114 7559 1284
8911	Eur	San Lorenzo	35	4048 6114 7559 1284
9254	Tor Vergata	Primavalle	35	4048 6114 7559 1284
10040	Termini	Finocchio	50	4048 6114 7559 1284
11178	Centocelle	Finocchio	35	4048 6114 7559 1284
12383	Tor Vergata	Primavalle	115	4048 6114 7559 1284
14086	Anagnina	Primavalle	35	4048 6114 7559 1284
14490	Anagnina	San Basilio	50	4048 6114 7559 1284

- Visualizza tutti i veicoli la cui assicurazione scadrà entro febbraio 2024

```

SELECT Targa, Modello, Marca, a.DataScadenza AS DataScadenza FROM Veicoli v
JOIN Assicurazioni a
ON v.ID_Assicurazione = a.ID_Assicurazione
WHERE YEAR(a.DataScadenza) = "2024" AND MONTH(a.DataScadenza) = "02";

```

Targa	Modello	Marca	DataScadenza
NG689AU	Panda	Seat	2024-02-25
BS954LQ	Panda	BMW	2024-02-19
YU992QG	Q8	Range Rover	2024-02-12
PM899AD	Punto	Seat	2024-02-29
PB970EP	Punto	Audi	2024-02-21
DL432AK	Q8	BMW	2024-02-27
XD647LI	Punto	Seat	2024-02-26
EE776NQ	Q8	Range Rover	2024-02-25
TA763RF	Punto	Audi	2024-02-03
EU994ZB	Q8	Fiat	2024-02-18
UH714NO	RS7	Range Rover	2024-02-01
KY684YD	Q8	Audi	2024-02-28
BW963CZ	RS7	Range Rover	2024-02-15
BW856TZ	RS7	Fiat	2024-02-24
M0215HT	RS7	Range Rover	2024-02-03
ZA664HT	Punto	Audi	2024-02-20
TS019EE	Punto	Fiat	2024-02-10
OR463PY	Panda	Fiat	2024-02-26
KT483KT	Q8	Seat	2024-02-21
FN630KS	RS7	BMW	2024-02-02
EF317NT	Punto	Seat	2024-02-20
M0528HS	Panda	Seat	2024-02-17
XE732DE	Panda	Fiat	2024-02-23
QD075QI	RS7	Seat	2024-02-12
HD649FG	RS7	BMW	2024-02-11
ES9000E	RS7	BMW	2024-02-14
DL142KS	Panda	Fiat	2024-02-02
CB962GE	Q8	Seat	2024-02-26
S01240S	RS7	Range Rover	2024-02-26
ZJ38560	Q8	BMW	2024-02-04
DX951FO	Panda	Range Rover	2024-02-15
FP861GO	Punto	Audi	2024-02-11
HD819NN	RS7	Range Rover	2024-02-18
IC211NA	RS7	Fiat	2024-02-11

- Visualizza il/i turno/i di un dato autista

```
SELECT p.Nome, p.Cognome, t.OrarioInizio, t.OrarioFine FROM Autisti a JOIN Personale p
ON a.ID_Autista = p.ID
JOIN Turni t ON a.Turno = t.ID_Turno
WHERE p.Nome = "Patrizio" AND p.Cognome = "Mattarella";
```

Nome	Cognome	OrarioInizio	OrarioFine
Patrizio	Mattarella	9	22

- Visualizza tutti gli autisti che hanno lo stesso turno

```
SELECT p.Nome, p.Cognome, a.Turno, a.ID_Autista FROM Autisti a JOIN Personale p
ON a.ID_Autista = p.ID
JOIN Turni t ON a.Turno = t.ID_Turno
WHERE a.Turno = 2;
```

Nome	Cognome	Turno	ID_Autista
Maria	Andreotti	2	1
Lorenzo	Toscani	2	3
Renata	Gonzaga	2	6
Arnulfo	Borromini	2	9
Rosalia	Giulietti	2	10
Durante	Ughi	2	11
Raffaellino	Strangio	2	17
Piersanti	Chiaramonte	2	21
Tiziano	Micheletti	2	22
Caterina	Minati	2	23
Temistocle	Palumbo	2	28
Alberto	Ceri	2	32
Nicoletta	Morpugo	2	41
Ottavio	Duodo	2	42
Arnulfo	Bocca	2	43
Vittoria	Malpighi	2	45
Lamberto	Scarpone	2	47
Rodolfo	Berlusconi	2	50
Durante	Pagliaro	2	56
Sophia	Roccabonella	2	57
Ornella	Bazzi	2	58
Adriana	Zoppetti	2	62
Pomponio	Comolli	2	66
Gaspare	Monaco	2	80
Coluccio	Ceravolo	2	82
Camilla	Pistoleto	2	85
Salvi	Gritti	2	95
Ubaldo	Ravaglioli	2	96
Valeria	Tamburello	2	100
Piermaria	Salieri	2	112
Alfio	Sbarbaro	2	115
Roland	Lippomano	2	116
Zaira	Binaghi	2	117
Emma	Passalacqua	2	127

- Visualizza la somma dei pagamenti effettuati dagli utenti in una data settimana

```
SELECT SUM(tc.Costo) AS Totale FROM TratteCompletate tc
JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
WHERE MONTH (rp.DataRichiesta) = "06" AND DAY (rp.DataRichiesta) BETWEEN 1 AND 7
```

Totale
17060

- Visualizza tutte le richieste di manutenzione relative ad uno specifico veicolo

```
SELECT cpg.Motivo, v.* FROM ContattaPerGuasto cpg
JOIN Autisti a ON cpg.ID_Autista = a.ID_Autista
JOIN Veicoli v ON a.Targa = v.Targa
WHERE v.Targa = "QD795XT";
```

Motivo	Targa	Marca	Modello	PostiDisponibili	ID_Assicurazione
Gomma Bucata	QD795XT	Seat	RS7	7	723
Radiatore bucato	QD795XT	Seat	RS7	7	723
Gomma Bucata	QD795XT	Seat	RS7	7	723
Differenziale rotto	QD795XT	Seat	RS7	7	723
Differenziale rotto	QD795XT	Seat	RS7	7	723
Spia dell motore accesa	QD795XT	Seat	RS7	7	723
Guarnizione della testata bruciata	QD795XT	Seat	RS7	7	723
Problema con il FAP	QD795XT	Seat	RS7	7	723
Batteria scarica	QD795XT	Seat	RS7	7	723
Guarnizione della testata bruciata	QD795XT	Seat	RS7	7	723

- Visualizza le 10 tratte più gettonate

```
SELECT PuntoDiRaccolta, PuntoDiRilascio, COUNT(*) AS NumeroRichieste
FROM RichiestePrenotazioni rp
GROUP BY PuntoDiRaccolta, PuntoDiRilascio
ORDER BY NumeroRichieste DESC
LIMIT 10;
```

