Progetto Basi di Dati

Dal Farra Christian

26 febbraio 2025

Indice

Presentazione progetto

Un agenzia di booking vuole realizzare una base di dati per la gestione della propria attività, in cui gli utenti tramite username e password possono accedere alla piattaforma, inoltre nel profilo dell'utente sarà presente un email e un numero di telefono per il contatto.
Una volta fatto l'accesso gli utenti possono scegliere tra i seguenti servizi:
gestire la prenotazione di un volo di cui ci interessa la compagnia aerea, data di partenza, l'aereoporto di partenza e destinazione, i rispettivi orari di partenza e arrivo, il numero di posti disponibili e il prezzo, una volta fatta la prenotazione viene generato il biglietto di cui ci interessa il gate di imbarco, nominativo del passeggiero e il "numero" del sedile, un utente può possedere più biglietti realtivi allo stesso volo basta pero che essi abbiano nominativo diverso
L'utente potrà anche prenotare un soggiorno in hotel di cui ci interessa il nome, la locazione (via e citta), la disponibilità degli locali (numero di stanze libere), le stanze hanno un codice, una "capacità" (numero di posti letto), un prezzo a notte fissato la stanza, ci servirà capire se la stanza è disponibile inoltre l'hotel può avere due tipi di camere quelle base e quelle luxury con degli optional, la prenotazione ha una data di inizio e di fine soggiorno
L'utente potrà inoltre noleggiare delle automobili di cui ci interessa il modello, la citta nella quale sono disponibili, il prezzo al giorno, posti passeggiero, la targa e se la macchina è disponibile o meno, del noleggio ci interessa salvare la data di ritiro della macchina e di restituzione

Suddivisione frasi

Frasi utente

Un utente può accedere alla piattaforma tramite username e password, inoltre nel profilo ci saranno anche un email e un numero di telefono per il contatto

Frasi volo

Del volo ci interessa il codice del volo, compagnia aerea, la data di partenza, l'aereoporto di partenza e l'aereoporto di destinazione e i rispettivi orari di partenza e arrivo, il numero di posti e il prezzo

Frasi biglietto

Fatta la prenotazione verrà generato un biglietto del quale ci interessano nominativo, il gate di imbarco e numero di posto

Frasi hotel

Un utente può prenotare un soggiorno in hotel, dei quali siamo interessati ad avere il nome, la sua locazione (citta/paese e via) e il numero di stanze libere, della prenotazione salveremo la data di entrata e di uscita della stanza