PuntoDiRaccolta	PuntoDiRilascio	NumeroRichieste
Anagnina	Ostia	589
Termini	San Basilio	588
Termini	Finocchio	582
Eur	Ostia	581
Colosseo	Finocchio	580
Tor Vergata	San Lorenzo	579
Colosseo	Ostia	576
Eur	San Lorenzo	574
Eur	Garbatella	573
Anagnina	Primavalle	572

- Visualizza gli utenti che hanno effettuato almeno 10 richieste

```
SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroRichieste
FROM RichiestePrenotazioni rp JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
GROUP BY u.ID_Utente, u.Nome, u.Cognome
HAVING NumeroRichieste >= 10
ORDER BY NumeroRichieste DESC;
```

ID_Utente	Nome	Cognome	NumeroRichieste
834	Carla	Raimondi	13
3579	Rossana	Caccianemico	13
2914	Amalia	Bembo	12
1308	Cirillo	Accardo	12
3393	Irma	Bernardi	12
4322	Giada	Farina	12
1981	Danilo	Gentilini	12
1874	Giulia	Molesini	11
3805	Matteo	Chigi	11
3368	Riccardo	Chigi	11
2779	Arnaldo	Crespi	11
2756	Maurilio	Roccabonella	11
2362	Maria	Alteria	11
3885	Marco	Gottardi	11
2259	Gloria	Scialpi	11
4148	Stefania	Giannini	11
4345	Gianfrancesco	Doria	11
3397	Antonina	Toscani	11
1523	Mauro	Mercantini	11
1517	Pellegrino	Correr	11
1245	Armando	Montanari	11
392	Antonia	Chindamo	11
348	Eugenio	Tomasetti	11
3628	Natalia	Ortolani	11
4059	Rocco	Liguori	10
3795	Valeria	Celentano	10
3793	Giorgio	Belletini	10
3710	Fabia	Contrafatto	10
3842	Milena	Maderna	10
3850	Lucia	Castioni	10
3855	Alina	Ciampi	10
3183	Lando	Giusti	10
4352	Sandro	Pertini	10
4569	Laureano	Branciforte	10

- Di un dato range di utenti (ID compreso tra 2000 e 6000 e l'iniziale del nome "F"), visualizza le offerte associate agli utenti, con le relative descrizioni

```

SELECT u.ID_Utente, u.Nome, u.Cognome, o.ID_Offerta, o.InfoOfferta
FROM Utenti u JOIN Offerte o ON u.ID_Offerta = o.ID_Offerta
WHERE u.ID_Utente
IN
(
    SELECT ID_Utente
    FROM Utenti u2
    WHERE ID_Utente BETWEEN 2000 AND 6000 AND Nome LIKE "F%"
)

```

ID_Utente	Nome	Cognome	ID_Offerta	InfoOfferta
2022	Fredo	Missoni	13	Credito 5€
2038	Francesca	Abbagnale	3	Credito 5€
2045	Franco	Catenazzi	9	Credito 10€
2047	Fabia	Rizzo	8	Credito 10€
2058	Federigo	Filippelli	5	Credito 5€
2064	Federico	Nadi	7	Credito 10€
2097	Francesca	Lucciano	7	Credito 10€
2103	Federica	Infantino	2	Credito 10€
2119	Fiorenzo	Tropea	3	Credito 5€
2136	Fortunata	Sordi	11	Sconto 10%
2153	Flora	Bonaventura	5	Credito 5€
2156	Fiorino	Gregorio	12	Sconto 20%
2178	Federica	Ughi	13	Credito 5€
2217	Fulvio	Giunti	11	Sconto 10%
2238	Fiorino	Corcos	7	Credito 10€
2244	Fredo	Canevascini	13	Credito 5€
2255	Federigo	Boaga	7	Credito 10€
2272	Fausto	Zetticci	1	Sconto 15%
2285	Fernanda	Sokolov	4	Credito 10€
2293	Fortunata	Franscini	9	Credito 10€
2317	Fabrizio	Ginesio	1	Sconto 15%
2326	Fernanda	Gagliano	10	Sconto 15%
2358	Fiamma	Morucci	0	Credito 5€
2385	Flavio	Sollima	10	Sconto 15%
2431	Flavia	Geraci	14	Sconto 20%
2447	Filippa	Scarpa	5	Credito 5€
2485	Fiorino	Montesano	1	Sconto 15%
2487	Francesca	Ammaniti	6	Sconto 20%
2488	Federigo	Celentano	2	Credito 10€
2514	Filippo	Zetticci	12	Sconto 20%
2524	Filippo	Luxardo	3	Credito 5€
2538	Flora	Gabrieli	1	Sconto 15%
2554	Francesca	Veltroni	2	Credito 10€
2555	Fortunata	Comisso	9	Credito 10€

- Visualizza il motivo di rifiuto delle richieste di prenotazione che occorre più spesso

```
SELECT tr.Motivazione, COUNT(*) AS NumeroOcorrenze
FROM TratteRifiutate tr GROUP BY tr.Motivazione
ORDER BY NumeroOcorrenze DESC
LIMIT 1;
```

Motivazione	NumeroOcorrenze
Troppo lontano	1715

- Di tutti gli autisti che hanno un ID compreso tra 50 e 400, mostra il turno assegnato e i dati del veicolo che utilizzano

```
SELECT a.ID_Autista, p.Nome, p.Cognome, t.*, v.*
FROM Autisti a JOIN Personale p ON a.ID_Autista = p.ID
JOIN Veicoli v ON a.Targa = v.Targa
JOIN Turni t ON a.Turno = t.ID_Turno
WHERE ID_Autista BETWEEN 50 AND 400;
```

ID_Autista	Nome	Cognome	ID_Turno	OrarioInizio	OrarioFine	Targa	Marca	Modello	PostiDisponibili	ID_Ascrizione
50	Rodolfo	Berlusconi	2	10	21	GT302MM	Range Rover	Panda	4	1013
51	Guglielmo	Parpinel	4	14	17	OI183VO	Range Rover	Q8	12	126
52	Piero	Morellato	1	11	22	QB5060G	Seat	Punto	4	1456
53	Marisa	Alfonsi	3	9	22	DI373WK	BMW	Panda	2	494
54	Donatello	Paganini	1	11	22	PE854RC	Fiat	Q8	2	1480
55	Raffaella	Meucci	4	14	17	RJ854GV	Audi	Panda	3	757
56	Durante	Pagliaro	2	10	21	ZQ737ZU	BMW	Punto	2	1116
57	Sophia	Roccabonella	2	10	21	FY816BA	BMW	Panda	3	1147
58	Ornella	Bazzi	2	10	21	RS564GH	Seat	Q8	5	493
59	Carla	Marinetti	3	9	22	QE385SM	Fiat	Panda	12	545
60	Beppe	Manzoni	3	9	22	00867K5	Audi	Q8	9	369
61	Etta	Tutino	1	11	22	SR844CE	Seat	Panda	2	446
62	Adriana	Zoppetti	2	10	21	QE292IK	BMW	Punto	5	177
63	Fredo	Schiavone	3	9	22	UV135EX	Range Rover	RS7	12	1333
64	Arnulfo	Bernardini	1	11	22	PQ556AX	Fiat	Q8	12	2160
65	Elisa	Casadei	1	11	22	ZY364JJ	Seat	Panda	6	1818
66	Pomponio	Comolli	2	10	21	MA919XB	Audi	Punto	1	1983
67	Ignazio	Manolessos	4	14	17	EZ384VX	Fiat	RS7	6	789
68	Ludovico	Guarana	1	11	22	P6098SL	BMW	Panda	4	2143
69	Massimiliano	Medici	1	11	22	GC170HO	Audi	Panda	5	1248
70	Santino	Oscuro	1	11	22	SV546TA	BMW	Panda	9	1445
71	Federica	Ciampi	1	11	22	JZ676ZS	BMW	Punto	10	2642
72	Benito	Boccioni	3	9	22	V0014LC	Audi	Punto	3	2020
73	Jacopo	Argentero	1	11	22	VT628VM	Seat	RS7	3	2846
74	Gabiella	Tebaldi	1	11	22	UA329GO	Range Rover	Punto	5	462
75	Roman	Vivaldi	4	14	17	ZP070UV	Audi	Punto	12	838
76	Piergiuseppe	Nordio	1	11	22	PR077WD	Audi	RS7	3	1284
77	Goffredo	Ungaretti	1	11	22	X44220U	Range Rover	Panda	12	1118
78	Romina	Abate	4	14	17	XK321LB	Audi	Punto	6	1051
79	Luigina	Fabrizi	4	14	17	BA973YJ	Audi	Punto	4	1949
80	Gaspare	Monaco	2	10	21	RR659HL	Range Rover	Panda	1	2724
81	Annetta	Federici	1	11	22	UB838RI	Audi	RS7	12	2793
82	Coluccio	Ceravolo	2	10	21	GE470NQ	Range Rover	Q8	7	580
83	Roberto	Marzorati	4	14	17	UP733YB	Seat	Q8	9	2167

- Visualizza tutte le tratte complete che non hanno un feedback

```

SELECT tc.* FROM TratteCompletate tc
WHERE tc.ID_TrattaC NOT IN
(
    SELECT ID_TrattaCompleta FROM Feedback f
);

```

ID_TrattaC	Costo	NumeroCarta
102	65	5277 7060 4814 6995
120	25	4676 4172 8752 0672
606	50	4604 9678 4062 6436
775	35	5785 6166 6221 0249
919	50	4083 5291 4979 9396
973	35	4850 5660 7069 4317
976	35	5211 1773 5037 4426
1021	50	5183 0576 5680 1109
1027	35	5058 4856 8902 5330
1318	35	5855 4569 3788 1195
2046	35	5896 0533 5832 1033
2150	115	5428 9082 9242 5704
2365	25	5098 8979 6526 5431
2490	50	4325 9639 8484 6346
2901	35	4132 0276 1178 5280
3256	50	4956 9882 9512 0429
4410	65	5727 0506 2267 8882
4885	115	4432 5673 5179 1973
5044	50	5070 7215 7477 2584
5204	65	5036 3168 5974 6237
5514	25	4828 9595 8824 4837
5527	35	5439 6303 7666 5186
5558	35	4146 1375 6774 7902
6059	35	5797 6634 4461 6684
6357	115	4114 0620 7225 4769
6936	35	4449 7299 5271 8500
7336	65	5606 6073 0100 7549
7770	65	4132 8474 2400 3269
8560	35	4768 1546 8125 0993
8576	35	5546 1568 0885 4147
9160	115	5035 4598 3328 3860
9383	35	5207 8818 4301 5440
9464	35	4124 0528 1860 6934
9663	35	4055 5702 9750 1915

- Visualizza tutte le richieste di prenotazione effettuate da un determinato utente

```
SELECT rp.* FROM RichiestePrenotazioni rp
JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
WHERE u.Nome = 'Carla' AND u.Cognome = 'Raimondi'
```

ID_Richiesta	PuntoDiRaccolta	PuntoDiRilascio	DataRichiesta	OrarioRichiesta	NumeroPasseggeri	ID_Utente	ID_Autista
4417	Centocelle	Primavalle	2022-12-10	9	11	834	2254
4428	Centocelle	Ostia	2022-05-09	10	10	834	652
6137	Eur	San Lorenzo	2023-02-22	22	1	834	2184
6818	Colosseo	Finocchio	2023-02-05	20	3	834	1280
7277	Eur	Finocchio	2023-03-21	11	6	834	865
8911	Eur	San Lorenzo	2023-10-07	10	9	834	1595
9254	Tor Vergata	Primavalle	2022-07-24	16	4	834	298
10040	Termini	Finocchio	2022-09-22	14	1	834	592
11178	Centocelle	Finocchio	2022-10-23	21	6	834	2174
12383	Tor Vergata	Primavalle	2022-06-15	10	10	834	578
14086	Anagnina	Primavalle	2022-12-29	14	7	834	2134
14490	Anagnina	San Basilio	2023-04-09	20	9	834	2330
18189	Tor Vergata	Ostia	2022-11-20	14	10	834	2134

- Visualizza la media delle stelle ottenute da un singolo autista

```
SELECT p.Nome, p.Cognome, AVG(f.StelleUtente) AS MediaStelle
FROM Feedback f
JOIN TratteCompletate tc ON f.ID_TrattaCompletata = tc.ID_TrattaC
JOIN RichiestePrenotazioni rp ON rp.ID_Richiesta = tc.ID_TrattaC
JOIN Autisti a ON rp.ID_Autista = a.ID_Autista
```

```
JOIN Personale p ON a.ID_Autista = p.ID
WHERE rp.ID_Autista = '500'
GROUP BY p.Nome, p.Cognome
ORDER BY MediaStelle DESC
```

Nome	Cognome	MediaStelle
Gianpaolo	Cuomo	2.6667

- Visualizza il numero totale delle assicurazioni Kasko

```
SELECT COUNT(a.Tipo) AS TotaleKasko
FROM Assicurazioni a
WHERE a.Tipo = 'Kasko';
```

TotaleKasko
767

- Visualizza tutti gli autisti che hanno una certa categoria di patente

```
SELECT p.Nome, p.Cognome, pt.Categoria
FROM Personale p JOIN Autisti a ON p.ID = a.ID_Autista
JOIN Patente pt ON a.NumeroPatente = pt.NumeroPatente
WHERE pt.Categoria = "B96"
```