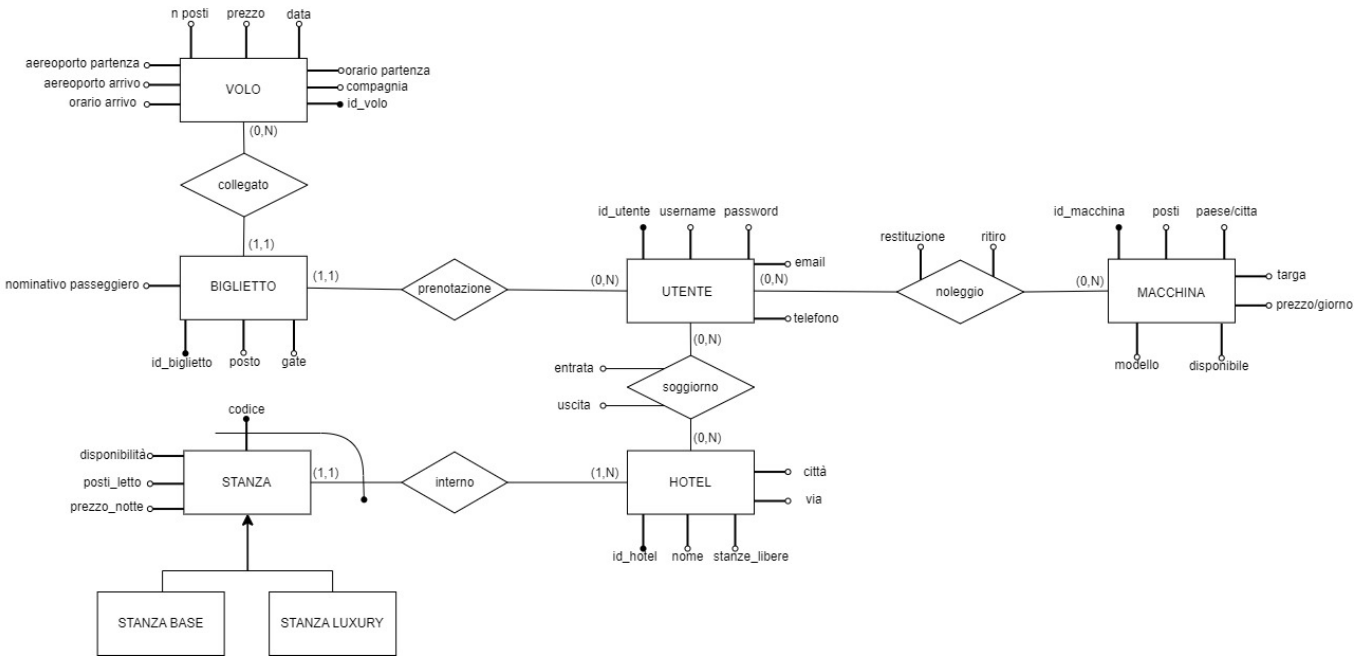
Frasi stanza

le stanze hanno un codice, una "capacità" (numero di posti letto), un prezzo a notte fissato la stanza, ci servirà capire se la stanza è disponibile inoltre l'hotel può avere due tipi di camere quelle base e quelle luxury con degli optional

Frasi automobile

Un utente potrà noleggiare un automobile di cui ci interessail modello, la città in cui sono, se sono disponibili, il prezzo giornaliero, posti passeggiero, la targa e se la macchina è disponibile o meno del noleggio salveremo la data di ritiro della macchina e di restituzione

Schema Entità-Relazione



Dizionario dei dati

Entità

Entità	Descrizione	Attributi	Identificatore
Utente	utente della piattaforma	id utente, username, password, email, telefono	id utente
Volo	voli disponibili	ID volo, compagnia, aeroporto partenza aeroporto di arrivo, numero posti, data prezzo, orario arrivo orario partenza	ID volo
Biglietto	biglietto di accedere al volo	id_biglietto, gate posto, nominativo	id_biglietto
Hotel	hotel associati	id hotel, nome, via, città stanze libere	id hotel
Stanza	stanze degli hotel	codice, posti letto prezzo per notte, disponibilità	codice, interno(R)
Base	stanze senza optional	codice, posti letto prezzo per notte, disponibilità	codice, interno(R)
Luxury	stanze con optional	codice, posti letto prezzo per notte, disponibilità	codice, interno(R)
Macchina	macchine accessibili	id macchina, posti, citta, targa, prezzo al giorno, disponibile, modello	id macchina

Relazioni

Relazione	Descrizione	Componenti	Attributi
Soggiorno	prenotazione verso un hotel	Utente, Hotel	entrata, uscita
Interno	stanze appartenenti a un hotel	Hotel, Stanza	
Noleggio	noleggio di una macchina	Utente, Macchina	ritiro, restituzione
Prenotazione	acquisto di un biglietto	Biglietto, Utente	
Collegato	biglietti relativi a un volo	Biglietto, Volo	

Vincoli non esprimibili

Non ci possono essere due biglietti con lo stesso nominativo collegati allo stesso volo, non ci possono essere due biglietti collegati allo stesso volo con lo stesso posto, il numero di biglietti generati non può essere superiore al numero di posti del volo
Nel soggiorno quando viene fatta una prenotazione la data di entrata dev'essere più "avanti" della data corrente in cui stiamo prenotando e la data di uscita dev'essere maggiore alla data di entrata
Per le macchine durante la prenotazione la data di ritiro della macchina dev'essere maggiore di quella corrente e la data di restituzione può essere maggiore o uguale (in caso di prenotazione per un giorno, ritiro sarà uguale a restituzione)
Nel volo l'aeroporto di partenza dev'essere diverso da quello di arrivo

Azioni interessanti

Azione	Tipo	Frequenza
Aggiornamento numero stanze libere	Batch	24/giorno
Aggiornamento macchine libere	Batch	96/giorno
Aggiornamento posti liberi sui voli	Batch	96/giorno
Inserimento nuova macchina	Interattiva	4/anno
Ricerca volo per data e destinazione	Interattiva	48/giorno
Ricerca hotel in base alla città	Interattiva	96/giorno