Nome	Cognome	Categoria
Fredo	Mazzeo	B96
Valeria	Bixio	B96
Antonella	Boito	B96
Annibale	Trobbiani	B96
Bruno	Abbagnale	B96
Lucio	Nadi	B96
Paulina	Foscari	B96
Gelsomina	Zaguri	B96
Calcedonio	Cibin	B96
Damiano	Serlupi	B96
Rolando	Lippomano	B96
Ludovico	Lattuada	B96
Antonello	Surian	B96
Gaspare	Garrone	B96
Irma	Scarpa	B96
Lidia	Bellucci	B96
Gianluigi	Beffa	B96
Giuseppina	Mercalli	B96
Ugo	Boitani	B96
Pietro	Bembo	B96
Simone	Borghese	B96
Dino	Giusti	B96
Cecilia	Zamorani	B96
Eliana	Trevisani	B96
Aldo	Buscetta	B96
Franco	Catenazzi	B96
Marina	Abbagnale	B96
Germana	Garozzo	B96
Vittoria	Parisi	B96
Fedele	Disdero	B96
Azeglio	Filangieri	B96
Panfilo	Righi	B96
Stella	Altera	B96
Ronaldo	Pulci	B96

- Visualizza il numero di feedback con almeno 3 stelle lasciati da ogni utente

```

SELECT u.Nome,u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
FROM Feedback f JOIN TratteCompletate tc ON f.ID_TrattaCompleta = tc.ID_TrattaC
JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
WHERE f.StelleUtente >= 3
GROUP BY u.Nome,u.Cognome
ORDER BY NumeroFeedback3Stelle DESC
    
```

Nome	Cognome	NumeroFeedback3Stelle
Greca	Giammusso	9
Maurilio	Roccabonella	9
Carla	Raimondi	8
Irma	Bernardi	8
Maria	Altera	8
Rodolfo	Ottino	8
Laureano	Branciforte	8
Cirillo	Accardo	8
Bianca	Bramante	8
Riccardo	Chigi	7
Gioele	Trevisani	7
Gabriele	Montesano	7
Giada	Farina	7
Tonino	Bertolucci	7
Luchino	Faloppio	7
Gustavo	Mantegazza	7
Gianfrancesco	Doria	7
Ermenegildo	Sforza	7
Cipriano	Farnese	7
Giulia	Molesini	7
Cristina	Palmisano	7
Clelia	Moschino	7
Uberto	Leoncavallo	7
Leopoldo	Guariento	7
Amalia	Bembo	7
Baccio	Corcos	7
Sophia	Eco	6
Corrado	Lamborghini	6
Allegra	Maglio	6
Raffaele	Righi	6
Antonina	Toscani	6
Elladio	Morosini	6
Melina	Leoncavallo	6
Irma	Ferrabosco	6

- Visualizza l'ultima richiesta di prenotazione di un certo utente, aggiungendo (**solo in output**) un campo che dice se la Richiesta fa parte di una tratta completata o no

```

SELECT rp.*,
IF(rp.ID_Richiesta IN (SELECT ID_TrattaC FROM TratteCompletate tc),'SI','NO' ) AS Completata
FROM RichiestePrenotazioni rp
JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
WHERE rp.ID_Utente = '3430' AND rp.DataRichiesta
IN
(
    SELECT MAX(DataRichiesta)
    FROM RichiestePrenotazioni
    WHERE ID_Utente = '3430'
)

```

ID_Richiesta	PuntoDiRaccolta	PuntoDiRilascio	DataRichiesta	OrarioRichiesta	NumeroPasseggeri	ID_Utente	ID_Autista	Completata
16533	Tor Vergata	Garbatella	2023-12-02	22	5	3430	929	NO

- Visualizza le tratte completate con un certo tipo di veicolo

```

SELECT TC.* , v.Marca , a.ID_Autista
FROM TratteCompletate TC
JOIN RichiestePrenotazioni rp ON TC.ID_TrattaC = rp.ID_Richiesta
JOIN Autisti a ON rp.ID_Autista = a.ID_Autista
JOIN Veicoli v ON a.Targa = v.Targa
WHERE v.Marca = 'Seat';

```

ID_TrattaC	Costo	NumeroCarta	Marca	ID_Autista
2617	25	5121 0795 7344 3374	Seat	1104
5705	65	4517 2296 0615 9529	Seat	1104
7141	50	5372 7760 5457 8121	Seat	1104
10536	50	4426 7021 9983 6014	Seat	1104
13078	50	5742 6042 4515 4186	Seat	1104
608	35	5725 9622 3140 9595	Seat	1170
1725	50	4119 8624 8245 6401	Seat	1170
2317	65	4699 2312 7424 3046	Seat	1170
4082	65	4307 8881 1403 0495	Seat	1170
8998	115	4487 1703 0235 8742	Seat	1170
9066	50	4390 0255 2193 5050	Seat	1170
9092	115	5122 9626 0024 5477	Seat	1170
3517	25	5382 4988 1181 0190	Seat	2498
4069	115	4037 5646 9208 7369	Seat	2498
5557	115	5569 5415 2146 4953	Seat	2498
6736	35	4294 1066 1660 4927	Seat	2498
7019	25	4979 5792 7271 4063	Seat	2498
7817	50	5418 2337 7741 9423	Seat	2498
8360	50	4789 2561 5393 0060	Seat	2498
8896	115	5490 5443 5211 1028	Seat	2498
9656	50	5987 2243 0518 5180	Seat	2498
11093	65	4794 0471 5759 0083	Seat	2498
1145	35	4483 7108 7156 6259	Seat	282
3302	115	4439 7207 4063 0070	Seat	282
13976	50	4857 7492 3270 5196	Seat	282
8812	65	5418 6831 3116 3952	Seat	2599
2640	25	4350 0965 2282 6590	Seat	2619
2950	50	4246 2771 8446 8763	Seat	2619
3216	25	4556 9613 2375 9571	Seat	2619
10298	65	4987 3964 1971 3375	Seat	2619
12693	50	4673 0925 9918 0166	Seat	2619
12738	115	4689 0793 7655 4080	Seat	2619
13235	50	5874 1065 3992 7996	Seat	2619
14176	115	5542 9398 3243 3796	Seat	2619

- Trova tutti gli autisti che non hanno mai effettuato una richiesta di manutenzione

```

SELECT A.*
FROM Autisti A
WHERE A.ID_Autista NOT IN
(
    SELECT cpg.ID_Autista FROM ContattaPerGuasto cpg
);

```

ID_Autista	NumeroPatente	Turno	Targa	Stipendio
15	0VJ5IZUF4	1	VS897AB	900
37	PMDT08VHO	3	0Y226TX	800
43	WQZKRMGVK	2	PB970EP	900
54	AROYGZA63	1	PE854RC	800
69	57ZUM345R	1	GC170HO	800
82	KNOK021V5	2	GE470NJ	800
92	PX4EPQMV1	4	EW0880A	800
99	EHX2KQAM1	3	IW083JC	800
100	VM6PB8M2X	2	DM130PU	1100
106	JRHIV0ANU	4	VE948JA	1100
107	DDH61N974	4	BB177IY	1200
128	LXS96NTXX	3	AF857QZ	900
136	7501XPNQ	2	AA9771S	800
154	I456ZFTM7	2	AC119T0	1100
156	LOBSY2QY	1	F746UY	1100
159	USSVUB5TH	4	NH819RF	900
164	SL3MKJ84Y	1	OB408QM	1200
187	8J5YXG2TN	3	JZ365GS	900
200	PJ1DN051P	3	UL491VU	1200
201	UTV68JCN9	3	OM225ZC	900
202	DNMJ96D70	3	VQ7290R	900
203	CTNXJ9P2A	3	XD219C0	900
204	FXIYNN2NJG	1	UM604TO	1100
205	7VVBFT3X0	4	QD118BL	1200
206	K13010BRE	4	SG247PN	1100
207	CT1R3IDHP	1	DN245KH	800
208	CE0ZRH00G	1	MN503HW	800
209	N2JZL9USM	2	PF383DP	1100
210	4BXD2Y3AZ	1	GD067CJ	900
211	3GUJENCJR	2	EF317NT	1100
212	AZZBN0208	2	PN586KV	1200
213	E66H7Q2BE	3	CZ089CX	800
214	G56DJSHE3	2	BI5780F	1200
215	FNMZ82XBI	2	HU706QD	1200

- Visualizza il totale dei pagamenti relativi ad un determinato giorno

```

SELECT SUM(tc.Costo) AS TotalePagamenti FROM TratteCompletate tc
JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta

```

```
WHERE rp.DataRichiesta = "2023-06-05"
```

TotalePagamenti
1270

- Visualizza gli autisti con lo stipendio più alto

```
SELECT *
FROM Autisti
WHERE Stipendio =
(
    SELECT MAX(Stipendio)
    FROM Autisti
);
```

ID_Autista	NumeroPatente	Turno	Targa	Stipendio
0	1VZAP65HF	3	QF3650B	1200
2	SIHTMVO46	1	VS694B0	1200
3	7XDZB028P	2	AD555MK	1200
24	8916KAW75	4	XY854YD	1200
29	QL3OL7NTT	4	PK982YB	1200
32	IWNR4XISH	2	SI220AN	1200
44	4F4ANZ4WW	3	FI492SI	1200
57	3WPQ2TB2M	2	FY816BA	1200
58	RTZOL6LVY	2	RS564GH	1200
60	HXSSHENXT	3	OQ867KS	1200
64	OV823MTRY	1	PQ556AX	1200
65	S5YKYA1NV	1	ZY364JJ	1200
67	LDADUYUZA	4	EZ384VX	1200
68	OH2TMRART	1	PG098SL	1200
72	7GAZ22CXI	3	V0014LC	1200
73	K78DB6HC5	1	VT628VM	1200
78	T8DUU91PR	4	XK321LB	1200
79	DAC7BAPRC	4	BA973YJ	1200
84	QQ188301J	4	VG466BR	1200
85	6TA7753N5	2	UY467AH	1200
91	6VH1TVDEA	4	NH0976P	1200
94	GNDYW2DE6	1	GP136VV	1200
96	OKF73C61Q	2	IA5456R	1200
97	V1LR0GBHZ	3	LJ790YI	1200
181	NZ035F583	3	KL764GU	1200
107	DDH61N974	4	BB177IY	1200
108	IUF14ICKU	3	YX365VM	1200
118	XLWMU9L7F	1	BC210WG	1200
121	9VJXAN1XL	3	BB177IY	1200
124	XR00IKZ4J	4	RX184KI	1200
126	XSMCRMPRL	4	LZ057QA	1200
127	SQ8GEN4SS	2	VF527WL	1200
130	QTNH298AG	1	IT166WC	1200
135	07HWG9TUG	1	HA254NX	1200

- Trova gli autisti che hanno completato il minor numero di corse in un determinato giorno

```
SELECT a.ID_Autista,p.Nome,p.Cognome, COUNT(*) AS NumeroCorseEffettuate
FROM TratteCompletate tc
JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
JOIN Autisti a ON rp.ID_Autista = a.ID_Autista
JOIN Personale p ON a.ID_Autista = p.ID
WHERE rp.DataRichiesta = "2023-06-05"
GROUP BY a.ID_Autista, p.Nome, p.Cognome
ORDER BY NumeroCorseEffettuate
```

ID_Autista	Nome	Cognome	NumeroCorseEffettuate
936	Lauretta	Salvemini	1
177	Bianca	Bramante	1
2501	Pierangelo	Marazzi	1
2426	Monica	Orsini	1
1359	Piero	Trillini	1
262	Vittoria	Norbiato	1
1678	Camilla	Seddio	1
127	Emma	Passalacqua	1
2203	Pier	Falcone	1
361	Azeglio	Togliatti	1
2676	Lodovico	Carli	1
586	Danilo	Tebaldi	1
1667	Antonia	Solimena	1
2153	Arnulfo	Argan	1
175	Agnolo	Schiaparelli	1
1035	Corrado	Proietti	1
539	Ludovico	Antonacci	1
2405	Rosalia	Curci	1
126	Gastone	Onio	1
2548	Pietro	Pavone	1

- Visualizza tutti i dati di un determinato utente, comprese le carte a lui associate

```
SELECT u.* , c.NumeroCarta, c.DataScadenza, c.CVV
FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
WHERE Nome = "Geronimo" AND Cognome = "Lucarelli"
```

ID_Utente	Nome	Cognome	Email	Password	ID_Offerta	NumeroCarta	DataScadenza	CVV
0	Geronimo	Lucarelli	Geronimo.Lucarelli@ligorio.it	UllKLu1aR	12	5516 6245 1261 0132	2030-10-18	479

- Visualizza tutti gli utenti che hanno almeno 2 carte associate

```
SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroCarteAssociate
FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
GROUP BY u.ID_Utente, u.Nome, u.Cognome
HAVING NumeroCarteAssociate > 2
ORDER BY NumeroCarteAssociate DESC
```

ID_Utente	Nome	Cognome	NumeroCarteAssociate
74	Ottavio	Garzoni	7
38	Angelina	Avogadro	7
56	Orlando	Tartaglia	6
32	Paloma	Scarpetta	6
97	Ermanno	Fagiani	6
12	Mario	Colletti	6
26	Lidia	Brenna	6
69	Alina	Campise	6
25	Romana	Trapanese	5
47	Leopoldo	Binaghi	5
46	Manuel	Cadorna	5
88	Monica	Ravaglioli	5
90	Salvatore	Loredan	5
8	Stella	Tencalla	5
21	Lina	Antonelli	4
64	Sandra	Petrassi	4
3	Licia	Interminelli	4
67	Valentina	Scaramucci	4
98	Graziano	Cipolla	4
9	Venancio	Sgarbi	4
86	Gabriele	Benussi	4
36	Coriolano	Montalti	4
34	Tiziano	Bersani	4
10	Donna	Tassoni	4
87	Nedda	Leone	4
30	Giuliano	Cocci	4
14	Pomponio	Scarpa	4
95	Patrizia	Ravaglioli	4
89	Antonio	Boito	4
68	Agnolo	Gaiatto	4
76	Olga	Barese	3
72	Carmelo	Verga	3
52	Rosario	Segrè	3
78	Umberto	Broschi	3

Ottimizzazione

Di seguito abbiamo selezionato degli attributi su cui creare indici secondari per velocizzare l'esecuzione delle query.