Tabella dei volumi

Le stime dei valori sono basate sul fatto che il database tenga comunque uno storico, dei voli degli ultimi 2 anni ho trovato che son circa 100.000 voli al giorno, quindi più o meno 360*100.000*2 = 70.000.000, per i biglietti ho contato circa 20 biglietti per volo quindi 1.400.000.000 (sempre considerando un po di storico), per gli utenti ho cercato un po di stime e alla fine ho ritenuto che cerca 2 milioni di persone utilizzino app di booking, per gli hotel ne ho stimati circa 800.000, quindi se circa tutti hanno una ventina di stanze 800.000*20 = 16.000.000, per le macchine sempre una stima molto approssimativa di 250.000

Concetto	**Tipo**	**Volume**
Utente	E	2.000.000
Volo	E	70.000.000
Biglietto	E	1.400.000.000
Hotel	E	800.000
Stanza	E	16.000.000
Macchina	E	250.000
Collegato	R	1.400.000.000
Prenotazione	R	1.400.000.000
Soggiorno	R	2.000.000
Interno	R	16.000.000
Noleggio	R	2.000.000

Analisi delle rindondanze

Noto che c'è una rindondanza su Hotel nel quale c'è il campo stanze_libere che potrebbe essere dedotto dalla disponibilità delle stanze collegate all'hotel
Azioni rigurardanti hotel:

- Aggiornamento numero stanze libere, 24/giorno (una volta all'ora), azione automatica
- Ricerca hotel in base alla città (mostrerà solo quelle che hanno stanze disponibili), 96/giorno(ogni quarto d'ora), azione interattiva

Concetto	Tipo	Volume
Hotel	E	800.00
Stanza	E	16.000.000
Interno	R	16.000.000

Presenza di rindondanza

Operazione 1

Concetto	Costrutto	Accessi	Tipo
Stanza	Entità	1	L
Hotel	Entità	1	L
Interno	Relazione	30.000	L
Hotel	Entità	30.000	S

Operazione 2

Concetto	Costrutto	Accessi	Tipo
Hotel	Entità	1	L

Conto doppi gli accessi in scrittura perchè sono più lenti
Operazione 1: 120.000*24 = 2.880.000 accessi
Operazione 2: 1*96 = 96 accessi
Per una stima di circa 2.880.000 accessi ogni giorno

Assenza di rindondanza

Operazione 1

Concetto	Costrutto	Accessi	Tipo
Stanza	Entità	1	L
Interno	Relazione	800.000	L
Hotel	Entità	800.000	L

Operazione 2

Concetto	Costrutto	Accessi	Tipo
Hotel	Entità	1	L
Stanza	Entità	10.000	L
Interno	Relazione	10.000	L

Operazione 1: 1.600.000*24 = 38.400.000 accessi
Operazione 2: 20.000*96 = 1920.000 accessi (supponendo che nella stessa città ci siano 10.000 hotel)
Per una stima di circa 40.320.000 accessi ogni giorno

Posso dedurre che nonostante nella prima operazione vi siano 30.000 accessi in scrittura in più dovuti all'aggiornamento del valore delle stanze vuote, senza la rindondanza saremo costretti, non solo a scrivere i dati che cambiano, ma a controllarli tutti e ogni volta andare a leggere a che hotel è collegato quella stanza quindi 1.600.000 accessi in lettura, mentre per la seconda operazione ci è comodo perchè mostreremo solo gli hotel di una città con stanze disponibili e avendo gia il dato memorizzato in un attributo ci basterà un unico accesso ad hotel per ottenere tutto, invece togliendo questo attriibuto dovremo andare a vedere tutte le stanze collegate a un determinato hotel di una città che porterebbe a 20.000 accessi in più in lettura, quindi infine penso sia meglio mantenere questa rindondanza per comodità nell'aggiornamento e nella ricerca

Eliminazione delle generalizzazioni

E' presente una generalizzazione che permette di dividere le due specilizzazioni di stanze tra luxury e base, ma visto che dal punto di vista dell'applicazione queste si differenziano per il prezzo a notte (notevolemente più alto per le stanze luxury) e per il fatto che presentino degli optional nel caso delle luxury, non hanno infatti comportamenti speciali o relazioni esclusive quindi si può accorpare le generalizzazioni al padre aggiungendo un attributo booleano che indica se una stanza sia luxury o no