Ovviamente, non abbiamo creato troppi indici per una questione di costi di memoria.

Gli indici occupano memoria e quindi abbiamo trovato un compromesso, creando indici solo per gli attributi più richiesti.

Creazione di indici in MySQL

Minor corse effettuate

```
CREATE INDEX idx_name
ON Personale(Nome);
```

```
CREATE INDEX idx_stip
ON Autisti(Stipendio);
```

```
CREATE INDEX idx_ut_star
ON Feedback(StelleUtente);
```

```
CREATE INDEX idx_ut_name
ON Utenti(Nome);
```

Utilizzando questi indici secondari, abbiamo la versione ottimizzata di alcune delle query descritte in precedenza, che vengono eseguite sui dati casuali generati dal programma Python. Riportiamo inoltre, la

frazione di miglioramento temporale delle ottimizzazioni.

Formula di miglioramento : $100 * (\text{originale-nuovo}) / \text{originale}$

Visualizza gli autisti con lo stipendio più alto

```
SELECT *
FROM Autisti
WHERE Stipendio =
(
    SELECT MAX(Stipendio)
    FROM Autisti
);
```

Prima della creazione dell'index sul campo "Stipendio", il tempo di esecuzione della query è il seguente

```
746 rows in set
Time: 0.063s
```

Dopo aver creato l'index, il tempo di esecuzione è il seguente:

```
746 rows in set
Time: 0.060s
```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{(0.063 - 0.060)}{0.063} \sim 4.7\%$$

Quindi, volendo arrotondare, abbiamo un miglioramento di circa il 5%

Visualizza il numero di feedback con almeno 3 stelle lasciati da ogni utente

```
SELECT u.Nome, u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
FROM Feedback f JOIN TratteCompletate tc ON f.ID_TrattaCompletata = tc.ID_TrattaC
JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
WHERE f.StelleUtente >= 3
GROUP BY u.Nome, u.Cognome
ORDER BY NumeroFeedback3Stelle DESC
```

Prima di aver creato l'index sul campo "StelleUtente", il tempo di esecuzione della query è il seguente

```
4126 rows in set
Time: 0.632s
```

Dopo aver creato l'index, il tempo di esecuzione risulta essere:

```
4126 rows in set
Time: 0.342s
```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{(0.632 - 0.342)}{0.632} \sim 45.9\%$$

Visualizza gli utenti che hanno effettuato almeno 10 richieste

```

SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroRichieste
FROM RichiestePrenotazioni rp JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
GROUP BY u.ID_Utente, u.Nome, u.Cognome
HAVING NumeroRichieste >= 10
ORDER BY NumeroRichieste DESC;

```

Prima di aver creato l'index sul campo "Nome", il tempo di esecuzione della query è il seguente

```

57 rows in set
Time: 0.161s

```

Dopo aver creato l'index, il tempo di esecuzione risulta essere:

```

57 rows in set
Time: 0.115s

```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{(0.161 - 0.115)}{0.161} \sim 28.6\%$$

Trova gli autisti che hanno completato il minor numero di corse in un determinato giorno

```

SELECT a.ID_Autista, p.Nome, p.Cognome, COUNT(*) AS NumeroCorseEffettuate
FROM TratteCompletate tc
JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
JOIN Autisti a ON rp.ID_Autista = a.ID_Autista
JOIN Personale p ON a.ID_Autista = p.ID
WHERE rp.DataRichiesta = "2023-06-05"
GROUP BY a.ID_Autista, p.Nome, p.Cognome
ORDER BY NumeroCorseEffettuate

```

Prima di aver creato l'index sul campo "Nome", dell'entità Personale, il tempo di esecuzione della query è il seguente

```

20 rows in set
Time: 0.032s

```

Dopo aver creato l'index, il tempo di esecuzione risulta essere:

```

20 rows in set
Time: 0.015s

```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{0.035 - 0.015}{0.035} \sim 57.15\%$$

Algebra Relazionale

L'algebra relazionale è un linguaggio query *procedurale* in notazione algebrica. In una query, si applicano sequenzialmente le operazioni alle relazioni. Ogni operazione (unaria o binaria) riceve in input una relazione e ne produce un'altra in output.

Le operazioni ***primitive*** sono:

- Selezione (σ)
- Proiezione (π)
- Unione (\cup)
- Differenza Insiemistica ($-$)
- Prodotto Cartesiano (X)
- Ridenominazione (ρ)

Esistono altre operazioni da esse derivabili, tra cui l'intersezione insiemistica (\cap).

Di seguito troviamo alcune query sul nostro database scritte in Algebra Relazionale:

Visualizza tutte le tratte completate che non hanno un feedback

```
SELECT tc.* FROM TratteCompletate tc
WHERE tc.ID_TrattaC NOT IN
(
    SELECT ID_TrattaCompleta FROM Feedback f
);
```

In algebra relazionale la query diventa

$$\pi_{tc.*}(\text{TratteCompletate} \bowtie (\pi_{ID_TrattaC}(\text{TratteCompletate}) - \pi_{ID_TrattaCompleta}(\text{Feedback})))$$

Visualizza tutti i dati di un determinato utente, comprese le carte a lui associate

```
SELECT u.*, c.NumeroCarta, c.DataScadenza, c.CVV
FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
WHERE Nome = "Geronimo" AND Cognome = "Lucarelli"
```

In algebra relazionale la query diventa:

$$\begin{aligned} & \text{Utenti} \bowtie_{ID_Utente=ID_Utente} \text{Carta} = A \\ & \sigma_{\text{Nome}='Geronimo', \text{Cognome}='Lucarelli'}(A) \end{aligned}$$

Dove:

- π rappresenta l'operazione di proiezione.
- σ rappresenta l'operazione di selezione.
- \bowtie rappresenta l'operazione di join.

L'operazione di join (\bowtie) viene eseguita sulla condizione $ID_Utente=ID_Utente$, e successivamente vengono selezionate le righe in cui $\text{Nome}=\text{"Geronimo"}$ e $\text{Cognome}=\text{"Lucarelli"}$, dopodiché viene applicata la proiezione sui campi specificati.

Per questioni di semplicità, abbiamo denominato con A tutta la parte del join

Visualizza tutti i veicoli la cui assicurazione scadrà entro febbraio 2024

```
SELECT Targa, Modello, Marca, a.DataScadenza FROM Veicoli v
JOIN Assicurazioni a
ON v.ID_Assicurazione = a.ID_Assicurazione
WHERE YEAR(a.DataScadenza) = "2024" AND MONTH(a.DataScadenza) = "02";
```

In algebra relazionale la query diventa:

$$\text{Veicoli} \bowtie_{\text{ID_Assicurazione} = \text{ID_Assicurazione}} \text{Assicurazioni} = A$$

$$\pi_{\text{Targa}, \text{Modello}, \text{Marca}, \text{DataScadenza}}(\sigma_{\text{DataScadenza} < '2024-02-01'}(A))$$

Visualizza tutti gli autisti che hanno una certa categoria di patente

```
SELECT p.Nome, p.Cognome, pt.Categoria
FROM Personale p JOIN Autisti a ON p.ID = a.ID_Autista
JOIN Patente pt ON a.NumeroPatente = pt.NumeroPatente
WHERE pt.Categoria = "B96"
```

In algebra relazionale diventa:

$$\text{Personale} \bowtie_{\text{ID} = \text{ID_Autista}} (\text{Autisti} \bowtie_{\text{NumeroPatente} = \text{NumeroPatente}} \text{Patente}) = A$$

$$\pi_{\text{Nome}, \text{Cognome}, \text{Categoria}}(\sigma_{\text{Categoria} = 'B96'}(A))$$

Trova tutti gli autisti che non hanno mai effettuato una richiesta di manutenzione

```
SELECT A./*
FROM Autisti A
WHERE A.ID_Autista NOT IN
(
    SELECT cpg.ID_Autista FROM ContattaPerGuasto cpg
);
```

In algebra relazionale la query diventa

$$\pi_{A.*}(\text{Autisti} \bowtie (\pi_{\text{ID_Autista}}(\text{Autisti}) - \pi_{\text{ID_Autista}}(\text{ContattaPerGuasto})))$$

Visualizza le tratte complete con un certo tipo di veicolo

```
SELECT TC.* , v.Marca, a.ID_Autista
FROM TratteCompletate TC
JOIN RichiestePrenotazioni rp ON TC.ID_TrattaC = rp.ID_Richiesta
JOIN Autisti a ON rp.ID_Autista = a.ID_Autista
JOIN Veicoli v ON a.Targa = v.Targa
WHERE v.Marca = 'Seat';
```

In algebra relazionale diventa

$$\text{TratteCompletate} \bowtie_{\text{ID_TrattaC} = \text{ID_Richiesta}} \text{RichiestePrenotazioni} \bowtie_{\text{ID_Autista} = \text{ID_Autista}} \text{Autisti} = A$$

$$\pi_{\text{TC}.*, \text{v.Marca}, \text{a.ID_Autista}}(\sigma_{\text{Marca} = 'Seat'}(A \bowtie_{\text{Targa} = \text{Targa}} \text{Veicoli}))$$

Per comodità abbiamo raggruppato tutti i join nella variabile A, per poi effettuare l'ultimo join partendo da A

Calcolo Relazionale

Il calcolo relazionale è un linguaggio query non procedurale ma *dichiarativo*. Invece dell'algebra, utilizza il calcolo dei predicati matematici del primo ordine in notazione logica. L'output di una query è una relazione che contiene solo tuple che soddisfano le formule logiche espresse. Il potere espressivo del calcolo relazionale è dunque equivalente a quello dell'algebra relazionale.

Versioni:

1. Calcolo relazionale sui domini
2. *Calcolo relazionale sulle tuple con dichiarazione di range*

Di seguito sono alcune query espresse tramite il *calcolo relazionale* sulle tuple con dichiarazione di range:

Visualizza tutte le tratte completate che non hanno un feedback

$$p = \text{tc.ID_TrattaCompletata} \in \text{tc} \wedge \text{tc.ID_TrattaCompletata} \notin f \\ \{\text{tc.*} \mid \text{tc}(\text{TratteCompletate}), f(\text{Feedback}) \mid p\}$$

Visualizza tutti i dati di un determinato utente, comprese le carte a lui associate

$$p = \{(u.\text{Nome} = 'Geronimo' \wedge u.\text{Cognome} = 'Lucarelli') \wedge (c.\text{ID_Utente} = u.\text{ID_Utente})\} \\ \{u.* \mid c.(\text{NumeroCarta}, \text{DataScadenza}, \text{CVV}) \mid u(\text{Utenti}), c(\text{Carta}) \mid p\}$$

Visualizza tutti i veicoli la cui assicurazione scadrà entro febbraio 2024

$$p = \{(a.\text{DataScadenza} < '2024-01-02') \wedge (v.\text{ID_Assicurazione} = a.\text{ID_Assicurazione})\} \\ \{v.(\text{Targa}, \text{Modello}, \text{Marca}), a.(\text{DataScadenza}) \mid v(\text{Veicoli}), a(\text{Assicurazione}) \mid p\}$$

Visualizza tutti gli autisti che hanno una certa categoria di patente

$$p = \{(pt.\text{Categoria} = 'B96' \wedge (p.\text{ID} = a.\text{ID_Autista}) \wedge a.\text{NumeroPat} = p.\text{NumeroPat})\} \\ \{p.(\text{Nome}, \text{Cognome}), pt.(\text{Categoria}) \mid p(\text{Personale}), pt(\text{Patenti}), a(\text{Autisiti}) \mid p\}$$

Trova tutti gli autisti che non hanno mai effettuato una richiesta di manutenzione

$$p = \{a.\text{ID_Autista} \in a \wedge a.\text{ID_Autista} \notin cpg\} \\ \{a.* \mid a(\text{Autista}), cpg(\text{ContattaPerGuasto}) \mid p\}$$

Visualizza le tratte completate con un certo tipo di veicolo

$$p = \{(tc.\text{ID_TrattaC} = rp.\text{ID_Richiesta}) \wedge (a.\text{Id_Aut} = rp.\text{Id_Aut}) \wedge (a.\text{Targa} = v.\text{Targa}) \\ \wedge v.\text{Marca} = 'Seat'\} \\ \{tc.* \mid v(\text{Marca}), a.(\text{Id_Aut}) \mid tc.(\text{TratCompl}), v(\text{Veicoli}), a(\text{Autisti}), rp(\text{RichPren}) \mid p\}$$

Sicurezza

Ovviamente in un database aziendale devono essere presenti diverse tipologie di utenti con diversi diritti, nella nostra modellizzazione della realtà, infatti, abbiamo definito 2 classi di utenti:

- un amministratore che ha tutti i diritti
- gli autisti, gli addetti marketing e i manutentori che possono aggiungere righe e fare query

Inoltre, si è definito un terzo utente che ha accesso solamente a delle view in modalità lettura, questo perché non gli si vuole dare accesso alle tabelle originali per questioni di sicurezza. Ovviamente la creazione di questo ultimo utente ha il solo fine dimostrativo e non sarebbe effettivamente inserito in un progetto reale.