Accorpamento o partizionamento

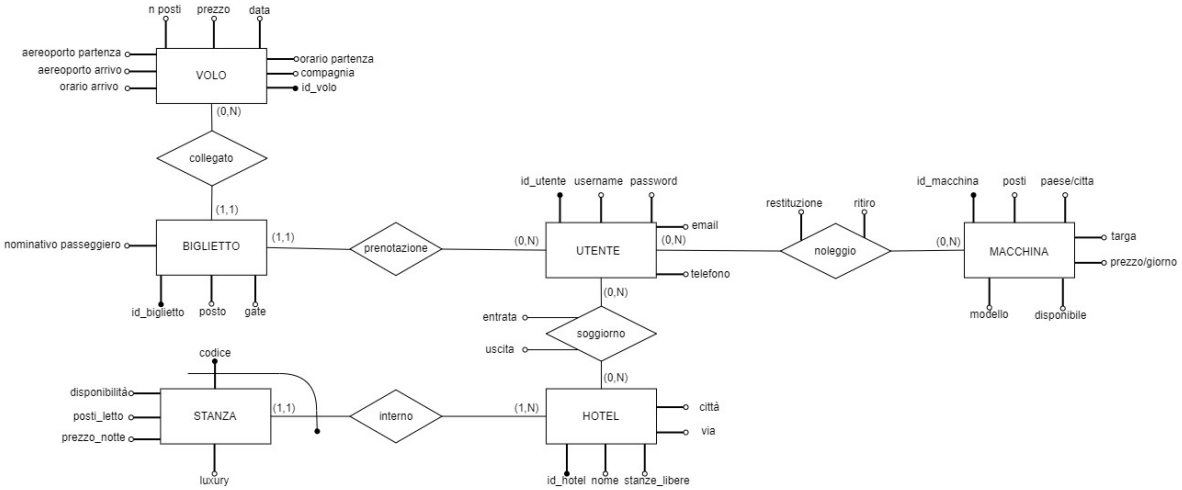
Il mio dubbio stava se nel dividere o meno la tabella volo in informazioni logistiche (aeroporto arrivo e partenza , orario arrivo e partenza) e volo (id, compagnia, numero posti, data e prezzo), ma guardando le azioni non avrò mai bisogno dei dati delle singole entità separate, quindi ho deciso di tenere tutto all'interno dell'entità volo

Scelta degli identificatori

Ho scelto i seguenti identificatori

- Utente: **id_utente**
- Volo: **id_volo**
- Biglietto: **id_biglietto**
- Hotel: **id_hotel**
- Stanza: **codice, id_hotel (esterno)**
- Macchina: **id_macchina**