Le view sono tabelle che non memorizzano dati, esse condividono lo stesso spazio delle tabelle originali. Spesso vengono assegnate ad altri utenti con specifici campi oscurati anche se il loro utilizzo inappropriato può portare all'inconsistenza del database.

Views

Visualizza tutti gli utenti che hanno almeno 2 carte associate

```
CREATE VIEW CartePerUtente AS
(
    SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroCarteAssociate
    FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
    GROUP BY u.ID_Utente, u.Nome, u.Cognome
    HAVING NumeroCarteAssociate > 2
```

```

        ORDER BY NumeroCarteAssociate DESC
    )

```

```

MySQL lfn@160.80.216.209:VroomA> CREATE VIEW CartePerUtente AS
    -> (
    ->     SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroCarteAssociate
    ->     FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
    ->     GROUP BY u.ID_Utente, u.Nome, u.Cognome
    ->     HAVING NumeroCarteAssociate > 2 ORDER BY NumeroCarteAssociate DESC
    -> );

```

Query OK, 0 rows affected

Time: 0.010s

```

MySQL lfn@160.80.216.209:VroomA> select * from `CartePerUtente`
+-----+-----+-----+-----+
| ID_Utente | Nome | Cognome | NumeroCarteAssociate |
+-----+-----+-----+-----+
| 74 | Ottavio | Garzoni | 7
| 38 | Angelina | Avogadro | 7
| 56 | Orlando | Tartaglia | 6
| 32 | Paloma | Scarpetta | 6
| 97 | Ermanno | Fagiani | 6
| 12 | Mario | Colletti | 6
| 26 | Lidia | Brenna | 6
| 69 | Alina | Campise | 6
| 25 | Romana | Trapanese | 5
| 47 | Leopoldo | Binaghi | 5
| 46 | Manuel | Cadorna | 5
| 88 | Monica | Ravaglioli | 5
| 90 | Salvatore | Loredan | 5
| 8 | Stella | Tencalla | 5
| 21 | Lina | Antonelli | 4
| 64 | Sandra | Petrassi | 4
| 3 | Licia | Interminelli | 4
| 67 | Valentina | Scaramucci | 4
| 98 | Graziano | Cipolla | 4
| 9 | Venancio | Sgarbi | 4
| 86 | Gabriele | Benussi | 4
| 36 | Coriolano | Montalti | 4
| 34 | Tiziano | Bersani | 4
| 10 | Donna | Tassoni | 4
| 87 | Nedda | Leone | 4
| 30 | Giuliano | Cocci | 4
| 14 | Pomponio | Scarpa | 4
| 95 | Patrizia | Ravaglioli | 4
| 89 | Antonio | Boito | 4
| 68 | Agnolo | Gaiatto | 4
| 76 | Olga | Barese | 3
| 72 | Carmelo | Verga | 3
| 52 | Rosario | Segrè | 3
| 78 | Umberto | Broschi | 3
+-----+-----+-----+-----+

```

Visualizza il numero di feedback con almeno 3 stelle lasciati da ogni utente

```

CREATE VIEW NumeroFeedbackTreStelle AS
(
    SELECT u.Nome, u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
    FROM Feedback f JOIN TratteCompletate tc ON f.ID_TrattaCompleta = tc.ID_TrattaC
    JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
    JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
    WHERE f.StelleUtente >= 3
    GROUP BY u.Nome, u.Cognome
    ORDER BY NumeroFeedback3Stelle DESC
)

```

```

MySQL lfn@160.80.216.209:VroomA> CREATE VIEW NumeroFeedbackTreStelle AS
-> (
->   SELECT u.Nome, u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
->   FROM Feedback f JOIN TratteCompletate tc ON f.ID_TrattaCompletata = tc.ID_TrattaC
->   JOIN RichiestePrenotazioni rp ON tc.ID_TrattaC = rp.ID_Richiesta
->   JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
->   WHERE f.StelleUtente >= 3
->   GROUP BY u.Nome, u.Cognome
->   ORDER BY NumeroFeedback3Stelle DESC
-> );
Query OK, 0 rows affected
Time: 0.021s

```

Nome	Cognome	NumeroFeedback3Stelle
Greca	Giammusso	9
Maurilio	Rocabbonella	9
Carla	Raimondi	8
Irma	Bernardi	8
Maria	Altera	8
Rodolfo	Ottino	8
Laureano	Branciforte	8
Cirillo	Accardo	8
Bianca	Bramante	8
Riccardo	Chigi	7
Gioele	Trevisani	7
Gabriele	Montesano	7
Giada	Farina	7
Tonino	Bertolucci	7
Luchino	Faloppio	7
Gustavo	Mantegazza	7
Gianfrancesco	Doria	7
Ermenegildo	Sforza	7
Cipriano	Farnese	7
Giulia	Molesini	7
Cristina	Palmisano	7
Clelia	Moschino	7
Uberto	Leoncavallo	7
Leopoldo	Guariento	7
Amalia	Bembo	7
Baccio	Corcos	7
Sophia	Eco	6
Corrado	Lamborghini	6
Allegra	Maglio	6
Raffaele	Righi	6
Antonina	Toscani	6

Creazione Utenti

Poiché il progetto rappresenta una realtà aziendale di una società, abbiamo creato 3 classi di utenti in ordine decrescente di grado di privilegi. Un amministratore è colui che gestisce il database e quindi ha tutti i diritti. Il personale ha il diritto di inserire nuove tuple e di effettuare query ai fini lavorativi. Gli utenti sono autorizzati ad effettuare query e inserire nuove tuple, in quanto usufruiscono del database solo ai fini di prenotare le corse. Infine, abbiamo creato anche un generico utente autorizzato solo ad interrogare le view esistenti.

1. **Amministratore:** ha tutti i diritti

```

CREATE USER 'administrator'@'localhost' IDENTIFIED BY 'adminpassword';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'administrator'@'localhost';
GRANT ALL ON VroomA.* TO 'administrator'@'localhost';

```

2. **Personale:** Può fare query e inserire tuple

```

CREATE USER 'personale'@'localhost' IDENTIFIED BY 'perspassword';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'personale'@'localhost';
GRANT SELECT ON VroomA.* TO 'personale'@'localhost';
GRANT INSERT ON VroomA.* TO 'personale'@'localhost';

```

3. Creazione di un utente che ha il solo diritto di eseguire le view sopra scritte

```
CREATE USER 'lfn'@'localhost' IDENTIFIED BY 'lfnpassword';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'lfn'@'localhost';
GRANT SELECT ON CartePerUtente TO 'lfn'@'localhost';
GRANT SELECT ON NumeroFeedbackTreStelle TO 'lfn'@'localhost';
```