Diagramma E-R ristrutturato

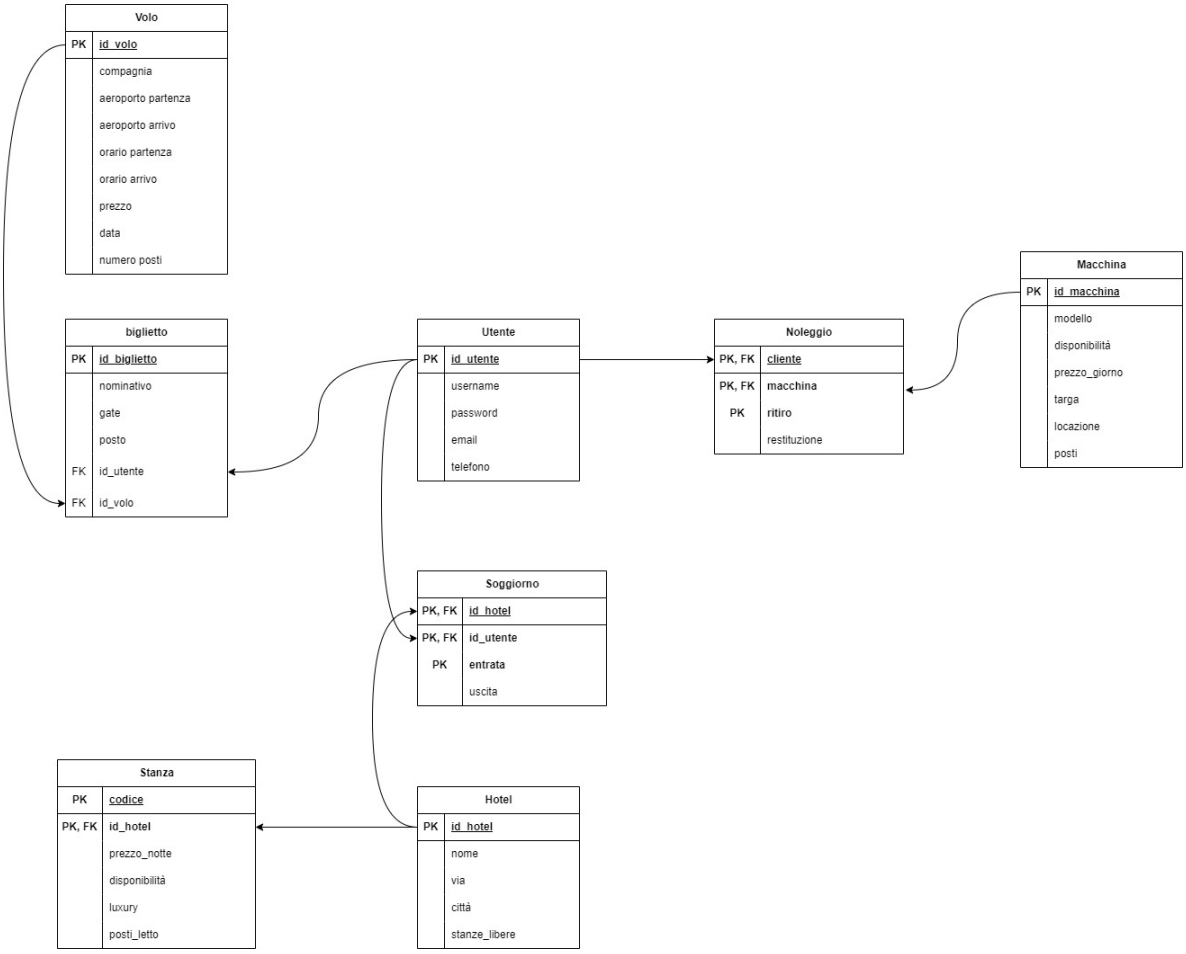


Passaggio allo schema logico

Utente(**id_utente**, username, password, email, telefono)
Soggiorno(**cliente (FK)**,**hotel (FK)**, **entrata**, uscita)
Hotel(**id_hotel**, nome, via, città, stanze libere)
Stanza(**codice**,**id_hotel**(FK), prezzo_notte, disponibilità, posti letto, luxury)
Biglietto(**id_biglietto**, nominativo, gate, posto, utente(FK), volo(FK))
Volo(**id_volo**, compagnia, aeroporto partenza, aeroporto arrivo, orario partenza, orario arrivo, prezzo, data, numero posti)
Macchina(**id_macchina**, modello, disponibilità, prezzo giorno, targa, locazione, posti)
Noleggio(**cliente**,**macchina**,**ritiro**,restituzione)

NB nella tabella soggiorno ho deciso di aggiungere anche entrata alla chiave primaria perchè altrimenti se un cliente faceva due prenotazioni sullo stesso hotel si veniva a creare un duplicato della coppia cliente e hotel, rendendola invalida come chiave primaria, stesso ragionamento per il noleggio visto che un utente potrà prenotare più volte la stessa macchina basta che non sia nella stessa data, quindi come chiave primaria avrò la terna cliente, macchina e data ritiro

Schema logico



Normalizzazione

Il database rispetta la prima forma normale poichè tutte le colonne sono atomiche non avendo attributi multivalore
Rispetta la seconda forma normale in quanto, dove ci sono chiavi composte (stanza, soggiorno e noleggio), la chiave non è separabile perchè in stanza diversi hotel potrebbero dare lo stesso codice per una stanza (puo esistere la stanza 1A in due hotel diversi, è inverosimile che gli hotel si mettano d'accordo sul nome delle stanze), in soggiorno pùò darsi che lo stesso utente prenoti sullo stesso hotel quindi ho deciso di differenziare in base alla data di entrata, non ha senso che un utente faccia più prenotazioni sullo stesso hotel per la stessa data e in noleggio stesso ragionamento l'utente potrebbe prenotare la stessa macchina basta che sia in date diverse per il ritiro
Rispetta anche la terza forma poichè non ho nessun campo calcolato e tutte le colonne sono indipendenti tra loro

Progettazione fisica

Visto che il database progetta di immagazzinare tante informazioni nel futuro e data l'operazione sugli hotel di ricerca in base alla città, si potrebbe impostare un indice secondario relativo al nome della città, cosi da far in modo che le ricerche siano molto più veloci avendo gli hotel della stessa città tutti vicini, inoltre si potrebbe mettere un indice secondario pure sull'aeroporto di destinazione nella tabella volo visto che le ricerche vengono fatte in base a quell'attributo

SQL relativo alle operazioni

Ricerca Hotel in base alla città(mostra solo quelli con stanze libere)

```
CREATE PROCEDURE searchHotelByCity(in cittaIn varchar(50))
BEGIN
    SELECT nome, via FROM hotel WHERE citta = cittaIn AND stanze_libere > 0;
END;
```

Ricerca volo in base alla data di partenza e all'aeroporto di destinazione

```
CREATE PROCEDURE searchFlyByDateAndDestination(in partenza date,
        in destinazione varchar(80))
BEGIN
    SELECT compagnia, prezzo, numero_posti, aeroporto_partenza,
        aeroporto_arrivo, orario_partenza
    from volo
    where data = partenza
        AND aeroporto_arrivo like concat('%',destinazione,'%');
END;
```

Aggiorno le disponibilità delle camere negli hotel

```
DELIMITER $$
CREATE PROCEDURE updateAvaibility()
BEGIN
    DECLARE finished INTEGER DEFAULT 0;

    DECLARE oldAvailability INTEGER;
    DECLARE newAvailability INTEGER;
    DECLARE idHotel INTEGER;

    DECLARE getHotelAvailability CURSOR
        FOR SELECT stanze_libere, id_hotel FROM hotel;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

    OPEN getHotelAvailability;

    onHotelLoop: WHILE (finished = 0) DO
            FETCH getHotelAvailability INTO oldAvailability, idHotel;

            IF (finished = 1) THEN
                LEAVE onHotelLoop;
            END IF;
            SELECT COUNT(*) INTO newAvailability
                FROM stanza WHERE hotel = idHotel;

            IF (oldAvailability != newAvailability) THEN
                UPDATE hotel SET stanze_libere = newAvailability
                    where id_hotel = idHotel;
            END IF;
    END WHILE;
    CLOSE getHotelAvailability;
END
$$
DELIMITER ;
```

Posti rimanenti su un volo dato dal codice

```
DELIMITER $$
CREATE PROCEDURE getFreeSeatsByFlyCode(in flyCode int, out free int)
BEGIN
    DECLARE occupied INTEGER DEFAULT 0;
    DECLARE numberOfSeats INTEGER DEFAULT 0;
    SELECT COUNT(*) INTO occupied
        FROM volo INNER JOIN biglietto ON volo.id_volo = biglietto.volo
        WHERE id_volo = flyCode;
    SELECT numero_posti INTO numberOfSeats FROM volo WHERE id_volo = flyCode;
    SET free = numberOfSeats - occupied;
END $$
DELIMITER ;
```

Aggiornamento disponibilità delle macchine (controlla se le disponibilità sono giuste in base alla data di prenotazione)

```
DELIMITER $$
CREATE PROCEDURE updateRental()
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE car INTEGER DEFAULT 0;
    DECLARE getCarOnRent CURSOR FOR SELECT macchina
            FROM noleggio n INNER JOIN macchina m ON n.macchina = m.id_macchina
            WHERE n.ritiro <= CURRENT_DATE AND n.restituzione > CURRENT_DATE;
    DECLARE getCarNotOnRent CURSOR FOR SELECT macchina
            FROM noleggio n INNER JOIN macchina m ON n.macchina = m.id_macchina
            WHERE n.ritiro >= CURRENT_DATE AND n.restituzione < CURRENT_DATE;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
    OPEN getCarOnRent;

    #IMPOSTO LE MACCHINE NOLEGGIATE COME NON DISPONIBILI
    WHILE (finished = 0) DO
            FETCH getCarOnRent INTO car;
            UPDATE macchina SET disponibilita = FALSE WHERE id_macchina = car;
    END WHILE;
    CLOSE getCarOnRent;

    #IMPOSTO LE MACCHINE FUORI NOLEGGIO COME DISPONIBILI
    OPEN getCarNotOnRent;
    WHILE (finished = 0) DO
            FETCH getCarNotOnRent INTO car;
            UPDATE macchina SET disponibilita = TRUE WHERE id_macchina = car;
    END WHILE;
    CLOSE getCarNotOnRent;
END $$
DELIMITER ;
```

Inserimento nuova macchina

```
CREATE PROCEDURE insertCar(in  new_modello varchar(50), in new_prezzo double,
    in new_targa varchar(7), in new_locazione varchar(80), in new_posti int)
    BEGIN
        insert into macchina(modello, prezzo_giorno, targa, locazione, posti)
        VALUE (new_modello, new_prezzo, new_targa, new_locazione, new_posti);
    END;
```

Trigger

Controllo che gli aeroporti di partenza e arrivo siano diversi

```
DELIMITER $$
CREATE TRIGGER checkVolo BEFORE INSERT ON volo FOR EACH ROW
    begin
        IF (NEW.aeroporto_arrivo <> NEW.aeroporto_partenza) THEN
            SIGNAL sqlstate '45001' SET message_text = "ERROR ON AEROPORT";
        END IF;
    end
    $$
DELIMITER ;
```

Controllo le date del soggiorno

```
DELIMITER $$
CREATE TRIGGER checkSoggiorno BEFORE INSERT ON soggiorno FOR EACH ROW
    BEGIN
        IF(NEW.entrata < CURRENT_DATE) THEN
            SIGNAL sqlstate '45001' SET message_text = "ERROR ON ENTRY DATE";
        END IF;
        IF(NEW.uscita < NEW.entrata) THEN
            SIGNAL sqlstate '45001' SET message_text = "ERROR ON LEFT DATE";
        END IF;
    END $$
DELIMITER ;
```

Controllo le date del noleggio

```
DELIMITER $$
CREATE TRIGGER checkNoleggio BEFORE INSERT ON noleggio FOR EACH ROW
    BEGIN
        IF(NEW.ritiro < CURRENT_DATE) THEN
            SIGNAL sqlstate '45001' SET message_text = "ERROR ON PICK UP DATE";
        END IF;
        IF(NEW.restituzione < NEW.ritiro) THEN
            SIGNAL sqlstate '45001' SET message_text = "ERROR ON RELEASE DATE";
        END IF;
    END $$
DELIMITER ;
```

Controllo se ci sono posti disponibili prima di generare il biglietto

```
DELIMITER $$
CREATE TRIGGER checkPosti BEFORE INSERT ON biglietto FOR EACH ROW
    BEGIN
        DECLARE freeSeats INTEGER DEFAULT 0;
        CALL getFreeSeatsByFlyCode(NEW.volo, freeSeats);

        IF(freeSeats <= 0) THEN
            SIGNAL sqlstate '45001'
            SET message_text = "NO MORE SEATS THE FLY IS FULL";
        END IF;
    END $$
DELIMITER ;
```

Aggiorno le macchine quando inserisco un noleggio

```
DELIMITER $$
CREATE TRIGGER updateNoleggio AFTER INSERT ON noleggio FOR EACH ROW
    BEGIN
        CALL updateRental();
    END $$
DELIMITER ;
```

Aggiorno le stanze libere quando aggiungo una nuova stanza

```
DELIMITER $$
CREATE TRIGGER onInsertHotelAvaibility AFTER INSERT ON stanza FOR EACH ROW
    BEGIN
        UPDATE hotel SET stanze_libere = stanze_libere + 1
            WHERE NEW.hotel = id_hotel;
        CALL updateAvaibility();
    END $$
DELIMITER ;
```

Aggiorno le stanze libere quando modifico una stanza

```
CREATE TRIGGER onUpdateHotelAvaibility AFTER UPDATE ON stanza FOR EACH ROW
    BEGIN
        CALL updateAvaibility();
    END
```

Create interessante

Metto questa create perchè permette di esprimere il vincolo dovuto al fatto che un nominativo non puo essere collegato a più posti dello stesso volo

```
CREATE TABLE biglietto (
    id_biglietto INT PRIMARY KEY AUTO_INCREMENT,
    nominativo VARCHAR(50) NOT NULL,
    gate VARCHAR(3) NOT NULL,
    posto VARCHAR(3) NOT NULL,
    utente INT,
    volo INT,
    FOREIGN KEY (utente) REFERENCES utente(id_utente) ON UPDATE CASCADE,
    FOREIGN KEY (utente) REFERENCES utente(id_utente) ON DELETE CASCADE,
    FOREIGN KEY (volo) REFERENCES volo(id_volo) ON UPDATE CASCADE,
    CONSTRAINT controlloVoloViaggiatore UNIQUE (volo, nominativo, posto)
);
```